Networked Applications
Outline

AL.1  Services, interfaces, and functions
AL.2  Application characteristics and types
AL.3  Information access applications
AL.4  Telepresence applications
AL.5  Distributed computing and networked storage
AL.6  Application adaptation and network interaction
AL.7  Social networking
Networked Applications

AL.1 Services, Interfaces, and Functions

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Application Layer
Application Primacy

• Application primacy
  – the sole and entire point of network infrastructure
  – is to support the distributed applications that need it

• Those of us who do networking for a living
  – need to remember the reason we do it
Layer 7: applications in data and control planes

“Layer 8” later
Application Layer

Application Association Definition

- **Application association**
  - uses *transport associations* between end systems
  - may be point-to-point or multipoint
  - the network is *translucent*: delay, errors, loss
    - depending on transport protocol these *may* be masked
• **Application protocol (E2E and A2A)**
  - is responsible coordinating networked applications
Application Layer
Service and Interfaces

- Application layer (L7) provides services to user
  - uses services of transport layer (L4) for E2E data transfer
  - may use services of session layer (L5) for coordination
  - may duplicate traditional network (L3) functions due to need for capabilities not provided by network
    - wide-area (interdomain) multicast
    - reliable multicast
    - P2P file lookup
Application Layer
Service and Interfaces

- **Application data unit** is A2A PDU
  - ADU – application data unit

![Diagram showing the relationship between application layer, transport layer, ADU, and application layer](image-url)
Application Layer
Functional Placement

- Application layer functionality only in end system
  - end system: host software
    or occasionally hardware accelerator
Networked Applications

AL.2  Application Characteristics and Types

AL.1  Services, interfaces, and functions

AL.2  Application characteristics
  AL.2.1  Delay
  AL.2.2  Bandwidth
  AL.2.3  Loss Tolerance
  AL.2.4  Flow characteristics and application categories

AL.3  Information access applications

AL.4  Telepresence applications

AL.5  Distributed computing and networked storage

AL.6  Application adaptation and network interaction

AL.7  Social networking
Application Characteristics

User Expectations

What performance metric matters to users?
Application Characteristics

User Expectations

What performance metric matters to users?

the only single metric?
Application Characteristics

User Expectations

• Users of applications care only about delay
  – end-to-end through the network
  – application processing
Application Characteristics

User Expectations

• Users of applications care only about \textit{delay}
  – end-to-end through the network
  – application processing

• Users of continuous media apps care about \textit{quality}
  – video: frame rate and resolution
  – audio: frequency range and distortion
  – note that if \textit{delay} is zero, all playout is local

\textit{Lecture MQ}
Application Characteristics
Network Performance Requirements

- Users expectations drive network performance
  - latency: directly from delay requirements
  - bandwidth: part of object transmission delay
  - error characteristics
    - can impact bandwidth and delay: why?
Application Characteristics

Network Performance Requirements

- Users expectations drive network performance
  - latency: directly from delay requirements
  - bandwidth: part of object transmission delay
  - error characteristics
    - can impact bandwidth and delay: retransmissions
Application Characteristics

AL.2.1 Delay

AL.1 Services, interfaces, and functions
AL.2 Application characteristics
  AL.2.1 Delay
  AL.2.2 Bandwidth
  AL.2.3 Loss Tolerance
  AL.2.4 Flow characteristics and application categories
AL.3 Information access applications
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Application Characteristics

Utility

- Performance requirements derived from *utility curve*
  - economic concept that measures usefulness of a product
  - vertical axis: *utility* to user in range \((0,1)\)
    - \(1 = \text{totally useful}\)
    - \(\theta = \text{completely useless}\)

\[
\begin{array}{c|c}
U & \text{usage} \\
\hline
1 & \text{very useful} \\
\theta & \text{worthless} \\
0 & \text{somewhat useful} \\
\end{array}
\]
Application Characteristics

Utility

- Performance requirements derived from *utility curve*
  - economic concept that measures usefulness of a product
  - vertical axis: *utility* to user in range \((0,1)\)
  - horizontal axis: how utility degrades w.r.t. performance
    - delay or bandwidth
Application Characteristics

Latency Requirements

- Latency requirements derived from delay utility curve
- Application types:
  - best effort
  - interactive
  - real-time
  - deadline
- These dictate:
  - transport layer service needed
  - traffic class and QoS

![Utility Curve Diagram](image)

- very useful
- somewhat useful
- worthless
Application Characteristics

Latency Requirements: Startup vs. Steady-State

• Some applications have distinct requirements for:
  – startup delay
  – steady-state delay

• Startup delay
  – delay to get application going
  – e.g. session or connection establishment

• Steady-state delay
  – delay during application operation of information transfer
  – e.g. delay for real-time communication
Application Characteristics

Latency: Best Effort

- Loose $d$ bound
  - delay insensitive
- Relative service
  - random
  - fair sharing or
  - based on policy
- *Examples?*
Application Characteristics

Latency: Best Effort

• Loose $d$ bound
  – delay insensitive

• Relative service
  – random
  – fair sharing or
  – based on policy

• Examples
  – email: user doesn’t care whether arrives in sec. or few min.
  – netnews, Internet fora
  – Facebook, twitter?
**Response time**

\[ T_r = t_r - t_a \]

- **\( t_a \)** user action
  - e.g. mouse click
- **\( t_r \)** usable response
  - e.g. 1st screen display

- **utility**
  - subsecond to avoid “click and wait”
  - ideally near 100ms (perceptually instantaneous)

**Examples?**
Application Characteristics

Latency: Interactive

- **Response time**
  \[ d = t_r - t_a \]
  - \( t_a \): user action
    - e.g. mouse click
  - \( t_r \): usable response
    - e.g. 1st screen display
  - utility
    - subsecond to avoid “click and wait”
    - ideally near 100ms (perceptually instantaneous)

- **Examples**
  - info access
  - Web browsing
Application Characteristics

Latency: Interactive Adaptation

- How to improve response time
  
  *for large Web pages?*
  
  *for pages with embedded images?*
Application Characteristics
Latency: Interactive Adaptation

- **Structured data**
  - presentation of partial data as it arrives (e.g. text)
    - don’t need to wait for un-needed data
  - presentation of smaller chunks in interactive response time
    - e.g. hierarchical structuring

- **Successive refinement**
  - presentation of progressively more information over time
  - each user can decide when enough is presented
  - supported by
    - structured data
    - layered and progressive coding
Application Characteristics
Latency: Real Time and Deadline

- **Hard real-time**
  - \( d \) bound
    - strict
    - small

- **Deadline**
  - \( d \) bound
    - relatively strict
    - large; allows adaptation well in advance of the deadline

- **Examples?**
  - real time:
  - deadline:

![Diagram showing latency and deadlines with U and D axes]
Application Characteristics
Latency: Real Time and Deadline

• Hard real-time
  – $d$ bound
    • strict
    • small

• Deadline
  – $d$ bound
    • relatively strict
    • large; allows adaptation well in advance of the deadline

• Examples
  real time:  process control, sensor with no memory, lifeline
  deadline:  remote backup
Application Characteristics

AL.2.2  Bandwidth

AL.1  Services, interfaces, and functions

AL.2  Application characteristics
   AL.2.1  Delay
   AL.2.2  Bandwidth
   AL.2.3  Loss Tolerance
   AL.2.4  Flow characteristics and application categories

AL.3  Information access applications

AL.4  Telepresence applications

AL.5  Distributed computing and networked storage

AL.6  Application adaptation and network interaction

AL.7  Social networking
Application Characteristics

Bandwidth Requirements

- Individual bandwidth
  - single application instance that needs significant bandwidth
  - example: interactive Web browsing needs ~ 1 Gb/s per user
    
    effect on network design?
Application Characteristics
Bandwidth Requirements

- Individual bandwidth
  - single application instance that needs significant bandwidth
  - example: interactive Web browsing needs ~ 1 Gb/s per user
  - dictates *end-system & per-flow* performance requirements

- Aggregate bandwidth
  - all instances of application use significant fraction bandwidth
    - more than several percent
  - examples: Web browsing, P2P file sharing, media streaming
  - example: HDTV streams to US households need \( \approx 1.5 \) Pb/s

*effect on network design?*
Application Characteristics

Bandwidth Requirements

- **Individual bandwidth**
  - single application instance that needs significant bandwidth
  - example: interactive Web browsing needs $\sim 1$ Gb/s per user
  - dictates *end-system* & *per-flow* performance requirements

- **Aggregate bandwidth**
  - all instances of application use significant fraction bandwidth
    - more than several percent
  - examples: interactive Web browsing, P2P file sharing
  - example: HDTV streams to US households need $\approx 1.5$ Pb/s
  - dictates *network* architecture and component requirements
Application Characteristics

AL.2.3 Loss Tolerance

AL.1 Services, interfaces, and functions
AL.2 Application characteristics
  AL.2.1 Delay
  AL.2.2 Bandwidth
  AL.2.3 Loss Tolerance
  AL.2.4 Flow characteristics and application categories
AL.3 Information access applications
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Application Characteristics

Loss Tolerance

- Loss or error tolerance
  
  *what is it and why important?*
Application Characteristics

Loss Tolerance

• Loss or error tolerance
  – the ability for an application to tolerate
    • data corruption (e.g. bit errors)
    • lost packets
      – congestion
      – discard due to corruption

*Examples?*
Application Characteristics

Loss Tolerance

• Loss or error tolerance
  – the ability for an application to tolerate data corruption

• Loss intolerant application examples
  – data transfer (document or file)
  – remote backup
  – stored multimedia

• Loss tolerant application examples
  – streaming media
Application Characteristics

AL.2.4  Flow Characteristics & App Categories

AL.1  Services, interfaces, and functions
AL.2  Application characteristics
   AL.2.1  Delay
   AL.2.2  Bandwidth
   AL.2.3  Loss Tolerance
   AL.2.4  Flow characteristics and application categories
AL.3  Information access applications
AL.4  Telepresence applications
AL.5  Distributed computing and networked storage
AL.6  Application adaptation and network interaction
AL.7  Social networking
Application Types
Flow Characteristics

- Individual bandwidth
- Delay
  - start/transient
  - steady state
  - latency budget
    - for high utility
- Loss tolerance
- Adaptability
  - ability to adapt to variations in bandwidth, delay, loss
### Application Types
#### Flow Characteristics

<table>
<thead>
<tr>
<th>Application Flow</th>
<th>Characteristic</th>
<th>Individual Bandwidth</th>
<th>Start/Transient Delay</th>
<th>Steady-State Delay</th>
<th>Latency Budget</th>
<th>Loss Tolerance</th>
<th>Adaptability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distributed computing</td>
<td></td>
<td>low–high</td>
<td>–</td>
<td>real-time</td>
<td>1µs–10ms</td>
<td>none</td>
<td>low</td>
</tr>
<tr>
<td>Process control</td>
<td></td>
<td>low</td>
<td>–</td>
<td>real-time</td>
<td>1µs–10ms</td>
<td>none</td>
<td>low</td>
</tr>
<tr>
<td>Haptics</td>
<td></td>
<td>very low</td>
<td>–</td>
<td>real-time</td>
<td>10 ms</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>Live interactive voice</td>
<td></td>
<td>low</td>
<td>interactive</td>
<td>real-time</td>
<td>30 ms</td>
<td>very low</td>
<td>limited</td>
</tr>
<tr>
<td>Live interactive video</td>
<td></td>
<td>med</td>
<td>interactive</td>
<td>real-time</td>
<td>300 ms</td>
<td>low</td>
<td>moderate</td>
</tr>
<tr>
<td>Stored streaming video</td>
<td></td>
<td>mod</td>
<td>interactive</td>
<td>–</td>
<td>1–10 s</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>Stored interactive video</td>
<td></td>
<td>mod</td>
<td>interactive</td>
<td>interactive</td>
<td>100 ms</td>
<td>low</td>
<td>moderate</td>
</tr>
<tr>
<td>Web browsing</td>
<td></td>
<td>med–high</td>
<td>interactive</td>
<td>–</td>
<td>100 ms – 1 s</td>
<td>none</td>
<td>moderate</td>
</tr>
<tr>
<td>Information push</td>
<td></td>
<td>low–med</td>
<td>push</td>
<td>–</td>
<td>1 min – 1 d</td>
<td>moderate</td>
<td>high</td>
</tr>
<tr>
<td>Telemetry</td>
<td></td>
<td>low–med</td>
<td>–</td>
<td>varies</td>
<td>varies</td>
<td>none</td>
<td>limited</td>
</tr>
<tr>
<td>Remote Backup</td>
<td></td>
<td>high</td>
<td>push</td>
<td>deadline</td>
<td>1 hour</td>
<td>none</td>
<td>high</td>
</tr>
<tr>
<td>email</td>
<td></td>
<td>low</td>
<td>push</td>
<td>best effort</td>
<td>1 min – 1 hr</td>
<td>very low</td>
<td>high</td>
</tr>
</tbody>
</table>
## Networked Applications

### Application Categories

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Information Access</th>
<th>Telepresence</th>
<th>Distributed Computing &amp; Networked Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application Relationship</td>
<td>client/server</td>
<td>peer-to-peer</td>
<td>varies</td>
</tr>
<tr>
<td>Bandwidth symmetry</td>
<td>asymmetric</td>
<td>symmetric</td>
<td>symmetric</td>
</tr>
<tr>
<td>Transfer granularity</td>
<td>large</td>
<td>continuous</td>
<td>varies</td>
</tr>
<tr>
<td>E2E Synchronisation</td>
<td>none</td>
<td>real time</td>
<td>varies</td>
</tr>
</tbody>
</table>

[NSF/DARPA/NIST 1994]
Application Categories

Information Access

- Client accessing information from a server
- Highly asymmetric bandwidth
- Response time is the important metric
  - 100 ms ≤ \( T_r \) ≤ 1 s target
- Significant bandwidth requirement
  - individual and aggregate
Application Categories

Information Access: Bandwidth Requirements

<table>
<thead>
<tr>
<th>Page Size [Byte]</th>
<th>Bandwidth / User [bit/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1B</td>
<td>1G</td>
</tr>
<tr>
<td>10B</td>
<td>10M</td>
</tr>
<tr>
<td>100B</td>
<td>100M</td>
</tr>
<tr>
<td>1k</td>
<td>1G</td>
</tr>
<tr>
<td>10k</td>
<td>10M</td>
</tr>
<tr>
<td>100k</td>
<td>100M</td>
</tr>
<tr>
<td>1M</td>
<td>1G</td>
</tr>
<tr>
<td>10MB</td>
<td>10M</td>
</tr>
<tr>
<td>10MB</td>
<td>100M</td>
</tr>
<tr>
<td>100MB</td>
<td>1G</td>
</tr>
</tbody>
</table>

- photo images
- images
- mixed page
- text page
- modem
- ISDN
- 10BaseT
- T-3
- T-1
- OC-12
- OC-3
- OC-48
Application Categories

Information Access: Server Push

- Reduce response time by pushing data to user
  - server knows what user is interested in
  - data already present when user requests
  - reduces peak bandwidth
Application Categories

Telepresence

• Peer-to-peer exchange of virtual presence
  – example: video conferencing
• Data streams with embedded synchronisation
• Relatively* symmetric bandwidth
  – may be relatively uniform, depending on codec
  *compared to information access
Application Categories

Distributed Processing and Networked Storage

- Distributed processing
  - distribution of computations beyond a room (>LAN)

- Networked storage
  - SAN: *storage area network* or *system area network*
  - NAS: *network attached storage* over LAN, MAN, or WAN
    - IP storage: protocols over Internet (e.g. iSCSI)

- Arbitrary exchange of control, data, state
  - application dependent symmetric bandwidth

- Application-dependent partitioning critical
Application Categories

Composed Applications

- Complex applications consist of multiple components
- Composition of
  - information access
  - telepresence
  - distributed computing
Application Categories
Composed Application Examples

- Complex applications consist of multiple components
- Composed application examples
  - distance learning
    - information access for class and reference materials
    - telepresence for student/teacher interaction
  - peer-to-peer file sharing
    - information access per file (except for swarming)
    - distributed computing for resource location
  - social networks (e.g. MySpace, Facebook)
    - information access: publish and access
    - telepresence for interactive communication
Networked Applications

AL.3 Information Access Applications

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
  AL.3.1 File transfer
  AL.3.2 Netnews and Internet fora
  AL.3.3 The Web
  AL.3.4 Peer-to-peer file sharing
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Networked Applications

Information Access

- Client/server

[Diagram of client and server]
Networked Applications

Information Access

- Client/server
  - client requests information
Networked Applications

Information Access

- Client/server
  - client requests information
Networked Applications

Information Access

- **Client/server**
  - client requests information
  - server responds
Networked Applications

Information Access

- Client/server
  - client requests information
  - server responds
Networked Applications

Information Access

- Client/server
  - client requests information
  - server responds

- Characteristics
  - granularity?
  - symmetry?
  - synchronisation?
Networked Applications

Information Access

- Client/server
  - client requests information
  - server responds

- Characteristics (typical)
  - *large granularity*: unit of information requested
  - *asymmetric bandwidth*: request control / response data
  - *end-to-end synchronisation*: little or none

*Components?*
Networked Applications

Information Access

- **Client/server**
  - client requests information
  - server responds

- **Characteristics (typical)**
  - *large granularity*: unit of information requested
  - *asymmetric bandwidth*: request control / response data
  - *end-to-end synchronisation*: little or none

- **Components**
  - client software: interfaces with user
  - server software: manages information store
  - client–server protocol
Information Access

AL.3.1 File Transfer

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
    AL.3.1 File transfer
    AL.3.2 Netnews and Internet fora
    AL.3.3 The Web
    AL.3.4 Peer-to-peer file sharing
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Information Access

File Transfer

- Moving files of information between computers
  - one of the first *killer apps* that drove demand for networks
    - file transfer, remote login, email

- Two modes of operation (data plane)
  - get, receive, or download a file
  - put, send, or upload a file

- Additional control operations
  - directory navigation and listing
  - file type specification, e.g. character (7-bit) vs. binary (8-bit)

- Examples
  - sendfile in BITNET (historic)
  - FTP in the ARPANET to Internet
File Transfer
FTP Overview

- **FTP**: file transfer protocol
  - client/server: client *gets* information from a server
    - may require user/password authorisation
    - may be *anonymous* login to public server
  - also provides a mechanism for client to *put* information
  - used for network file transfer and document access

- **Components**
  - client:
    - traditionally command line interface (CLI) FTP client
    - later GUI such as XFTP or web browser
  - server: FTP server software
  - client/server protocol: FTP protocol [RFC 0959 / STD 0009]
    - security extensions [RFC 2228] internationalisation [RFC 2640]
File Transfer
FTP Protocol Details

• Separated control and data connections
  – assumption: a number of commands for given FTP session

• Control connection
  – initiated by client by logon command
    • TCP connection opened using port 21
  – used for commands
  – connection maintained for multiple transfers
  – closed on client logout

• Data connection
  – established for directory or file transfer (dir, ls, get, put)
    • TCP connection opened using port 20
  – closed after each transfer (but multiple files for single mget)
File Transfer
FTP Protocol Example

$ ftp ftp.eecs.ku.edu
File Transfer
FTP Protocol Example

$ ftp ftp.eecs.ku.edu
Connected to tesla.eecs.ku.edu.
220 tesla.eecs.ku.edu FTP server (Version 6.00LS) ready.
User (tesla.eecs.ku.edu): jpgs
331 Password required for jpgs.
Password: ********
230 User jpgs logged in.
ftp>

magic ‘til lecture TL
File Transfer
FTP Protocol Example

$ ftp ftp.eecs.ku.edu
Connected to tesla.eecs.ku.edu.
220 tesla.eecs.ku.edu FTP server (Version 6.00LS) ready.
User (tesla.eecs.ku.edu): jpgs
331 Password required for jpgs.
Password: *******
230 User jpgs logged in.
ftp> cd public_html
250 CWD command successful.
ftp>
File Transfer

FTP Protocol Example

$ ftp ftp.eecs.ku.edu
Connected to tesla.eecs.ku.edu.
220 tesla.eecs.ku.edu FTP server (Version 6.00LS) ready.
User (tesla.eecs.ku.edu): jjpgs
331 Password required for jjpgs.
Password: *******
230 User jjpgs logged in.
ftp> cd public_html
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
total 53
-rw-r--r-- 1 jjpgs faculty 9709 Jul 25 2005 index.html
-rw-r--r-- 1 jjpgs faculty 42878 Jul 25 2005 jjpgs.gif
226 Transfer complete.
ftp>
File Transfer
FTP Protocol Example

$ ftp ftp.eecs.ku.edu
Connected to tesla.eecs.ku.edu.
220 tesla.eecs.ku.edu FTP server (Version 6.00LS) ready.
User (tesla.eecs.ku.edu): jpgs
331 Password required for jpgs.
Password: *******
230 User jpgs logged in.
ftp> cd public_html
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
total 53
-rw-r--r-- 1 jpgs faculty 9709 Jul 25 2005 index.html
-rw-r--r-- 1 jpgs faculty 42878 Jul 25 2005 jpgs.gif
226 Transfer complete.
ftp> get index.html
200 PORT command successful.
150 Opening ASCII mode data connection for index.html (9709 bytes).
226 Transfer complete.
9963 bytes received in 0.018 seconds 55.35Kbytes/sec.
ftp>
File Transfer
FTP Protocol Example

$ ftp ftp.eecs.ku.edu
Connected to tesla.eecs.ku.edu.
220 tesla.eecs.ku.edu FTP server (Version 6.00LS) ready.
User (tesla.eecs.ku.edu): jpgs
331 Password required for jpgs.
Password: *******
230 User jpgs logged in.
ftp> cd public_html
250 CWD command successful.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for '/bin/ls'.
total 53
-rw-r--r-- 1 jpgs faculty 9709 Jul 25 2005 index.html
-rw-r--r-- 1 jpgs faculty 42878 Jul 25 2005 jpgs.gif
226 Transfer complete.
ftp> get index.html
200 PORT command successful.
150 Opening ASCII mode data connection for index.html (9709 bytes).
226 Transfer complete.
9963 bytes received in 0.018 seconds 55.35Kbytes/sec.
ftp> quit
221 Goodbye.
Information Access
AL.3.2 Netnews and Bulletin Boards

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
  AL.3.1 File transfer
  AL.3.2 Netnews and Internet fora
  AL.3.3 The Web
  AL.3.4 Peer-to-peer file sharing
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Information Access

Netnews

- Netnews (network news)
  - users can read from and post to *newsgroups*
  - hierarchical structure of groups (e.g. comp.protocols.tcp-ip)
  - replaced dial-in bulletin board systems (BBS) of early 1980s
    - initially used USENET over UUCP

- Components:
  - client: news reader (with posting capabilities)
    - early: CLI such as *rn*
    - later: fullscreen text interface such as *trn*
    - later: GUI (e.g. Mozilla) or Web interface (e.g. Google groups)
  - server: news servers
  - protocol: NNTP (network news transfer protocol) [RFC 0977]
Information Access

Internet Fora

- Modern version of dial-in bulletin boards
  - using Web interface (HTML) and protocols (HTTP)
- Forum hosted on a web server
  - with one of many forum software packages (e.g. vbulletin)
- Users
  - post messages in HTML form
  - read messages in HTML pages
    - threaded messages and search capabilities typical
  - other features such as private messaging
- Administrators
  - user management and moderation
Information Access

AL.3.3 The Web

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
  AL.3.1 File transfer
  AL.3.2 Netnews and Internet fora
  AL.3.3 The Web
  AL.3.4 Peer-to-peer file sharing
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
Information Access

The Web

• World wide web (the Web)
  – global infrastructure for information access
  – has replaced most other access mechanisms
    • gopher, archie, etc.
  – browser replaced other remaining interfaces for most users
    • FTP, netnews

• Foundations
  – hypertext [Bush 1945, many others]
  – information access protocols (e.g. FTP)
  – Client GUI (e.g. xftp)
The Web
Architecture and Components

- Components:
  - client: Web browser e.g. Firefox
  - server: Web server e.g. Apache
  - client/server protocol: HTTP
  - supporting infrastructure
    - web cache, content distribution servers
The Web
Client Software

- Web browser: client software to
  - display access information
  - permit servers to store/access cookies and execute scripts
  - display retrieved information
Web Browsers
Text Based: Lynx

• Text-based Web browser: Lynx [lynx.isc.org]
  – originally developed at KU in 1992 by students Lou Montulli, Michael Grobe, & Charles Rezac
    • still useful for text-only clients and low-bandwidth connections
  – cross-platform support: Unix, MS-Windows, Mac OS, etc.
    • included in Cygwin release for MS-Windows

• Useful to text-only clients
  – and to test web pages for support of limited clients

• Good for accessibility
  – using text-speech converters

[demo]
Web Browsers

GUI Evolution: Mosaic

• GUI Web browser evolution: Mosaic tree
  – Mosaic: first popular graphical browser
    • developed 1993
    • at UIUC NCSA by Marc Andreessen & Eric Bina
      – University of Illinois at Urbana-Champaign
        National Center for Supercomputing Applications
    • funded by HPCC act (Gore bill)
  – decline
    • innovation lagged behind Netscape (e.g. tables)
    • development and support ended 1997
Web Browsers

GUI Evolution: Netscape

• GUI Web browser evolution: Mosaic → **Netscape**
  – Netscape Navigator: commercial evolution of Mosaic
    • founded by Marc Andreesen & Jim Clark in 1994
  – supported on multiple platforms: Unix, MS-Windows, MacOS
  – dominant browser in 1990s replacing Mosaic
    • innovative features, e.g. progressive display, plugins, Javascript
    • introduced non-standard-compliant features `<blink>`, `<layer>` ...
    • bloatware takes over with release 4: Netscape Communicator
GUI Evolution: Netscape

- Mosaic → Netscape
  - Netscape Browser 8.0 in 2005 based on Firefox
  - Decline: loss of market share to Microsoft IE
    - Netscape bought by AOL in 1998
    - 2003: AOL receives $750M antitrust settlement from Microsoft
      - Chooses IE, lays off most Netscape developers
      - Support for Netscape 9 (based on Firefox) ended in 2008
Web Browsers
GUI Evolution: Mozilla

• GUI Web browser evolution: 
  Mosaic → Netscape → Mozilla
  - Mozilla [www.mozilla.org]
    • codename, mascot, user agent name of Netscape
    • 1998: AOL/Netscape release browser to open source
      - internal spinoff of group to “Mozilla Organization”
    • 2002: Mozilla 1.0 browser released
      - Netscape continued to be released using Mozilla codebase
    • 2003: separated from AOL as Mozilla Foundation
Web Browsers
GUI Evolution: Firefox

- GUI Web browser evolution:
  Mosaic → Netscape → Mozilla → Firefox
  - Firefox browser: open source
    - rewrite of Mozilla to eliminate bloat
    - clean uncluttered interface
    - automatic software updates with better security than IE
      - fewer vulnerabilities, quicker patches by open source community
    - changed to rapid release policy
      - difficult for extension developers
GUI Web browser evolution:
Mosaic → Netscape → Mozilla → Firefox

- Flexible extension APIs: addons.mozilla.org
  - plugins (as with Netscape), e.g. Flash, Acrobat reader
  - search engines in dropdown menu
    - e.g. Google, Wikipedia, IMDB, Amazon
- extensions: some significantly enhance security, e.g.
  - No Script
  - Cookie Monster
  - Adblock Plus
  - Ghostery

- market share increasing significantly: ~31% in Jan. 2011
Web Browsers
MS Internet Explorer

- MS Internet Explorer
  - Microsoft browser for Windows platforms
    - developed in 1995 based on Spyglass Mosaic
    - brief previous support for Unix and Mac OS
  - overtook Netscape in market share in late 1990s
    - bundled beginning with Windows 95: default for most users
      - Microsoft (amazingly) claimed that Web browser was *part* of OS
    - resulted in antitrust action against Microsoft
  - now losing market share to Firefox and Chrome
    - 91% in 2004 → 10%
Web Browsers

MS Internet Explorer

- MS Internet Explorer
- Problems: IE notorious for...
  - poor security and slow fixes
  - exposure from ActiveX controls (more later)
  - vulnerabilities due to integration with other Windows apps
    - e.g. Outlook
    - poor standards compliance
- Replaced by Edge in 2015 for Windows 10
Web Browsers

Opera

- Opera
  - began as Telenor research project in 1994
  - spun off as separate company; software released 1996
    - initially for Windows; now cross-platform
    - originally trialware; now free for desktops/laptops
      - funding from Google and mobile clients [www.opera.com](http://www.opera.com)
  - 2% market share in Jan. 2011
    - low but stable market share over time

- Design goals
  - efficiency (speed) and standards compliance
  - includes bittorrent support
    - much to the annoyance of the RIAA and MPAA
Web Browsers

Amaya

- Amaya:
  - W3C browser and editor developed 1996
  - primarily intended for standards compliance and testing
- Also an *authoring tool*
  - generates HTML using editor
Web Browsers

Chrome

• Chrome
  – recent entrant from Google

• Design goals
  – efficiency
    • many would disagree; it is a resource hog on MacOS
  – security: sandboxed plugins

• Now a significant player
  – captured 5% market share in first year
    • 14% in Jan. 2011
    • prominent link on google.com search page
  – now 50% share (depending on methodology)
  – Google apps may be a significant threat to Microsoft
Web Browsers
Mac: Safari and Camino

• Safari
  – Apple Mac OS X browser developed 2003
    • packaged with Mac OS
    – and with iOS (iPhone, iPod touch, and iPad)
    • also runs on iPhone and iPod Touch
    • additionally supported on MS-Windows
  – market share low but stable: 8% in Jan 2011

• Camino
  – open source browser for Mac OS released 2002
  – based on Firefox Gecko rendering engine
  – Aqua Mac OS user interface
Web Browsers
Market Share Trends

- IE declining
  - EU-mandated browser selection in Windows 7
- Chrome
  - increasing
- Firefox
  - decreased
  - probably now stable
- Safari and Opera
  - low but stable
The Web
Web Pages and HTML

• Web page consists of
  – text information
  – embedded objects (e.g. images)
  – hyperlinks

• Key concept
  – hypertext: information itself may contain navigation
  – hyperlink to other web pages using URIs

• HTML (hypertext markup language)
  – rules for displaying content
  – standards maintained by W3C [www.w3.org/MarkUp]
  – current version: XHTML 1.1 (XML compliant HTML)
    • XHTML and HTML 5
The Web
Uniform Resource Identifiers

• **URI**: *uniform resource identifier* [RFC3986/STD0066]
  - uniform way of identifying a resource
    • electronic documents, bound (non-electronic) books, people ...
  - types: http://www.iana.org/assignments/uri-schemes.html
  - ⟨scheme⟩://⟨authority⟩/⟨path⟩?⟨query⟩#fragment

• Types of URIs
  - URLs
  - URNs
  - terminology has evolved and is frequently confused
The Web
Uniform Resource Identifiers: URLs and URNs

• **URL:** *uniform resource locator*
  – URI identified by access mechanisms and network location
    • *(protocol)://*(hostname)*(path):(port)?*(query)#*fragment
  – therefore http and ftp URI schemes are URLs
    • but not all URIs are URLs

• **URN:** *uniform resource name*
  – persistent global identifier within a namespace
  – does *not* specify location
  – e.g. issn & isbn URNs refer to content of periodical/book
    • not specific copy or location
The Web

Uniform Resource Identifiers: Examples

<table>
<thead>
<tr>
<th>Scheme</th>
<th>Use</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>ftp</td>
<td>file</td>
<td>ftp://ftp.ittc.ku.edu/</td>
</tr>
<tr>
<td>mailto</td>
<td>email address</td>
<td><a href="mailto:jpgs@eecs.ku.edu">mailto:jpgs@eecs.ku.edu</a>&lt;br&gt;<a href="mailto:jpgs@comp.lancs.ac.uk">mailto:jpgs@comp.lancs.ac.uk</a>&lt;br&gt;<a href="mailto:jpgs@sterbenz.org">mailto:jpgs@sterbenz.org</a></td>
</tr>
<tr>
<td>tel</td>
<td>PSTN address</td>
<td>tel:+15089443067&lt;br&gt;tel:+447903409203</td>
</tr>
<tr>
<td>telnet</td>
<td>remote terminal access</td>
<td>telnet://telnet.ittc.ku.edu</td>
</tr>
<tr>
<td>news</td>
<td>news group</td>
<td>news:comp.protocols.tcp-ip</td>
</tr>
<tr>
<td>isbn</td>
<td>book</td>
<td>urn:isbn:0471330361</td>
</tr>
</tbody>
</table>
The Web: XHTML

Example: Page Preamble and Header

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<title>KU EECS 780: Communication Networks</title>
<meta http-equiv="content-type" content="text/html;charset=UTF-8"/>
</head>
```

XML version and character set

XHTML doc type def

begin page header

page title

HTML text with UTF-8

end page header
The Web: XHTML
Example: Page Body

<body>
  <h1>KU EECS 780: Communication Networks</h1>
  <p>
    <a href="http://www.ku.edu">Univ. of Kansas</a>,
    <a href="http://www.eecs.ku.edu">Dept. EECS</a>,
    <a href="http://www.ittc.ku.edu/~jpgs/courses/nets">KU EECS 780: Communication Networks</a>
  </p>
  <h2>Instructor</h2>
  <address>
    <a href="http://www.ittc.ku.edu/~jpgs" title="James Sterbenz, KU academic page">Prof. James P.G. Sterbenz</a>
    &lt;<a href="mailto:jpgs@eecs.ku.edu">jpgs@eecs.ku.edu</a>&gt;
  </address>
  <h2>Time and Location</h2>
  ...
</body>
The Web: XHTML
Page Authoring

• Creating HTML
  – text editor such as Emacs (with HTML mode)
    • only way to control your own formatting *homework exercise*
  – Web browser with authoring mode
  – word processing program
  – Web development package
  – Web-based tools from Web-hosting services

• Checking HTML
  – try opening with Lynx (lynx.isc.org)
  – Web browser plugin (e.g. Firebug for Firefox)
  – W3C compliance checkers (e.g. validator.w3.org)
  – accessibility checkers (e.g. Total Validator)
The Web
Dynamic Content

• Dynamic content permits
  – Web pages to be modified or created on the fly
  – execution of code

• Location
  – client side (returned content modified or executed by client)
  – server side (content dynamically created by server)

• Techniques sometimes called DHTML
The Web: Dynamic Content

Scripting

- Scripting language
  - embedded in web page
  - stored in associated file
- JavaScript – original scripting language (Netscape)
- PHP (hypertext preprocessor)
  ... others
The Web: Dynamic Content

Scripting: JavaScript

- Scripting language
  - embedded in web page
  - stored in associated file

- JavaScript – original scripting language (Netscape)
  - not related to Java; name was a marketing gimmick
  - relatively safe but can be abused by servers
    - URL hiding and rebinding for *phishing* attacks
    - popup advertisements
  - modern browsers and personal firewalls control behaviour
    - but users need to set preferences
    - use script manager plugins (Firefox)
The Web: Dynamic Content
Program Execution and Java

- Server returns an *applet* to be executed on client
- Java *java.com*
  - developed by Sun Microsystems (not Oracle)
  - object-oriented language with restricted capabilities
  - relatively safe
    - operates in a restricted security *sandbox*
    - applet cannot write files on client computer
    - may consume significant resources in DOS attack
  - now platform independent
    - Unix, Linux, Windows, Mac OS
    - OSs for mobile devices such as PDAs and mobile phones
  - also used as general-purpose programming language
The Web: Dynamic Content

ActiveX

- ActiveX (Microsoft)
  - Windows-specific dynamic content
- Allows execution of arbitrary code
  - user must trust ActiveX code and server
  - certificates are used to authenticate code
- Beware of significant security risks!
  - frequent vector for malware and viruses
  - only Windows platforms at risk
    - third-party personal firewalls can be used to block
    - IE settings can restrict use
    - other browsers such as Firefox or Opera don’t support
The Web

Wikis

- Wiki (Hawaiian for “quick”)
  - server software and databases
  - interactive web authoring
- Directly editable web pages
  - simplified markup easier for novices
    - combination of wiki-specific and standard HTML
  - upload of HTML and style sheet files not needed
- Fosters collaboration
  - multiple users simultaneously edit different sections of page
  - full history maintained; ability to revert to earlier versions
The Web: Wikis

WikiMedia Markup Example

See http://meta.wikimedia.org/wiki/Help:Editing for full syntax

This page is for use by EECS 780 students at [http://www.ku.edu The University of Kansas].

Course information is maintained on a conventional HTML Web page &ndash; http://www.ittc.ku.edu/~jgps/courses/nets. "This page is only for Wiki exercises."

This page is only for Wiki exercises.

List of student pages in the Spring 2009 course offering.

See http://meta.wikimedia.org/wiki/Help:Editing for full syntax
The Web: Wikis
Servers and Software

- Multiple alternatives; no single standard
  - each has different API and markup language
- MediaWiki www.mediawiki.org
  - originally written for Wikipedia
  - Magnus Manske, Lee Daniel Crocker, Brion Vibber
  - open source, originally PERL now written in PHP
    - database uses binary files; relatively inflexible to administer
    - probably the most popular wiki; used by KU ITTC
- DocuWiki www.docuwiki.org
  - written in PHP, plain text internal files
  - many plugins (e.g. doodle for schedule)
The Web: Wikis

Wiki Application Examples

- Wikipedia en.wikipedia.org (English version)
  - free online encyclopedia anyone can edit
  - one of the few killer applications of the 2000’s
  - has developed its own subculture
  - many alternative, some parodies
    - e.g. Uncyclopedia unencyclopedia.wikia.com

- Wikis for project management and dissemination
  - e.g. wiki.ittc.ku.edu/resilinets

- Wikis to assist education
  - e.g. wiki.ittc.ku.edu/EECS780

*you will use this to create a simple wiki page*
The Web
Multimedia Content

• Multimedia downloads: conventional files download
  – then played using media player
    • e.g. open source VLC player videolan.org

• Play during download
  – some formats support playing during download
  – embedded in Web page with plugin, e.g. .flv
  – note: iPod and iPad do not support flash

• Multimedia streaming Lecture MS
  – separate streaming protocols play without download
  – knowledgeable users can rip from stream

• HTML5 now supports embedded streaming
The Web
Video Sharing Servers

• Video sharing servers
• Users upload videos
  – self produced
  – ripped from other sources
• Videos searchable (by metadata) for viewing
• Popular videos go *viral*
  – links shared by email or on social networking sites
• Sharing sites frequently target of RIAA and MPAA
  – DMCA takedown notices
    • for clear violations
    • for probable fair use: music in background of amateur videos
The Web

Video Server Examples

- Metacafe: online July 2003
- DailyMotion: online March 2005
- YouTube: online April 2005
  - not nearly the first, but became best known
  - bought by Google in 2006 for $1.56 billion
  - required Google id, but Google backing off

The Web
Video Server Examples: VOD

- Hulu VOD (video on demand):
  - publicly available March 2008
  - currently free with advertisements
    - paid subscription model being considered
  - IP addresses outside US and proxies blocked

- Netflix VOD
  - started in 1997 to deliver DVDs by mail
  - began digital deliver in 1999
  - billionth DVD delivered in 2007
  - 50 M subscribers in 2014
  - began to produce content in 2011
The Web
Mashups

• Web becomes defacto application infrastructure
  – Web browser is universal GUI

• **Mashup**
  – hybrid Web-based composed application
  – uses standard interfaces for rapid development, e.g.
    • Google maps
    • webcam stream embedded in page
    • streaming video images
  – iGoogle example of user-friendly mashup editor
    • like many Google services, it had been terminated
The Web
Web Complexity

• The current Web is a complex beast
  – static (X)HTML pages and style sheets
  – dynamically created web pages and interactive media
  – software programs and databases
  – location-based services and restrictions

• Web 2.0
  – dreadful term popularized by Tim O’Reilly in 2004
  – there is no such thing as a Web release number!
  – attempts to capture current capabilities and complexity
  – no precise definition
  – we may be stuck with it; used by marketing & popular press
The Web: HTTP
Protocol Overview

• HTTP (hypertext transfer protocol) [RFC 2616]
  – stateless (less server complexity)

• Uses TCP for end-to-end client–server transport
  – client initiates TCP connection (typ. port 80) on server
  – server accepts TCP connection from client
  – but TCP is not a transaction protocol!

• HTTP messages exchanged
  – HTTP GET (or POST) client → server
  – web page client ← server
    • typically HTML page

• TCP connection closed Lecture TL
The Web: HTTP
Connection Types: Non-Persistent

- Non-persistent HTTP
  - original version in HTTP 1.0
The Web: HTTP
Connection Types: Non-Persistent

- Non-persistent HTTP
  - original version in HTTP 1.0
  - HTTP GET to request page
The Web: HTTP
Connection Types: Non-Persistent

- Non-persistent HTTP
  - original version in HTTP 1.0
  - HTTP GET to request page
The Web: HTTP
Connection Types: Non-Persistent

- Non-persistent HTTP
  - original version in HTTP 1.0
  - HTTP GET to request page
  - at most one object per TCP connection
The Web: HTTP
Connection Types: Non-Persistent

- Non-persistent HTTP
  - original version in HTTP 1.0
  - HTTP GET to request page
  - at most one object per TCP connection
    - even if embedded objects
The Web: HTTP
Connection Types: Non-Persistent

- Non-persistent HTTP
  - original version in HTTP 1.0
  - HTTP GET to request page
  - at most one object per TCP connection
    - even if embedded objects
The Web: HTTP

Connection Types: Non-Persistent

- Non-persistent HTTP
  - original version in HTTP 1.0
  - HTTP GET to request page
  - at most one object per TCP connection
    - even if embedded objects

```plaintext
  SYN      SYNACK
  (ACK) GET

  page
  FIN      ACK
  SYN
  SYNACK
  (ACK) GET

  image_1
```

The Web: HTTP

Connection Types: Non-Persistent

- Non-persistent HTTP
  - original version in HTTP 1.0
  - HTTP GET to request page
  - at most one object per TCP connection
    - even if embedded objects

(disadvantages?)
The Web: HTTP
Connection Types: Non-Persistent

• Non-persistent HTTP
  – original version in HTTP/1.0
  – at most one object per TCP connection
    • even if embedded objects
  – significant overhead
    *two* round trip delays *per object*
    1. TCP connection setup
    2. HTTP transaction
    • TCP teardown does not block

*alternatives?*
The Web: HTTP
Connection Types: Parallel

- Non-persistent HTTP
- Parallel HTTP
  - multiple simultaneous connections per client

advantages? disadvantages?
The Web: HTTP

Connection Types: Parallel

- **Non-persistent HTTP**

- **Parallel HTTP**
  - multiple simultaneous connections per client
  - small number exploits multiplexing efficiency
  - too many create excessive overhead & congestion
    - typically limited by browser (about:config in Firefox)

_alternatives?_
The Web: HTTP

Connection Types: Persistent

- Non-persistent HTTP
- Parallel HTTP
- Persistent HTTP
  - multiple objects sent over single TCP connection
    - embedded objects such as images
    - successive pages from same server within timeout parameter

advantages? disadvantages?
The Web: HTTP
Connection Types: Persistent

- Non-persistent HTTP
- Parallel HTTP
- Persistent HTTP
  - multiple objects sent over single TCP connection
    - embedded objects such as images
    - successive pages from same server within timeout parameter
  - approaches 1 RTT per object
  - but each HTTP transaction waits for the previous alternatives?
The Web: HTTP
Connection Types: Pipelined

- Non-persistent HTTP
- Parallel HTTP
- Persistent HTTP
- Persistent HTTP with pipelining
  - multiple HTTP requests as soon as referenced
    - GETs are pipelined
    - responses are pipelined
    - server may multiplex responses
  - default in current HTTP/1.1
    - along with parallel connections initiated by browser
The Web: HTTP
Message Format

- Human-readable text
  - ASCII format
  - variable length fields
- Request line
  - HTTP request only
- Header lines
  - variable number
- Body
  - when object transferred

<table>
<thead>
<tr>
<th>method</th>
<th>$s_p$</th>
<th>URL</th>
<th>$s_p$</th>
<th>HTTP ver.</th>
<th>$c_R \ L_F$</th>
</tr>
</thead>
<tbody>
<tr>
<td>field-name</td>
<td>:</td>
<td>value</td>
<td>$c_R \ L_F$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

body
The Web: HTTP Requests

• Human-readable text: ASCII format
• Request method
  GET return web page
   – parameters may be sent in URL using form
     ?<name>&<value> ...
  POST send form input to server
   – parameters sent in body of request
     <name>=<value> ...
  HEAD return only header (not requested object)
  PUT upload file to path specified by URL field
  DELETE delete specified file
• Other parameters to negotiate request
The Web: HTTP
Get Request Example

GET /~jpgs/courses/nets HTTP/1.1
Host: ittc.ku.edu
Connection: close
Accept-Encoding: gzip
Accept: text/xml,application/xml,application/xhtml+xml,
text/html;q=0.9,text/plain;q=0.8,image/png,*/*;q=0.5
Accept-Language: en-us,en;q=0.5
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.7
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US;
   rv:1.8.0.1) Gecko/20060111 Firefox/1.5.0.1
[extra CRLF]
The Web: HTTP

Responses

• Response codes
  – 2nn successful
    • e.g. 200 OK body contains resource
  – 3nn redirection codes
  – 4nn client error codes
    • e.g. 404 Not found
  – 5nn server error codes

• Status information
  – time/date, server info, etc.

• Object metadata
  – length, content encoding, etc.
The Web: HTTP
Response Example

HTTP/1.1 200 OK
Date: Thu, 02 Feb 2006 17:52:46 GMT
Server: mod-xslt/1.3.8 Apache/2.0
Accept-Ranges: bytes
Content-Length: 27304
Connection: close
Content-Type: text/html; charset=ISO-8859-1

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <title>KU EECS 780: Communication Networks</title>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8" />
</head>
<body>
    ...
</body>
</html>
**The Web: HTTP**

**Examining Headers**

- Using telnet from command line interface
  - telnet to web server using port 80
    
    $telnet www.ittc.ku.edu 80
  
  - enter request
    
    GET /~jpegs/courses/nets/index.html HTTP/1.1
    Host: www.ittc.ku.edu
    [CRLF]
    [CRLF]

    *try it!*
The Web: HTTP
Examining Headers

- Using telnet from command line interface
  - telnet to web server using port 80
    
    ```
    $telnet www.ittc.ku.edu 80
    ```
  - enter request
    ```
    GET /~jpsgs/courses/eecs780/index.html HTTP/1.1
    Host: www.ittc.ku.edu
    [CRLF]
    [CRLF]
    ```

- Use web page, e.g.
  - http://web-sniffer.net
  - http://gemal.dk/browserspy

  - you should know what information your client is exposing!
The Web: Persistent State

Client vs. Server

- Persistent server state enables a number of services
  - user session state, e.g.
    - shopping carts
    - webmail interfaces
  - identification of users for preferences and recommendations
  - etc.

- State maintained either by:
  - server: *fat URLs* with per user id and parameters
    - e.g. `www.ittc.ku.edu/~jgps?userid=33958846&got_milk=yes`
  - client: cookies
The Web: Persistent State

Cookies

- Persistent client state maintained in *cookies*
  - stored in file on client computer
  - correlating information may be stored on server by cookie id

- Types
  - persistent cookie: kept until expiration date
  - session cookie: deleted when browser closed

- Cookies set by set-cookie2 message in response
  - e.g. Set-cookie: id="2235663"; domain="www.ku.edu"

- Cookies are sent as parameter in request header
  - e.g. Cookie: id="2235663"
The Web: Persistent State

Cookie Privacy Risks

- Cookies can be very useful
  - fat URLs unwieldy and require per session login

- Cookies are frequently abused
  - Web sites use them even if not needed
    - some web sites require them even if not needed
  - cookies can be used to track long-term behaviour
  - cookies can expose private information across multiple sites
    - e.g. email addresses, phone number, credit card information
  - cookies are frequently used for advertising

solution?
The Web: Persistent State

Cookie Privacy Risks

- Cookies are frequently abused
- Users should be selective in cookie use
  - set browser options to notify when a cookie is offered
  - permit/deny based on actual need and cookie domain
    - e.g. permit when shopping from trusted source (e.g. Amazon)
    - permit only for session when practical
    - deny advertising cookies even when allowing other cookies
  - browser options control this; don’t use defaults!
    - menu: tools → options → cookies
    - √ Allow sites to set Cookies
    - select Keep Cookies: ask me every time
    - cookie managers (e.g. Cookie Monster for Firefox)
The Web
Publish/Subscribe Web Channels

- **RSS**: really simple syndication
  - began at Netscape in late 1990s
    - hyped as “push technology” (but isn’t)
  - RSS advisory board maintains current specification
    [http://www.rssboard.org/rss-specification](http://www.rssboard.org/rss-specification)

- **Publish / subscribe (pub/sub)**
  - content providers publish feed of changing information
  - users subscribe; provided with .rss URI of RSS channel

- **Client periodically polls RSS feed (e.g. hourly)**
  - XML file returned; client can either:
    - download content or index items for later retrieval
The Web
RSS Clients

• RSS: really simple syndication
• Client periodically polls RSS feed (e.g. hourly)
  – XML file returned; client can either:
    – download content or index items for later retrieval
• Example clients
  – Google Reader launched 2005 was widely used
    • but Google terminated in 2013
  – The Old Reader
    • emerged as alternative to Google reader in 2013
    • backend for many reader apps
The Web
Locating Information

- As the web grew in the 1990s problem:
  - how to find information?

- Two solutions emerged around 1994:
  - structured index (e.g. Yahoo)
  - search engines (e.g. Lycos)

- Most evolved into Web portals attempting everything
  - home page, news, search, index, email
    - dir.yahoo.com is now buried way down on the Yahoo page
  - struggling with business model: ads, subscription
The Web

Web Indices

• Structured index of Web content
  – structured tree of Web pages based on type of information
• Yahoo! founded in 1995 as Web index
  – Yahoo! staff surfed and organised content
  – individuals submitted links for approval and insertion

Advantages and disadvantages?
The Web

Web Indices

- Structured index of Web content
  - structured tree of Web pages based on type of information
- Yahoo! founded in 1995 as Web index
  - Yahoo! staff surfed and organised content
  - individuals submitted links for approval and insertion
- Advantages and disadvantages?
  - useful paradigm when Web was small (1995 – ~1998)
  - manual indexing doesn’t scale
  - no longer useful way to find information
  - dir.yahoo.com charged to assure link consideration
  - directory now gone
Selection of influential search engines

- 1994: Lycos (Lycosidae – wolf spiders)
  - Mauldin at CMU
  - first widely used search engine
- 1994: Metacrawler
  - Selberg, Kalili, Etzioni at U.Wash.
  - parallel search to several engines with combined results
- 1995: Altavista (view from above)
  - Monier et al. at DEC WRL
  - reputation for much better searches
  - became cluttered Web portal
The Web
Search Engines

• Selection of influential search engines
  – 1998 Google
    • simple alternative to Altavista with very effective searching
      – Altavista simplified in response
    • non-intrusive advertising model: sponsored links
    • continues to innovate searching
      – image results
      – shopping
    • branched out but has kept the search page simple
      – Gmail
      – Google maps
  • privacy concerns
    – Google uses to data mine user behaviour
The Web
Search Engines

• Selection of influential modern search engines
  – Yahoo! Search
    • evolution of Yahoo! from index to search engine
    • now mostly a front end using Bing for search queries
    • now default search engine for Firefox
  – 2009 Microsoft Bing
    • attempt to compete with Google
  – 2006 DuckDuckGo
    • founded by Gabriel Weinberg
    • initially self-funded; now by non-tailored ads
    • intended for those who do not want to be tracked by Google

*this is my default search engine*
Information Access
Media Streaming Overview

- Media streaming
  - delivery and playback of continuous media streams
  - e.g. video clip or movie

- Motivation
  - avoid delay of transfer-then-play (especially for long files)
  - prevent users from keeping copy (but they can be ripped)

- Components:
  - client: media player such as VLC, iTunes, or WMP
    - may also play a stored file
  - server: media server
  - client/server protocol: RTSP [RFC 2326]
The Web
Performance Improvement

• Web was dominant source of Internet traffic
  – mid 1990s to early 2000s
  – still a significant fraction of traffic

• Problem
  – delay of request response loop
  – aggregate bandwidth of serving requests

Solution?
Web Performance Improvement

Mirroring

• Mirroring: locate *copy* of data close to users

• Location techniques:
  1. manual selection
  2. proxy redirection
  3. anycast
  4. server redirection
  5. active mirroring (network node redirects request)
The Web: Performance

Caching

- Local caching
The Web: Performance

Caching

- Local caching
  - client/server
  - request
The Web: Performance

Caching

- Local caching
  - client/server
  - request
The Web: Performance

Caching

- Local caching
  - client/server
  - request/response
The Web: Performance

Caching

- Local caching
  - client/server
  - request/response

- Keep local copy of response in *cache*
The Web: Performance
Caching

- Local caching
  - client/server
  - request/response

- Keep local copy of response in cache
  why?
The Web: Performance

Caching

- Local caching
  - client/server
  - request/response

- Keep local copy of response in *cache*
  - assume there may be later requests for same object

*benefit?*
The Web: Performance
Caching

- Local caching
  - client/server
  - request/response

- Keep local copy of response in *cache*
  - assume there may be later requests for same object
  - subsequent requests do not need to traverse the network
The Web: Performance

Caching

- Local caching
  - client/server
  - request/response

- Keep local copy of response in *cache*
  - assume there may be later requests for same object
  - subsequent requests do not need to traverse the network

*Implications?*
Local caching
- client/server
- request/response

- Keep local copy of response in cache
  - assume there may be later requests for same object
  - subsequent requests do not need to traverse the network

- Significant benefits
  - reduced response time: eliminates client/server loop
  - reduced network traffic for repeated requests
The Web: Performance

Caching

- Local caching
  - client/server
  - request/response
- Keep local copy of response in *cache*
  - assume there may be later requests for same object
  - subsequent requests do not need to traverse the network

*Problem?*
The Web: Performance

Caching

• Local caching
  – client/server
  – request/response

• Keep local copy of response in *cache*
  – assume there may be later requests for same object
  – subsequent requests do not need to traverse the network

• Problem: cached items *age* and lose *freshness*
  – significant issue for changing content such as news

*solution?*
The Web: Performance
Caching

• Local caching
  – client/server
  – request/response

• Keep local copy of response in cache
  – assume there may be later requests for same object
  – subsequent requests do not need to traverse the network

• Problem: cached items age and lose freshness
  – local date modified determines if new copy might be needed
    – only request copy from server if local copy stale
The Web: Performance
Caching

- Local caching
  - client/server
  - request/response

- Keep local copy of response in *cache*
  - assume there may be later requests for same object
  - subsequent requests do not need to traverse the network

- Problem: cached items *age* and lose *freshness*
  - local date modified determines if new copy *might* be needed
  - send *request* with If-modified-since: ⟨date-time⟩
The Web: Performance

Caching

• Local caching
  – client/server
  – request/response

• Keep local copy of response in *cache*
  – assume there may be later requests for same object
  – subsequent requests do not need to traverse the network

• Problem: cached items *age* and lose *freshness*
  – local date modified determines if new copy *might* be needed
  – send *request* with If-modified-since: ⟨date-time⟩
Local caching

- client/server
- request/response

Keep local copy of response in cache

- assume there may be later requests for same object
- subsequent requests do not need to traverse the network

Problem: cached items age and lose freshness

- local date modified determines if new copy might be needed
- send request with If-modified-since: <date-time>
  - if server copy identical return 304 Not Modified message

Implication?
The Web: Performance

Caching

- Local caching
  - client/server
  - request/response

- Keep local copy of response in cache
  - assume there may be later requests for same object
  - subsequent requests do not need to traverse the network

- Problem: cached items age and lose freshness
  - local date modified determines if new copy might be needed
  - send request with If-modified-since: ⟨date-time⟩
    - if server copy identical return 304 Not Modified message
    - response loop only if client decides; network load still reduced
The Web: Performance

Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - on cache miss go up the tree
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - on cache miss go up the tree
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - on cache miss go up the tree
The Web: Performance

Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - on cache miss go up the tree
The Web: Performance

Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - on cache miss go up the tree
  - caches populated with response
  
  *why?*
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - on cache miss go up the tree
  - caches populated with response
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - on cache miss go up the tree
  - caches populated with response
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - on cache miss go up the tree
  - caches populated with response
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - on cache miss go up the tree
  - caches populated with response
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
The Web: Performance

Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
The Web: Performance

Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
The Web: Performance

Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
The Web: Performance

Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
  - repeat for original client with other server

*what happens?*
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
  - repeat for original client with other server
  what happens?
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
  - repeat for original client with other server

*what happens?*
The Web: Performance

Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
  - repeat for original client with other server
  - query served by first cache with content
The Web: Performance
Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
  - repeat for original client with other server
  - query served by first cache with content
- no need to go to server
The Web: Performance

Network Caching

- Network caching hierarchy
  - cache hierarchy organised as a tree
  - root cache accesses servers
  - repeat for a different client/server
  - repeat for original client with other server
  - query served by first cache with content
    - no need to go to server
Web Performance Improvement

CDNs

- Content distribution network (CDN)
  - overlay to push contents to set of proxy caches
  - examples
    - Akamai (commercial)
    - Coral (NYU research)
- Content Internetworking (CDI) [RFC 2003]
  - CDN peering in the Internet
Web Performance Improvement

CDN Example: Akamai

- Akamai
  - proprietary CDN service provider
    - formed in 1995 by Leighton and Lewin (MIT)
    - very few details publicly available

- Akamai FreeFlow
  - pages on Akamai customer server are “Akamaiised”
  - embedded URLs replaced with ARLs
    - Akamai Resource Locator, e.g.
      - http://a56.g.akamai.net/7/56/7207/1579300b2b6778/www.united.com/ual
        /asset/swn_2007_united_brand.gif
    - try it e.g. www.united.com → view source → search for “akamai”
Information Access

AL.3.4  Peer-to-Peer File Sharing

AL.1  Services, interfaces, and functions
AL.2  Application characteristics and types
AL.3  Information access applications
   AL.3.1  File transfer
   AL.3.2  Netnews and Internet fora
   AL.3.3  The Web
   AL.3.4  Peer-to-peer file sharing
AL.4  Telepresence applications
AL.5  Distributed computing and networked storage
AL.6  Application adaptation and network interaction
AL.7  Social networking
Peer-to-Peer File Sharing
Overview

• P2P file sharing between users
  – dominant traffic in the Internet early 2000s to early 2010s

• Composed application consisting of two major phases
  – location of content in distributed storage
  – client/server information access

• Motivation
  – allow any user to be both client and server of information
    • sometimes called “servent” from server+client
  – in spite of asymmetric ISP infrastructure and TOS
    • ISPs engineered for the Web in the 1990’s and can’t cope
Peer-to-Peer File Sharing

Meta-Issues

• Beware of the hype
  – P2P is *not* a paradigm shift ...
    ... it’s the way the Internet was designed!
  – short sighted service providers were caught off guard ...
    ... when they designed asymmetric infrastructure
    • based on the dominant Web traffic of the 1990’s

• There are many related legal and political issues
  – that are mostly out of scope in this course
  – like *everything* else, ∃ both legitimate and illegitimate uses
    • there is massive potential for legitimate use
    • e.g. software distribution, free content
Peer-to-Peer File Sharing
Evolution of Location Mechanisms

- 0th generation: traditional client/server – FTP, Web
  - individual Web servers of content
- 1st generation: centralised directory
  - e.g. Napster
- 2nd generation: unstructured decentralised
  - e.g. Gnutella 0.4
- 2.5 generation: hybrid
  - e.g. Gnutella 0.6, KaZaA/FastTrack, eDonkey2000, Skype
- 3rd generation: structured using DHTs
  - e.g. Kademila, Chord, Pastry
Peer-to-Peer File Sharing
FTP and the Web

• We’ve been file sharing for the last 3 decades!
  – FTP with out-of-band notification of content (e.g. email)
  – Web servers serving content including multimedia

• But most average users can’t be a Web server
  – ISPs resist: DHCP and TOS restrictions
  – business service permits but is much more expensive
  – Web hosting has higher entry barrier than client software

• So peer-to-peer is a response to non-technical issues:
  – response to greedy over-litigious luddite RIAA and MPAA*
  – subversion of ISP restrictions on server hosting

*whilst this statement is personal opinion, it seems obvious to the instructor
Peer-to-Peer File Sharing

RIAA Background

- **RIAA**: Recording Industry Association of America
  - many members
    - dominated by big four (EMI, Sony BMG, Universal, Warner)
    - RIAA claims membership of labels that aren’t!
- Originally (1952) a vital enabler of technology
  - influenced standards for recording media and formats
- Now a significant impediment to technology
  - pushes legislation to restrict technology: $Ms/yr lobby
  - pushes legislation to limit rights far beyond past fair use
    - creating backups for personal use
    - loading on iPod or MP3 player
Peer-to-Peer File Sharing
RIAA Litigation

• Over 20 000 lawsuits filed against individuals for
  – music sharing using P2P networks

• Example lawsuits
  – sued and shut down Napster and other P2P networks
  – attempts to get customer records from ISPs
  – XM radio for allowing subscribers to play recorded tunes
  – individuals (generally for $750 per tune)
    • who don’t own computers
    • with Internet access but who never shared tunes
    • estates (families) of deceased individuals
    • elderly (e.g. 88 yrs. old) and minor children (e.g. 12 yrs. old)
    • recently, college students for $150 000 per file
I am contacting you on behalf of the Recording Industry Association of America, Inc. (RIAA) and its member record companies...

We believe a user on your network is offering an infringing sound recording for download through a peer to peer application. We have attached below the details of the infringing activity.

We have a good faith belief that this activity is not authorized by copyright owners, their agent, or the law. We are asking for your immediate assistance in stopping this unauthorized activity. Specifically, we request that you remove or disable access to the infringing sound recording...

You should understand that this letter constitutes notice to you that this network user may be liable for the infringing activity occurring on your network. In addition, under the Digital Millennium Copyright Act, if you ignore this notice, your institution may also be liable for any resulting infringement. This letter does not constitute a waiver of any right to recover damages incurred by virtue of any such unauthorized activities, and such rights as well as claims for other relief are expressly retained. Moreover, this letter does not constitute a waiver of our members' right to sue the user at issue for copyright infringement...

First Found: date
Network: BTPeers
IP Address: n.n.n.n
IP Port: nnnn
Protocol: BitTorrent
List of infringing content: name
Peer-to-Peer File Sharing

RIAA Recent Tactics

• RIAA announced no longer will sue individuals
  – probably due to bad publicity and countersuits

• RIAA now targetting ISPs
  – attempting to strike deals to terminate users’ service
  – threatening other ISPs

• RIAA targetting video sharing sites
  – aggressively issuing DMCA takedown messages...
    ...whether or not justified

• Six strikes (2013)
Peer-to-Peer File Sharing
RIAA Dirty Tricks

- Congressional staff attorney sneaks language into bill
  - classifying music recordings as “work made for hire”
  - strips copyright from artists
  - later hired into high position in RIAA
- Lead to formation of Recording Artists Coalition
  - successfully lobbied for repeal of measure
Peer-to-Peer File Sharing

MPAA Background

• MPAA: Motion Picture Association of America
  – formed in 1922 as a trade association
  – membership (big six)
    • Buena Vista (Disney), Sony (Columbia), Paramount, Fox (NewsCorp), NBC Universal, Warner Bros. (Time Warner)

• Film rating system: devised and rates films
  – protection of copyrighted content

• Now a significant impediment to technology
  – pushes legislation to restrict technology
  – pushes legislation to limit rights far beyond past fair use
    • e.g. creating backups for personal use
Peer-to-Peer File Sharing

MPAA Litigation

- Movie sharing behind music sharing in volume
  - ~3 orders of magnitude larger files (GB vs. MB)
- Example legal action
  - stopped sales of DVDXCopy in 2004 for DVD backup
    - court ruled that copies are legal, but decryption software isn’t
  - shut down SuperNova, LokiTorrent, EliteTorrents
  - targeting Pirate Bay
  - threats to > 400 students at 18 Universities on Internet2
    - shuts down I2hub
    - MPAA and RIAA joined Internet2 to
      “collaborate with consortium to study advanced content distribution technologies”
Peer-to-Peer File Sharing

MPAA Dirty Tricks

• MPAA grossly exaggerates monetary impacts
  – claims that Internet piracy costs industry many billions of dollars
  – been forced to admit that figures grossly inflated

• MPAA piracy: illegally used Forest Blog [www.hostforest.co.uk](http://www.hostforest.co.uk)
  – didn’t link back nor pay £25 license
    • removed blog after bad press, claiming it was a “test”
Peer-to-Peer File Sharing
Lack of Architectural Discipline in Deployments

- Most widely deployed P2P file sharing systems
  - quick hacks by amateur users or inexperienced startups
    - deployed without testing or analysis
    - many have serious security and privacy issues
  - proprietary closed systems or poorly documented open systems

- Lack of detailed information on deployed systems
  - finally an IRTF peer-to-peer research group formed 2003
    - attempts to form IETF working group based on JXTA failed
  - little or no documentation on most architecture & protocols
    - at best, incomplete protocol specs for open systems
    - reverse engineering attempts of some closed systems
P2P File Sharing

Centralised

- Content indexed in central database
P2P File Sharing
Centralised

- Content indexed in central database
  1. users register content to be served
P2P File Sharing
Centralised

- Content indexed in central database
  1. users register content to be served
  2. clients search directory to locate server
P2P File Sharing
Centralised

- Content indexed in central database
  1. users register content to be served
  2. clients search directory to locate server
  3. directory returns location(s) of content
P2P File Sharing
Centralised

- Content indexed in central database
  1. users register content to be served
  2. clients search directory to locate server
  3. directory returns location(s) of content
  4. clients requests content from a server
P2P File Sharing
Centralised

- Content indexed in central database
  1. users register content to be served
  2. clients search directory to locate server
  3. directory returns location(s) of content
  4. clients requests content from a server
  5. client gets content directly from server
P2P File Sharing
Centralised

- Content indexed in central database
  1. users register content to be served
  2. clients search directory to locate server
  3. directory returns location(s) of content
  4. clients requests content from a server
  5. client gets content directly from server

- Example: Napster

*Problems?*
P2P File Sharing
Centralised

- Content indexed in central database
  1. users register content to be served
  2. clients search directory to locate server
  3. directory returns location(s) of content
  4. clients requests content from a server
  5. client gets content directly from server

- Example: Napster

- Problems
  - central bottleneck and point-of-failure
  - easy target for legal attack

content directory
P2P File Sharing
Growth Complexity

- Scaling complexity measured as number of nodes $n$
  - per node state: information in each of $n$ nodes that index
  - communication overhead: messages exchanged by $n$ nodes
- each traversing $h$ hops
Centralised P2P File Sharing
Growth Complexity

- Scaling complexity measured as number of nodes $n$
  - per node state: information in each of $n$ nodes that index
  - communication overhead: messages exchanged by $n$ nodes
- each traversing $h$ hops

<table>
<thead>
<tr>
<th>Node state</th>
<th>Comm. overhead</th>
<th>Fuzzy queries</th>
<th>Resilience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
<td>yes</td>
</tr>
</tbody>
</table>

$O(1)$ $O(n)$ $O(n \log n)$ $O(n^2)$
Centralised P2P File Sharing

Napster History

• Brief history
  – May 1999: began in by NEU freshman Shawn Fanning
    • used primarily to share ripped MP3 music files
      – many but not all illegally
  – Dec. 1999: Napster targetted by RIAA lawsuit
  – Mar. 2000: 25% of U.Wisconsin traffic is Napster
  – Feb. 2001: 2.8 billion files exchanged in single month
  – Mar. 2001: Gnutella released as open source by Nullsoft
  – Mar. 2001: Napster ordered to cease indexing © content
  – Jul. 2001: Napster shuts down
  – Aug. 2001: 3.1 billion files exchanged via Gnutella in 1 mo.
Centralised P2P File Sharing
Napster Overview

- Operation: centralised directory
- TCP is used to advertise content by *server* ...
  - file IP address, port, and characteristics (QoS)
- ... and query content by *server*
  - search by partial artist and title and desired QoS
- HTTP is used to transfer files
  - directly from server to client using advertised address:port
Centralised P2P File Sharing
Napster Message Format

- **Message format**: fixed header
  - \( \langle \text{length}_{16B} \rangle \) payload length
  - \( \langle \text{function}_{1B} \rangle \) message opcode
- **Variable payload**:
  - \( \langle \text{payload}_n \rangle \) variable length ASCII text fields
    - fields separated by space
Centralised P2P File Sharing
Napster Protocol Messages

- **LOGIN** to join overlay
  
  | len | 02 | nick | passwd | port | client-info | link-type |

- **LOGIN ACK**
  
  | 00 | 03 |

- **CLIENT NOTIFICATION OF SHARED FILE** to offer content
  
  | len | C4 | "name" | MD5 | size | bitrate | freq. | time |

- **SEARCH** query for content on partial string match
  
  | len | C8 | "artist" | max-reslt | "song" | relop | link-type | relop | bitrate | relop | freq |

- **RESPONSE** to return peer with matching content
  
  | len | C9 | "name" | MD5 | size | bitrate | freq | length | nick | IP | link-type |
Centralised P2P File Sharing

Napster Protocol Fields

- **Servent fields**
  - **nick**: nickname of servent
  - **passwd**: password of servent
  - **IP**: IP address
  - **port**: TCP port on which server listens
  - **client-info**: version of Napster client software

- **Content fields**
  - **name**: file name
  - **MD5**: 128-bit MD5 fingerprint of file
  - **size**: file size in bytes
  - **time, length**: file length in sec.
Centralised P2P File Sharing
Napster Protocol Fields

- **QoS (quality of service) fields**
  - link-type: bandwidth of content server access link
  - bitrate: encoding rate
  - freq.: sampling frequency

- **Search fields**
  - “artist”: partial string of artist
  - “song”: partial string of song title
  - max-reslt: max. num. of results desired from search
  - relop: relational operator for match: \( \{ \leq | = | \geq \} \)
Centralised P2P File Sharing
Napster Signalling Example

Based on [Steinmetz-Wehrle-2005]
Centralised P2P File Sharing
Napster Signalling Example

LOGIN
LOGIN ACK
NOTIFY

content directory
Centralised P2P File Sharing
Napster Signalling Example

content directory

LOGIN
LOGIN ACK
NOTIFY
SEARCH

Napster

1

2

content directory
Centralised P2P File Sharing
Napster Signalling Example

1. LOGIN
2. NOTIFICATION
3. RESPONSE

content directory
Centralised P2P File Sharing

Napster Signalling Example

HTTP
GET

LOGIN
LOGIN ACK
NOTIFY
SEARCH
RESPONSE

content directory

21 September 2016
KU EECS 780 – Comm Nets – Networked Applications
Centralised P2P File Sharing
Napster Signalling Example

HTTP
GET
content
directory

file

Napster

1. LOGIN
2. NOTIFY
3. SEARCH
4. RESPONSE
5. RESPONSE

content directory
P2P File Sharing
Centralised

- Content indexed in central database
- Example
  - Napster
- Problems
  - central bottleneck and point-of-failure
  - easy target for legal attack

*Alternative?*
P2P File Sharing
Decentralised

- Response to centralised problems
  - central bottleneck and point-of-failure
  - easy target for legal attack
- No central content index
P2P File Sharing
Decentralised

- No central content index
  1. users join and expose content served
P2P File Sharing
Decentralised

- No central content index
  1. users join and expose content served
  2. clients *flood* queries among peers
P2P File Sharing
Decentralised

- No central content index
  1. users join and expose content served
  2. clients *flood* queries among peers
P2P File Sharing
Decentralised

- No central content index
  1. users join and expose content served
  2. clients *flood* queries among peers
  3. clients answer if they have content
P2P File Sharing
Decentralised

- No central content index
  1. users join and expose content served
  2. clients *flood* queries among peers
  3. clients answer if they have content
  4. clients requests content from a server
P2P File Sharing
Decentralised

- No central content index
  1. users join and expose content served
  2. clients *flood* queries among peers
  3. clients answer if they have content
  4. clients requests content from a server
  5. client gets content directly from server
P2P File Sharing
Decentralised

- No central content index
  1. users join and expose content served
  2. clients *flood* queries among peers
  3. clients answer if they have content
  4. clients requests content from a server
  5. client gets content directly from server

- Example: Gnutella

*Problems?*
P2P File Sharing

Decentralised

- No central content index
  1. users join and expose content served
  2. clients *flood* queries among peers
  3. clients answer if they have content
  4. clients requests content from a server
  5. client gets content directly from server

- Example: Gnutella

- Problems
  - flooding is very inefficient *Lecture NR*

*Alternative?*
Decentralised P2P File Sharing

Growth Complexity

- Scaling complexity measured as number of nodes $n$
  - per node state: information in each of $n$ nodes that index
  - communication overhead: messages exchanged by $n$ nodes
  - each traversing $h$ hops
Decentralised P2P File Sharing

Gnutella History

• Brief history
  – Dec. 1999: Napster targetted by RIAA lawsuit
  – Mar. 2001: Gnutella 0.4 by Frankel & Pepper by Nullsoft
    • parent AOL orders Gnutella removed from Nullsoft servers
      – but too late
  – Jul. 2001: Napster ordered to cease indexing © content
  – Aug. 2001: 3.1 billion files exchanged via Gnutella in 1 mo.
    • remarkably rapid substitution of technology
  – Gnutella 0.6 became hybrid system in late 2001
Decentralised P2P File Sharing

Gnutella Overview

• An *overlay* is used to construct Gnutella graph
  – set of peer-to-peer TCP connections
  – has no direct relationship to underlying network topology

• Operation  [Gnutella 0.4 Protocol Specification]
  – get Gnutella peer list from bootstrap server
  – CONNECT to at least one to three of them
  – broadcast PING messages to find additional peers via PONG
  – search for content with QUERY flood; TTL typ. 7 hops
  – select from QUERYHIT replies
  – HTTP to get file from selected peer
  – PINGs periodically broadcast for keep-alive
Decentralised P2P File Sharing
Gnutella Message Format

- **CONNECT** messages: ASCII text
- **Message format**: fixed header
  - \(\langle \text{ID}_{16B} \rangle\) 128b node id
  - \(\langle \text{function}_{1B} \rangle\) message opcode
  - \(\langle \text{TTL}_{1B} \rangle\) number of hops a message may pass
  - \(\langle \text{Hops}_{1B} \rangle\) number of hops message has traversed
  - \(\langle \text{Len}_{4B} \rangle\) payload length

- **Variable payload**:
  - \(\langle \text{payload}_n \rangle\) variable length ASCII text fields
    - field length predefined or null-terminated
    - subfields separated by space
Decentralised P2P File Sharing
Gnutella Protocol Messages

- **GNUTELLA CONNECT** to join overlay
  
<table>
<thead>
<tr>
<th>ID</th>
<th>TTL</th>
<th>Hop</th>
<th>Len</th>
<th>port</th>
<th>IP addr</th>
<th>#files shared</th>
<th>kB shared</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

- **GNUTELLA OK** acknowledgement to CONNECT
  
<table>
<thead>
<tr>
<th>ID</th>
<th>TTL</th>
<th>Hop</th>
<th>Len</th>
<th>port</th>
<th>IP addr</th>
<th>#files shared</th>
<th>kB shared</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
</tbody>
</table>

- **PING** to explore neighbourhood and keep-alive
  
<table>
<thead>
<tr>
<th>ID</th>
<th>TTL</th>
<th>Hop</th>
<th>Len</th>
<th>port</th>
<th>IP addr</th>
<th>#files shared</th>
<th>kB shared</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

- **PONG** to reply to ping with address
  
<table>
<thead>
<tr>
<th>ID</th>
<th>TTL</th>
<th>Hop</th>
<th>Len</th>
<th>port</th>
<th>IP addr</th>
<th>#files shared</th>
<th>kB shared</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
</tr>
</tbody>
</table>

- **QUERY** for content on partial string match
  
<table>
<thead>
<tr>
<th>ID</th>
<th>TTL</th>
<th>Hop</th>
<th>Len</th>
<th>port</th>
<th>IP addr</th>
<th>#files shared</th>
<th>kB shared</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
</tr>
</tbody>
</table>

- **QUERYHIT** to return peer with matching content
  
<table>
<thead>
<tr>
<th>ID</th>
<th>TTL</th>
<th>Hop</th>
<th>Len</th>
<th># hits</th>
<th>port</th>
<th>IP addr</th>
<th>speed</th>
<th>result set</th>
<th>ServentID</th>
</tr>
</thead>
<tbody>
<tr>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
<td>81</td>
</tr>
</tbody>
</table>
Decentralised P2P File Sharing
Gnutella Protocol Fields

- **Servent fields**
  - ServentID: unique ID of servent within peer group
  - IP addr: IP address of server
  - port: TCP port on which server listens
  - protocol-ver: version of Gnutella protocol

- **Flooding fields**
  - TTL: max # hops
  - Hops: # hops traversed
Decentralised P2P File Sharing
Gnutella Protocol Fields

- **Search fields**
  - search criteria: null terminated search string
  - # hits: number of hits in result set
  - result set: query responses, each consisting of:
    - file index: unique identifier of file
    - file size: file size [B]
    - file name: double null (0000) terminated file name

- **Content fields**
  - #files shared: # of files server has to share
  - kB shared: total kB server has to share
  - min speed: minimum data rate data for transfer
  - speed: data rate server can transfer
Decentralised P2P File Sharing
Gnutella Signalling Example

Peer7  Peer3  Peer1  Peer5  Peer2  Peer4  Peer6  Peer8

PING  PONG  PONG  PING  PONG  PING  PING  PING

Gnu-Con  OK  Gnu-Con  OK  Gnu-Con  OK

[SW2005]
Decentralised P2P File Sharing
Gnutella Overlay Performance

Performance issues?
Decentralised P2P File Sharing
Gnutella Overlay Performance

- Flooding overhead: PING/PONG traffic
- Overlay topology not related to underlay
  - PING/PONG may traverse *long thin* links multiple times
    - long delay low bandwidth

*Alternatives?*
P2P File Sharing
Performance Enhancements

- Search
  \textit{centralised vs. flooding alternative?}
- File transfer
  \textit{later}
P2P File Sharing

Hybrid

- Response to problems with lookup
  - centralised bottleneck and point-of-failure
  - flooding performance of pure peer-to-peer

Solution?
P2P File Sharing

Hybrid

- Response to problems with lookup
- Solution: middle ground
  - build hierarchy
    - **ultrapeers, superpeers, or supernodes** with high degree of connectivity
    - normal servant peers connect to ultrapeer
Hybrid P2P File Sharing

Growth Complexity

- Complexity of **hybrid** somewhere between
  - centralised and decentralised
  - depending on topology

<table>
<thead>
<tr>
<th></th>
<th>Node state</th>
<th>Comm. overhd.</th>
<th>Fuzzy queries</th>
<th>Resilience</th>
</tr>
</thead>
<tbody>
<tr>
<td>Central</td>
<td>$O(n)$</td>
<td>$O(1)$</td>
<td>yes</td>
<td>poor</td>
</tr>
<tr>
<td>Flooding</td>
<td>$O(1)$</td>
<td>$O(hn^2)$</td>
<td>yes</td>
<td>excellent</td>
</tr>
<tr>
<td>?</td>
<td>$O(n)$</td>
<td>$O(n^2)$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Hybrid P2P File Sharing

Gnutella 0.6 Overview

- Addition of ultrapeer functionality to Gnutella 0.4
  - [rfc-0_6-draft.html]
- New nodes join ultrapeer
  - ultrapeers exchange routing updates with ROUTE TABLE UPDATE (opcode 30)
- Leaf nodes may be promoted to ultrapeer, based on
  - topology (splitting of ultrapeer)
  - stable connection to network
- Significantly lower message overhead
  - but flooding still occurs among ultrapeer clusters
Hybrid P2P File Sharing
FastTrack/Kazaa Protocol

  - FastTrack proprietary protocols
    - client–supernode and supernode–supernode
  - KaZaA client
- UUHash (combination of MD5 and CRC-32)
  - permits suspend/resume and downloading from multiple peers
  - susceptible to poisoning by the RIAA
- Client–supernode protocol
  - encrypted but reverse engineered
Hybrid P2P File Sharing
FastTrack/Kazaa Clients

• Kazaa (formerly KaZaA) client
  – adware/spyware financial support model

• Other clients
  – Kazaa lite: unauthorised version without malware
  – Grokster  (shut down by RIAA lawsuit)
  – Morpheus  (current version checks for content copyright)
Hybrid P2P File Sharing

eDonkey Protocol

- eDonkey
  - originally central servers
  - evolved to hybrid P2P network
- MD4 hash of chunked files
  - permits download from multiple peers
- MFTP: multisource FTP
  - US Patent 6,339,785
- pDonkey open source project at sourceforge
  - protocol “specification” available
Hybrid P2P File Sharing

eDonkey Clients

• eDonkey2000 client
  – developed by MetaMachine
  – uses MFTP over eDonkey network
  – last version supported BitTorrent
  – RIAA shut down MetaMachine in Sep. 2006
    • $30M settlement

• Other clients now in wide use
  – aMule (also supports Kademila)
  – aMule (also supports Kademila)
  – MLDonkey
Unstructured P2P File Sharing

Growth Complexity

- Unstructured file sharing
  - location of content has no inherent structure
Peer-to-Peer File Sharing
Improving Growth Complexity

- Question: can we do better?
  - $O(n)$ node state of centralised and hybrid
  - poor resilience of centralised
  - $O(hn^2)$ comm. overhead of distributed flooding

What is the goal?
Peer-to-Peer File Sharing

Improving Growth Complexity

• Question: can we do better?
  – $O(n)$ node state of centralised and hybrid
  – poor resilience of centralised
  – $O(hn^2)$ comm. overhead of distributed flooding

• Goal: locate information by content
  – avoid flooding for search query

• CAMs (content addressable memories)
  – return value in 1 cycle by simultaneous matching all entries

• Structured P2P using DHTs

Lecture NL
Structured P2P File Sharing

Growth Complexity

- Structured P2P using DHTs
P2P File Sharing
Evolution of Protocols on Abilene

[Steinmetz-Wehrle-2005]
Darknets
Motivation and Introduction

• Darknet: private encrypted overlay
  – invisible to all but invited members

• Motivation
  – hiding from authorities for illegal activity
  – hiding from repressive governments to support free speech
  – hiding from RIAA and MPAA to share media files

• Examples
  – small closed groups
  – Freenet
    • provides strong anonymity
    • encrypted distributed P2P file sharing network
Peer-to-Peer Applications
File Sharing Hacks

• Problem: most popular P2P applications are hacks
  – stovepipe monoliths: client integrated with protocols
  – designed and deployed by single individual
    • research community not originally involved
  – some bundled with malware
  – recent closed source clients suspect, e.g. bittorrent
    • back door for RIAA and MPAA?

Response?
Peer-to-Peer Applications

JXTA History

- **JXTA** (from *juxtapose*): open source P2P framework
  - begun at Sun Microsystems in 2001
  - submitted to IETF to begin working group
  - IRTF p2prg formed instead; no standards development
  - Sun continued involvement in JXTA development community

- **JXTA project goals** [https://jxta.dev.java.net]
  - interoperability across different P2P systems & communities
  - platform independence (languages, systems, and networks)
  - ubiquity on every device
Peer-to-Peer Applications

JXTA Overview

• JXTA: protocol framework for app. development
  – for application development
    • P2P file sharing only one of many P2P apps
    • JXTA is not a particular file-sharing client

• JXTA open development
  – originally Java (Sun heritage)
  – later also C and C++
  – numerous JXTA subprojects
    • architectural aspects
    • applications
    • protocols
Peer-to-Peer Applications

JXTA Software Architecture

- JXTA layered software and protocol architecture
  - application, peer group
    - services, communication and monitoring, security
Peer-to-Peer Applications

JXTA Overlay Architecture: Peers

- JXTA edge peers (servents)
  - full-edge peer
    - full protocol and service implementation
  - minimal-edge peer
    - only required core services and rely on proxy peers
    - limited capability devices, e.g. sensors and home automation

- JXTA super-peers
  - relay peers: forward to punch through firewalls or NATs
  - rendezvous: advertisement pub/sub and message broadcast
  - proxy: provide services to minimal-edge peers

- Peer groups: self-organised set of groups
  - peers may belong to more than one peer group
Peer-to-Peer Applications

JXTA Overlay Architecture: Networking

- JXTA pipes: inter-peer (overlay) association
  - characteristics
    - asynchronous
    - unidirectional
    - unreliable (by default)
  - types
    - point-to-point
    - propagate: connects one output to multiple input pipes
    - secure unicast: secure and reliable channel

- Networking
  - IDs: uniform peer addressing (no reliance on DNS)
  - DHT indexing and routing
    Lecture NL
Peer-to-Peer Applications

JXTA Protocols

- JXTA protocols specified as XML messages
  - PRP: peer resolver protocol – query/response
  - PDP: peer discovery protocol resource – publish/discover
  - PIP: peer information protocol – status query of other peers
  - PBP: pipe binding protocol – channel establishment
  - ERP: endpoint routing protocol – path as sequence of hops
  - RVP: rendezvous protocol – propagation service subscription
Networked Applications

AL.4  Telepresence Applications

AL.1  Services, interfaces, and functions
AL.2  Application characteristics and types
AL.3  Information access applications
AL.4  Telepresence applications
  AL.4.1  Email
  AL.4.2  Chat
  AL.4.3  Conferencing
AL.5  Distributed computing and networked storage
AL.6  Application adaptation and network interaction
AL.7  Social networking
Internet Applications

Telepresence

• Characteristics
  – peer-to-peer
    • peers interact with one another
    • may also use servers to assist
  – fine granularity: unit of information requested
  – relatively symmetric bandwidth
    • assuming peers have similar communication characteristics
  – end-to-end synchronisation: may be significant

• Key components
  – client software: user interface
  – client–client protocol
Networked Applications

Telepresence Applications

- Range of telepresence applications
  - email
  - chat, texting
  - voice
  - video conferencing

*increasing* interactivity and bandwidth
Networked Applications

AL.4.1 Email

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
AL.4 Telepresence applications
  AL.4.1 Email
  AL.4.2 Chat
  AL.4.3 Conferencing
AL.5 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Electronic Mail Overview

- **Email**: electronic mail
  - one of the first killer apps that drove demand for networks
    - file transfer, remote login, email
- **Users exchange messages**
  - but not in real time
  - analogue to traditional postal system
- **One of the few truly best effort applications**
  - doesn’t matter exactly how quickly messages arrive
  - but users generally expect no more than a few minutes
Electronic Mail
Selected History

• Significant (selected) early events in email history
  – ~1965: email within one system  [Van Vleck at MIT]
  – 1971: networked email using @  [Tomlinson at BBN]
  – 1973: header definition  [RFC 0561]
  – 1975: reply and forward programs  [Vital at USC-ISI]
  – 1977: modern complete headers  [RFC 0733]
  – 1982: SMTP transfer protocol  [RFC 0821]
  – 1984: POP for server retrieval  [RFC 0918]
  – 1986: MX records for relaying  [RFC 0974]
  – 1992: MIME multimedia attachments [RFC 1341]
  – first use of email term unknown  [emailhistory.org/Email-Timeline.html]
Electronic Mail
Non-History

- VA Shiva Ayyadurai did \textit{not} invent email
  - nor any significant aspect of it
- He very publicly claims to have invented "EMAIL"
  - based on a program he wrote in 1979–1982 called EMAIL
    - 15 years after MIT/Van Vleck; 8 years after BBN/Tomlinson
    - 6 years after RFC 561; 2 years after RFC 733 defining headers
  - he has conned a number of media outlets
    - some of whom have corrected or retracted their reports
  - he created the \textit{MIT EMAIL Lab}
    - until they kicked him off campus; renamed \textit{THE EMAIL Lab}
  - has a stunning self-promoting web page full of false claims
Electronic Mail

Components

- Email is a peer-to-peer application
  - that now usually uses supporting server infrastructure

- Components:
  - client: email program
    - compose, insert into message queue, and send
    - receive from mailbox and read
  - client–client protocol: SMTP and MIME
  - supporting infrastructure
    - email servers
    - client/server protocol: POP or IMAP
Electronic Mail
Early Client P2P Interaction

• Email client allows user to compose and read email
• Early clients used CLI
  – example: Unix mail
• Operation was pure peer-to-peer using only SMTP
  – frequently relayed along multiple SMTP–SMTP hops
    • no longer allowed to resist anonymous spam
  – required user machine to be (mostly) connected to Internet
Electronic Mail
Text-Based Clients

- **Text-based CLI (command-line interface)**
  - Unix mail
    - default mail client in Unix systems
  - mh: message handling system
    - alternative mail client for Unix systems

- **Test-based full-screen**
  - elm (1986) for Unix and variants
  - pine (1998) for Unix and Windows

- **Graphical window**
  - xmh X-Window graphical front end for mh
Electronic Mail

Example Unix mail Session

```
jpgs@wopr[~].448: mail jpgs@localhost
Subject: Test email
This is a test of old style CLI /bin/mail. If I make a mistake, I have to start over from
scratch.
Cheers,
James
^D
EOT

jjpgs@wopr[~].449: mail jpgs@localhost
Subject: Another test
CC: jpgs@ittc.ku.edu
Isn't this fun?
^D
EOT

jjpgs@wopr[~].450: mail
"/var/mail/jpgs": 2 messages 2 new
>N 1 jpgs@dhcp193.ittc.ku Wed Feb 3 12:05 17/608 "Test email"
N 2 jpgs@dhcp193.ittc.ku Wed Feb 3 12:05 15/532 "Another test"
```
Electronic Mail
Example Unix mail Session

jg@sowpr[~].450: mail
"/var/mail/jg": 2 messages 2 new
>N 1 jg@dchp193.ittc.ku Wed Feb 3 12:05 17/608 "Test email"
>N 2 jg@dchp193.ittc.ku Wed Feb 3 12:05 15/532 "Another test"
>?
Message 1:
From jg@dchp193.ittc.ku.edu Wed Feb 3 12:05:04 2010
X-Original-To: jg@localhost
Delivered-To: jg@localhost.ittc.ku.edu
To: jg@localhost.ittc.ku.edu
Subject: Test email
Date: Wed, 3 Feb 2010 12:05:03 -0600 (CST)
From: jg@dchp193.ittc.ku.edu (James P.G. Sterbenz)

This is a test of old style CLI /bin/mail. If I make a mistake, I have to start over from scratch.
Cheers,
James
Electronic Mail

Example Unix mail Session

? 2
Message 2:
From jpgs@dhcp193.ittc.ku.edu Wed Feb 3 12:05:49 2010
X-Original-To: jpgs@localhost
Delivered-To: jpgs@localhost.ittc.ku.edu
to: jpgs@localhost.ittc.ku.edu
Subject: Another test
Date: Wed, 3 Feb 2010 12:05:49 -0600 (CST)
From: jpgs@dhcp193.ittc.ku.edu (James P.G. Sterbenz)

CC: jpgs@ittc.ku.edu
Isn't this fun?

? d 2
?q
Saved 1 message in mbox
jpgs@wopr[~].451:
Electronic Mail

GUI Clients: Eudora and Apple Mail

- **Eudora**
  - developed 1988 at UIUC [Dorner]
  - basis for standard mail GUIs; widely used on Macs
  - acquired by Qualcomm in 1991
  - used on PCs by people concerned by Outlook insecurity
  - development stopped and open-sourced in 2006
  - 2010: OSE 1.0 based on Thunderbird code base

- **Apple Mail**
  - 2002: OS X based on NeXTSTEP mail client
    - significant similarities with Eudora
    - outstanding search capabilities
Electronic Mail
GUI Clients: MS Outlook

• MS Outlook
  – email client bundled with Microsoft Office
    • Outlook Express is less-capable version bundled with Windows
      – replaced by Window Live Mail after XP
    • general purpose email client
    • optimised for MS Exchange servers
    • Entourage version available for Mac OS X

• Security concerns
  – notorious as a vector for worms, chain letters, and spam
  – integration with MS Internet Explorer made this worse
  – Outlook 2003 addressed some of these concerns
  – Outlook 2007 eliminated use of IE renderer
Electronic Mail
GUI Clients: Thunderbird

• Mozilla Thunderbird
  – free open-source client based on success if Firefox
  – multiplatform: Windows, Mac OS X, Linux, and others
  – released Dec. 2004
    • 1 M downloads in the first 10 days
  – 2006: Qualcomm moves Eudora to Thunderbird base

• Features
  – Firefox-like GUI in typical email client layout (e.g. Eudora)
  – supports add-ons (e.g. Adblock Plus)
  – standard security features and protocol support (SSL, etc.)
Electronic Mail

SMTP

- **SMTP** (simple mail transfer protocol)  [RFC 2821]
- **User agent** (UA) is the interface to the user
  - composition and reading of email
  - e.g. Thunderbird, Unix mail, Eudora, MS Outlook, Mac Mail
- **SMTP server** is the program for mail transfer
  - note that “server” is historical but confusing terminology
  - transfer from client SMTP server to SMTP server
  - uses TCP port 25 for reliable email transfer
- **UA & server**
  - originally typically on same machine
  - now typically on separate computers (e.g. smtp.ittc.ku.edu)
Electronic Mail
SMTP

• Transfer of email
  – from sending client to receiving server
  – [crlf].[crlf] to terminate message
  – header formatted according to [RFC 2822]

• SMTP phases
  – handshake (hello)
  – transfer of messages in 7-bit ASCII via TCP port 25
  – close (quit)

• Command/response interaction
  – commands: ASCII text
  – response: status code and phrase
Electronic Mail
SMTP Example

- Each computer has:
  - email UA
    - possibly multiple users per computer
  - SMTP “server” for outgoing email
    - server is unfortunate name for sending software program
Electronic Mail
SMTP Example

1. Alice composes email to \textit{bob@ku.edu}
Electronic Mail
SMTP Example

1. Alice composes email to bob@ku.edu
2. Alice’s UA sends to her server which queues message
Electronic Mail
SMTP Example

1. Alice composes email to bob@ku.edu
2. Alice’s UA sends to her server which queues message
3. Client side of Alice’s server opens TCP 25 port to ku.edu
Electronic Mail

SMTP Example

1. Alice composes email to \textit{bob@ku.edu}
2. Alice’s UA sends to her server which queues message
3. Client side of Alice’s server opens TCP 25 port to \textit{ku.edu}
4. SMTP client sends message
Electronic Mail
SMTP Example

1. Alice composes email to bob@ku.edu
2. Alice’s UA sends to her server which queues message
3. Client side of Alice’s server opens TCP 25 port to ku.edu
4. SMTP client sends message
5. SMTP server ku.edu receives, inserts in Bob’s mailbox
Electronic Mail
SMTP Example

1. Alice composes email to bob@ku.edu
2. Alice’s UA sends to her server which queues message
3. Client side of Alice’s server opens TCP 25 port to ku.edu
4. SMTP client sends message
5. SMTP server ku.edu receives, inserts in Bob’s mailbox
6. Bob invokes UA to read mail
Electronic Mail: SMTP
Manually Composing Email

- Telnet to SMTP server using port 25
  
  $telnet mail.eecs.ku.edu 25
  
  220 stephens.ku.edu ESMTP Postfix
Electronic Mail: SMTP
Manually Composing Email

• Enter email commands and body

HELO ku.edu
250 mailstore.eecs.ku.edu Hello 97.96.cm.sunflower.com
[24.124.96.97], pleased to meet you
MAIL FROM: <jpgs@eecs.ku.edu>
250 2.1.0 <jpgs@eecs.ku.edu>... Sender ok
RCPT TO: <jpgs@ittc.ku.edu>
250 <jpgs@ittc.ku.edu>... Recipient ok
DATA
354 Please start mail input.
This is a test for EECS 780.
I used cygwin to telnet to port 25 and manually composed.
.
250 Message queued for delivery
QUIT
221 Closing connection. Good bye.
Electronic Mail
Message Format

- Human-readable text
  - ASCII [RFC 2822]
    - formerly [RFC 822]
  - variable length fields
- Header fields
  - email metadata
  - variable number
- Body
  - text of email

<table>
<thead>
<tr>
<th>field-name</th>
<th>:</th>
<th>value</th>
<th>c_r l_F</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>field-name</th>
<th>:</th>
<th>value</th>
<th>c_r l_F</th>
</tr>
</thead>
<tbody>
<tr>
<td>c_r l_F</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

body
Electronic Mail
Header Fields (Sender)

- **Mandatory fields** [RFC 2822]
  - Date: `<day>, <date> <time> <time-zone>`
  - From: `<user>@<domain>` email address of sender

- **Common optional fields**
  - To: `<user>@<domain>`, list of recipients
  - Subject: `<subject-text>` meaningful subject
  - Cc: list of copied recipients
  - Bcc: list of blind recipients
  - Reply-to: reply address
  - X- comment field

- **postmaster@<domain>**
  - required to be valid address for administrative enquiries
Electronic Mail
Header Fields (Transit and Server)

• Trace fields
  – path email took through network
  – Received: from ⟨last-hop⟩ by ⟨this-hop⟩ with ⟨SMTP-info⟩
    for ⟨user⟩@⟨domain⟩
  – very useful for tracing email problems

• Server information fields
  – information such as spam identification

*Examine full headers of some emails*
you probably have to set an option
(e.g. “blah-blah” in Eudora)
Electronic Mail

Netiquette: Formatting

- Email was originally text-only with no formatting
  - many people still like it that way!
    - some people still use text only clients
    - increasing use of PDAs
  - not all clients are MS Outlook
- Unless pre negotiated with the recipient:
  - use plain text with no formatting
    - some email clients have formatting on by default
    - some misconfigured (MS-Exchange) servers format anyway
  - do not send email as HTML
  - do not embed images
Electronic Mail

Netiquette: Sender Identification Name

- Configure email client with your *name* in ISO-Latin
  - e.g. *Ima K.U. Student* <ima.student@eecs.ku.edu>
- CJK (Chinese, Japanese, Korean) encodings
  - appear as gibberish to older and non-GUI email clients
  - use only *after* ISO-Latin name
  - e.g. *James P.G. Sterbenz* 司徒傑莫 <jpgs@eecs.ku.edu>
Electronic Mail
Netiquette: Content and Attachments

• Content issues
  - use meaningful subject lines
  - spellcheck (most modern clients do this)
  - *think* before you send

• Simple emails should not be sent as attachments
  - e.g. MS-Word document containing “where are you now?”

• Do not send very large attachments unless
  - receiver is able to handle them (broadband access)
  - small enough to pass relay & server limits (typ. <10 MB)
Electronic Mail
Netiquette: Professionalism

• Email is like conversation, writing, and attire
  – adapt style to context
  – academic and professional more formal than with friends

• Style
  – punctuate and spell check
  – judicious jargon and emoticons only to someone you know
  – plain text, no embedded images

• Beware of free email accounts
  – may look unprofessional
  – frequently auto-append advertisements
  – construct meaningful signature no longer than 4 lines

• Proofread and *think before you send!*
Electronic Mail
Netiquette: Professionalism Examples

To: James Sterbenz <jgps@eecs.ku.edu>
From: Ima <cool_dude@stupidmail.com>
Subject: need a job!!!

e'm new at KU and really like it :-) 
e'm interested in everything and am bombing this email to all KU professors. I've stopped by your office in Eaton but your never there!!! :-P
do you even have office hours dude? LOL! Just tell me when I can drop by too learn what you do ru there???!!

cu later,
im

________________
Get your free email at Stupidmail.com!

Eat at Joe's: 1234 Mass. Lawrence, KS

To: James Sterbenz <jgps@eecs.ku.edu>
From: Ima Student <student@eecs.ku.edu>
Subject: Interest in ResiliNets group

Dear Professor Sterbenz,

I am a new M.S. student in the EECS department with interests in networking. I have looked at your Web pages and read the SUMOWIN paper. I am very interested in this research and will come to the next ResiliNets group meeting on Friday. I would like to discuss the possibility of you becoming my advisor and want to understand if there are any funding opportunities.

Sincerely,
Ima Student

---------------------------------------
Ima Student       EECS, Univ. of Kansas
student@eecs.ku.edu     +1 785 555 1212
Electronic Mail
Netiquette: Forwarding

- *Never* mass-forward emails that tell you to!
  - sometimes this is called a “thought virus”
  - goodtimes is the canonical example

- If it sounds questionable, it *is* bogus!
  - check www.snopes.com

- Jokes, political, and religious email
  - think first
    - do the recipients really want it?
  - use email lists and Internet for intended purpose
    - otherwise hundreds if not thousands of people will be angry
Electronic Mail
Safety Issues

• Be *very* careful of attachments
  – use a virus scanner that scans incoming and outgoing email
    • run full system scans weekly to catch emergent worms
  – don’t open attachments from sources that are from
    • untrusted sources
    • clueless individuals (including friends and relatives)
Electronic Mail
Safety Issues

• Very dangerous
  – executables
  – macro-enabled apps
  – compressed files (e.g. .zip, .tar)

• Risky
  – content with embedded scripts (should be disabled in apps)
    • MS Word, Excel (has gotten better with default warnings)
    • PDF

• Relatively safe
  – images and media clips
  – but be careful of MIME type associations
Electronic Mail
Privacy Issues

• Email is not private unless encrypted (e.g. with PGP)
  – assume that your sysadmin or corporate security is reading
  – consider separate work and personal accounts

• Sensitive information can be snooped in transit
  – never send credit card or banking information in the clear

• Email may be permanent record of correspondence
  – even if deleted by user
  – persists in backups

• Email content is easily altered
Electronic Mail
Binary Information

• Email was originally text-only
  – 7 bit ASCII

• Binary (8-bit) information had to be converted
  – manually encoded, included inline, decoded by receiver
  – e.g. uu{en|de}code in Unix
  – e.g. binhex on MAC

• Multimedia extensions to enhance process: MIME
Electronic Mail

MIME

- MIME (multimedia mail extensions) [RFC 2045, 2046]
- Additional mail header fields defines content
  - version MIME-Version: 1.0
  - encoding Content-Transfer-Encoding: base64
  - data type Content-Type: image/jpeg
Electronic Mail

MIME Attachments

- Multipart MIME
  - allows multiple files per email with different codings
- MIME header field
  - Content-Type: multipart/mixed; boundary=
- Most email clients automate this
  - sender attaches file(s)
  - receiver clicks on attachment(s)
Electronic Mail
SMTP Peer-to-Peer

• Receiving computer needs to be online to get mail
  – in 1970s/1980s office workstations continuously on
  – users dialed into their computers to remotely read mail
    • typically with Unix `mail` CLI or text-based full screen client

*Problem?*
Electronic Mail
SMTP Peer-to-Peer

- Receiving computer needs to be online to get mail
  - in 1970s/1980s office workstations continuously on
  - users dialed into their computers to remotely read mail
    - typically with Unix `mail` CLI or text-based full screen client

- Problem: PCs intermittently connected
  - home machines frequently powered down
  - laptops frequently disconnected
  - mail bounces with host unreachable

*Alternative?*
Electronic Mail
Email Servers: Inbound

• Use email servers to receive email
  – *not the same as SMTP server which sends email*
  – server always connected to net to receive email
  – administered and backed up by IT department
  – accessible from multiple user computers
  – de facto DNS names, e.g.
    • mail.ku.edu, pop.ku.edu, or imap.ku.edu
Electronic Mail

Email Servers: Outbound

• Also use email servers to relay outbound email
  – permits SMTP server to not be administered by user
  – sendmail.cf configuration is a dark art
    • simple defaults for SMTP from user machine to server
  – accessible from multiple user computers
  – de facto DNS names, e.g.
    • smtp.ku.edu, mail.ku.edu (port 25)
Electronic Mail

POP

- POP (post office protocol) [RFC 1939+2449]
  - emails downloaded from POP server to client
  - filters may select what is downloaded (e.g. less than size)
  - relatively simple protocol
  - very little state kept between sessions
  - optimised for client email storage of email
Electronic Mail

**IMAP**

- **POP (post office protocol)** [RFC 1939+2449]
  - emails downloaded from POP server to client
- **IMAP (Internet mail access protocol)** [RFC 1730]
  - email headers downloaded; user decides per email to read
    - may also store locally like POP
  - more complex protocols
  - significant state kept between sessions
  - optimised for server storage of email
Electronic Mail
POP/IMAP Example

- Each computer has
  - email UA and SMTP server *for outgoing email*
- Institution has
  - POP or IMAP server *for incoming email*
Electronic Mail
POP/IMAP Example

1. Alice composes email to bob@u.edu
2. Alice’s UA sends to her server which queues message
3. Client side of Alice’s server opens TCP 25 port to ku.edu
4. SMTP client sends message
5. SMTP server ku.edu receives, inserts in Bob’s mailbox
Electronic Mail
POP/IMAP Example

1. Alice composes email to \texttt{bob@u.edu}
2. Alice’s UA sends to her server which queues message
3. Client side of Alice’s server opens TCP 25 port to \texttt{ku.edu}
4. SMTP client sends message
5. SMTP server \texttt{ku.edu} receives, inserts in Bob’s mailbox
6. Bob invokes client
Electronic Mail

POP/IMAP Example

1. Alice composes email to bob@u.edu
2. Alice’s UA sends to her server which queues message
3. Client side of Alice’s server opens TCP 25 port to ku.edu
4. SMTP client sends message
5. SMTP server ku.edu receives, inserts in Bob’s mailbox
6. Bob invokes client
7. client POPs down email (or gets headers); can then be read
Electronic Mail

Web Mail

- Web-based email increasingly common
  - web browser is used as email client
    - user doesn’t need to have or configure separate client software
- Web mail server used to store mail
  - always available to receive email on behalf of user
  - may permit access using standard protocols (IMAP)
Electronic Mail
Web-Based Free Mail Services

• Examples: Hotmail, Yahoo mail, GMail
  – completely decouples email identity from user ISP accounts
    • users keep email address independent of ISP
    • lowers barrier to anonymous email and spam
      – spam blacklists frequently filter these email addresses

• Gmail
  – has managed to maintain more professional reputation
    • doesn’t append advertisements
    • has recently received bad press for user data mining
  – interesting hacks
    • GMail as a free file server
Electronic Mail
Reliability

*Is email reliable?*
Electronic Mail
Reliability

• Email does not provide end-to-end reliability
  – *bounce* messages when SMTP delivery fails
    • nothing to do with POP or IMAP access
  – no end-to-end *positive* acknowledgement

• Mail delivery failure resulting from many errors
  – sender: user, client, SMTP server
  – transit
  – receiver: server, client, user
  – bounce messages: subject to most of the same errors
Electronic Mail
Errors: Sending User

Errors?
Electronic Mail
Errors: Sending User

- User enters invalid or wrong email address
  - bounce if invalid
  - may get reply from non-intended recipient
- User doesn’t use meaningful subject line
  - spam-block fodder
  - ignored, misinterpreted, or mis-sorted by receiver
- Message content likely to be spam blocked
  - looks like ad, MMF (make money fast), or 419 scam
  - contains ads from free email services (e.g. Hotmail)
- User name blank or not in ISO Latin
  - CJK may be spam blocked, mis-sorted, or ignored
Electronic Mail
Errors: Sending Client and SMTP Server

Errors?
Electronic Mail
Errors: Sending Client and SMTP Server

• Email client auto-completes to different email address
  – may get email reply from non-intended recipient

• Outgoing virus/trojan filter blocks
  – email client
  – personal firewall
  – institutional firewall

• SMTP communication failure
  – misconfigured sending system
  – SMTP relay blocking or requirements on From: address
  – not authorised in guest domain

• Sending SMTP server node down
Electronic Mail

Errors: Transit

*Errors?*
Electronic Mail

Errors: Transit

- Problems with network connectivity
- SMTP relay nodes
  - down
  - message size exceeded
  - spam blocked
Electronic Mail

Errors: Receiving Server

Errors?
Electronic Mail

Errors: Receiving Server

- Receiving mail server or network down
- Server doesn’t have space
  - full disk or user quota exceeded
  - disk crash losing already received email
- Spam filters
  - institutional firewall blocks and discards
    - based on headers and message content
    - based on sender IP block (e.g. spammer using your ISP)
  - email server blocks and discards
  - email server discards or tags subject line as spam
Electronic Mail
Errors: Receiving Client and User

Errors?
Electronic Mail

Errors: Receiving Client and User

- Receiving mail client
  - POP or IMAP protocol failure
  - doesn’t have enough disk space
  - spam
    - personal firewall or client blocks and discards
    - filtered to spam folder and ignored
    - previously tagged mail filtered to spam folder and ignored
  - misfiltered
    - complex and conflicting rules

- User
  - fails to notice due to mail overload
Electronic Mail

Errors: Bounce Messages

Errors?
Electronic Mail

Errors: Bounce Messages

- Bounce messages subject to same errors
  - sender
  - transit
  - receiver
- Sender doesn’t know original message bounced
Electronic Mail

End-to-End Acknowledgements

- Attempts to provide E2E positive ACKs:
- Return receipts
  - non-standard; discouraged in RFC 2076
  - header fields `Return-receipt-requested:` and `Return-receipt-to:`
  - many people consider them intrusive
- MDN: message disposition notification [RFC 3798]
  - header field `Disposition-Notification-To`
  - disposition returned as multipart MIME message
    - displayed (but doesn’t guarantee was read or understood)
    - deleted (but might be undeleted)
Electronic Mail

Mail Lists

• Mail lists with central administration
  • generally more convenient than per user address book lists

• Examples
  – LISTSERV (originally from BITNET)
  – mailman and majordomo (Unix-based)
  – Google groups

• Mail list netiquette (or how to avoid the wrath of thousands)
  – in big lists default Reply-to: should be to sender, not list
  – never email to more than one list per email
  – never construct mail list containing an entry that is mail list
    • subject to mail storms that say “remove me”
    • may cause mail loops
Electronic Mail Trends

- Disk storage is very cheap (< $50/TB)
  - mail quotas for normal users make no sense
  - more expensive to selectively delete mail than to store it
- New challenge is how to manage this information
  - how to search for string in body of 1,000,000 emails?
Electronic Mail

Spam: Problem

- Spam: unsolicited bulk email (UBE)
- Problem
  - cost in time of spamee to process
  - significant resources: bandwidth and mail server storage
    - 90 – 95% of all email is spam!
Electronic Mail

Spam: Issues

• Spam: unsolicited bulk email (UBE)

• Issues
  – essentially free to send (unlike junk postal mail)
  – easy to hide identity of sender
    • throw-away domains
    • forged sending address
  – increasingly botnets of infected Windows PCs source spam
    • some estimate that 25% of Windows PCs are in a botnet
    • Conficker 2008 botnet: ~10 M PCs sending 10 Gspams /day

Solutions?
Electronic Mail
Spam: Solution Attempts

- Current spam reduction techniques
  - spam filters (e.g. SpamAssassin open source filters)
    - server e.g. SpamAssassin
    - personal firewall e.g. Kaspersky
    - email client e.g. Eudora spam filters
  - blacklisting, e.g. Spamhaus
    - users
    - domains (many false positives)
    - legislation: legal deterrents
- But much gets through
  - cat-and-mouse game

Proposals?
Electronic Mail
Spam: Solution Proposals

- Whitelisting
  - very annoying pre-negotiation with each new user
- Authentication of sender
  - harder to hide behind forged email and IP addresses
- Postage to send email
  - really cheap (e.g. 1¢ per email sent)
  - almost free for normal user...
  - ...but too expensive to send millions of spam messages
  - other mechanisms for legitimate email lists
- Better security to limit botnets
  - much easier said than done
Networked Applications

AL.4.2 Chat

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
AL.4 Telepresence applications
  AL.4.1 Email
  AL.4.2 Chat
  AL.4.3 Conferencing
AL.5 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Chat: users exchange messages in real time
  - analogue to live discussion
  - but with text based interface
  - example of interactive application
    - early example: ntalk on Unix
  - current examples
    - SMS (text message) in mobile PSTN
    - Skype chat
Chat Components

- Component: chat or IM (instant messaging) client
  - proprietary examples
    - Skype, Netscape, AOL, MSN, Yahoo, ...
    - incompatibility a problem
      - metaclients, e.g. Trillian
  - non-proprietary: IRC (Internet relay chat)
    [RFC 1459, 2810, 2811, 2812, 2813]

- Supporting infrastructure
  - chat servers
Networked Applications

AL.4.3 Conferencing

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
AL.4 Telepresence applications
  AL.4.1 Email
  AL.4.2 Chat
  AL.4.3 Conferencing
AL.5 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Telepresence
Conferencing

- Audio- and video-conferencing
  - real-time synchronisation end-to-end

*More about this in Lecture MS*
Teleconferencing
Traditional Telephony

- Audio call between two participants
  - using POTS (plain ordinary telephone service) phones
    - *not* plain old telephone service!
  - caller dials callee
  - all intelligence in the network

- Multiparty teleconference
  - conference *bridge* mixes audio and deliver to all parties
Teleconferencing
Overview

- Audio or video conference among client computers
- Client GUI used to:
  - interface with directory, if any (e.g. skype)
  - set user preferences and bind to installed codecs
  - send/receive real-time traffic, e.g. RTP
  - session management, e.g. call, join, etc.

- Topologies
  - mesh of clients
  - star with reflector

Tradeoffs?
Teleconferencing
Topologies: Mesh of Clients

- Clients each negotiate pair-wise with one-another
  - selection from set of compatible codecs (coder/decoders)
  - each client can display as desired

Performance issues?
Teleconferencing
Topologies: Mesh of Clients

- Clients each negotiate pair-wise with one-another
  - selection from set of compatible codecs (coder/decoders)
  - each client can display as desired

- Performance issues
  - significant burden on clients & access links: $O(n^2)$
  - no central point of administration or failure
Teleconferencing
Topologies: Star with Reflector

- Reflector is shared resource (e.g. Cornell CUSeeMe)
  - transcodes to common format
  - audio mix and video tile; all clients see same view

*Performance issues?*
Teleconferencing
Topologies: Star with Reflector

- Reflector is shared resource (e.g. Cornell CUSeeMe)
  - transcodes to common format
  - audio mix and video tile; all clients see same view

- Performance issues
  - less resource burden on client end systems & access links
  - central point of administration, failure, and bottleneck
Networked Applications

AL.5 Distributed Computing and Storage

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
  AL.5.1 Remote login
  AL.5.2 Peer-to-peer file swarming
  AL.5.3 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Internet Applications
Distributed Computing and Storage

• Characteristics
  – arbitrary
    • relationship between computing & storage element
    • granularity: unit of information requested
  – bandwidth symmetry
    • application dependent
  – end-to-end synchronisation: application dependent

• Key components
  – processing elements
  – storage elements
Networked Applications

AL.5.1 Remote Login

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
  AL.5.1 Remote login
  AL.5.2 Peer-to-peer file swarming
  AL.5.3 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Distributed Computing
Remote Login

• Logging in to a remote computer system
  – one of the first killer apps that drove demand for networks
    • file transfer, remote login, email

• Components
  – login client at the user
  – server is the computer

• Examples
  – pvm in BITNET (historic)
  – telnet
  – SSH – secure shell
  – XWindows – remote graphical windows
Remote Login

Telnet

- Telnet [RFC0854] + many options RFCs
  - bidirectional byte-oriented communication
  - intended primarily for terminal–host communication
Remote Login
Telnet Components

• Components:
  – client:
    • interacts with user on local system
    • opens TCP connection between client and server computers
    • default port 23 for remote login
  – client–server protocol: telnet
  – server:
    • terminal interface to remote system

• Security concerns
  – traffic not encrypted
Remote Login
SSH

- SSH (secure shell) [RFC 4250–4256]
  - encrypted tunnel for remote login
  - prevents eavesdropping on sensitive information
  - secure file transfer also supported
  - user authentication support

More on mechanisms in Lecture IS
Distributed Computing
Remote GUI Access

- Remote login limited to text access
  - character granularity (e.g. DEC VT100)
  - screen granularity (e.g. IBM 3270)
- Extension to window-based GUI
  - 1973: Xerox PARC Alto (research system)
    - WIMP paradigm – windows, icons, menus, pointers
  - 1981: Xerox 8100 Star
  - 1984: Apple Macintosh
  - 1985: MS-Windows
- Example of remote GUI access
  - 1984: MIT X (X Window System)
Networked Applications

AL.5.2 Peer-to-Peer File Swarming

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
   AL.5.1 Remote login
   AL.5.2 Peer-to-peer file swarming
   AL.5.3 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Peer-to-Peer File Sharing
Wasn’t Really Peer-to-Peer

- Peer-to-peer file sharing
  - wasn’t originally peer-to-peer at all!
  - it was client-server
    - typically using HTTP
  - node are sometimes clients
    sometimes servers (servents)

*Problem?*
Peer-to-Peer File Sharing

Wasn’t Really Peer-to-Peer

- Peer-to-peer file sharing
  - wasn’t originally peer-to-peer at all!
  - it was client-server
    - typically using HTTP
  - node are sometimes clients
    sometimes servers (servents)

- Problem: *hot spots*
  - servers of popular content become bottlenecks
    - Zipf’s law: popularity inversely proportional to rank
  - overwhelmed with requests

*alternative?*
Distributed File Storage

File Swarming

• P2P file *swarming*
  – true P2P file distribution
• Files broken into chunks
  – chunks shared
  – distributed among clients
  – until all clients have complete file
• Example
  – BitTorrent
Decentralised P2P File Swarming

BitTorrent Overview

- Infrastructure and protocol for file swarming
  - seeds
  - trackers
  - peers
Decentralised P2P File Swarming

BitTorrent Seeding

- **Seed** servant with file to share
  - creates `.torrent` descriptor file
    - filename, size, SHA1 fingerprint
    - address of tracker server
  - `.torrent` file distributed to interested peers

*problem?*
Decentralised P2P File Swarming

BitTorrent Seeding

- **Seed** servant with file to share
  - creates `.torrent` descriptor file
    - filename, size, SHA1 fingerprint
    - address of tracker server
  - `.torrent` file distributed to interested peers
    - torrent servers point of legal attack by RIAA and MPAA
      - even though they don’t necessarily host content...
        as with Napster
Decentralised P2P File Swarming

BitTorrent Tracker

- **Tracker** server maintains list of
  - servents receiving file
  - location of *chunks* available for download

- **Example:** The Pirate Bay
  - Swedish ad-funded bittorrent site
  - raided in 2006 by police
  - founders convicted in 2009 of copyright infringement
    - appeal upheld in 2012 by Swedish Supreme Court
  - Pirate Bay is still up and running with proxies worldwide
Decentralised P2P File Swarming

BitTorrent Joining

- Downloading peer joins the torrent
  - typically clicks on .torrent hyperlink on tracker
- Tracker randomly selects set of peers in torrent
  - returns IP addresses to download peer
- Download peer establishes TCP connections to others
Decentralised P2P File Swarming

BitTorrent Downloading

- Peers periodically query others for list of chunks
- Request missing chunks from peers that have them
  - request *rarest first*: helps propagate less-distributed chunks
Decentralised P2P File Swarming

BitTorrent Serving

- Peers measure bandwidth of chunk arrival
- Serve chunks on 4 highest rate TCP connections
  - *unchoked peers* recalculated every 10 s
- Serve chunks to other peer
  - *optimistically unchoked peer* recalculated every 30 s
  - *tit-for-tat*: rate-compatible peers exchange with one-another
Networked Applications

AL.5.3 Distributed Computing and Storage

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
  AL.5.1 Remote login
  AL.5.2 Peer-to-peer file swarming
  AL.5.3 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Distributed Computing
Overview and Options

• Motivation: flexibility
• RPC: remote procedure call
  – process on one computer calls execution on another
• Distributed virtual memory
  – virtual memory space distributed among networked systems
• Cluster computing
  – multiprocessors distributed across a network
    • system area network (SAN)
Distributed Storage
Overview and Options

• Motivation: flexibility
• SANs (storage area networks)
  – interconnection of processors and memories in a local area
• NAS (network attached storage)
  – using network to access storage, perhaps over a wide area

much more on these in KU EECS 881

• Distributed file systems
Distributed Storage
Network Attached Storage

• Network attached storage (NAS)
• **SCSI:** small computer systems interface (pr. scuzzy)
  – interface and control commands for peripheral attachment
  – commonly used for disk drives and other mass storage
    • high performance bus
    • physical specifications for very short range
• **iSCSI:** Internet SCSI  [RFC 3720]
  – generalisation of SCSI to operate over the Internet
  – iSCSI PDUs (header+data) transported over TCP
Distributed Storage
Distributed File Systems

• Distributed file systems
  – allow files to be distributed throughout the network
  – may or may not be transparent

• NFS (network file system) [RFC 3530]
  – originally implemented by Sun over UDP for LAN file servers
  – allows local systems to *mount* files on remote systems
  – later extended to TCP and WANs with influence from AFS

• AFS (Andrew file system)
  – originated from CMU project Andrew
  – designed for greater security and scalability than NFS

• Others ...
Distributed Computing and Storage

The Grid

- **Grid computing**
  - paradigm for sharing heterogeneous resources
  - generally supercomputers and other specialised components
- Middleware for distributed resource management
  - roughly corresponding to session layer 5
- Protocols in support of high-performance computing

*EECS 881*
Open Grid Forum Documents

- Open Grid Forum  [www.ogf.org](http://www.ogf.org)
- OSGA: open grid services architecture
  - framework for integration, virtualisation, and management
  - services
    - execution, data, resource management, security, self-management, information
- Resource management specifications
  - JSDL: XML-based job submission description language
- Grid file system
  - under development
- GridFTP extensions to FTP
Distributed Computing and Storage

Cloud Computing

- **Cloud computing**
  - paradigm for using computing services in public Internet: “the cloud”
Distributed Computing and Storage

Cloud Computing

• Roots
  – 1960s time sharing services
    • back to the future: yet another return to old ideas
  – outsourcing of IT services

Advantages?
Distributed Computing and Storage
Cloud Computing

• Roots
  – 1960s time sharing services
    • back to the future: yet another return to old ideas
  – outsourcing of IT services

• Advantages
  – typical outsourcing advantages, e.g. downsizing of IT staff
  – eliminate need for software and infrastructure expertise
  – fine-grained provisioning and payments for capabilities

Disadvantages?
Distributed Computing and Storage

Cloud Computing

• Roots
  – 1960s time sharing services
    • back to the future: yet another return to old ideas
  – outsourcing of IT services

• Advantages
  – typical outsourcing advantages, e.g. downsizing of IT staff
  – eliminate need for software and infrastructure expertise
  – fine-grained provisioning and payments for capabilities

• Disadvantages
  – loss of control
  – privacy issues: data on 3rd party servers
Distributed Computing and Storage

Cloud Computing

- NIST cloud computing taxonomy
  [http://csrc.nist.gov/groups/SNS/cloud-computing]
  - characteristics
  - service models
  - deployment models
Distributed Computing and Storage

Cloud Computing Characteristics

- Cloud computing characteristics
  - on-demand self-service
  - broad network access across a variety of platforms
  - resource pooling among users
  - rapid elasticity of provisioned resources
  - measured services
Distributed Computing and Storage

Cloud Computing Service Models

• Cloud computing service models
  – SaaS: cloud software as a service
    • software such as email servers, office suites
  – PaaS: cloud platform as a service
    • users develop and deploy applications in the cloud
  – IaaS: cloud infrastructure as a service
    • storage, compute servers
Distributed Computing and Storage

Cloud Computing Deployment Models

- Cloud computing deployment models
  - private: operated solely for a particular organisation
  - community: shared by organisations with common interest
  - public: available to general public and industry as service
  - hybrid: two or more unique clouds interacting
    - e.g. cloud-bursting for load balancing
Internet History
Cloud Computing: Providers

- Example providers and services
  - Amazon Web Services (AWS)
    - EC2 (Elastic Computing Cloud)
    - EBS (Elastic Block Store)
  - Google cloud services
    - Google Docs
    - Google App Engine
    - Google Cloud Storage
  - IBM SmartCloud
  - Microsoft
    - Office 365
    - Windows Azure
Networked Applications

AL.6 Application Adaptation & Net Interaction

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Application Adaptation Overview

- **Goal**
  - optimal interaction between application and network

- **Applications can adapt to the network environment**
  - available bandwidth
  - end-to-end delay
  - loss and error characteristics
Application Adaptation Techniques

- **Caching**
  - reduces delay and aggregate bandwidth

- **Progressive refinement and structured data**
  - presents useful information more quickly to user
  - information access applications

- **Adaptive quality of services**
  - tunable frequency range, image resolution, frame rate
  - telepresence applications

- **Compression**
  - reduces storage and bandwidth at the cost of processing
Application Adaptation
Compression

- Compression to reduce
  - bandwidth and delay
  - total delay includes
    - transmission delay
    - compression/decompression
  - benefit tradeoff between
    - path bandwidth
    - processing rate and cycles
Application–Network Interaction

Cross-Layer Optimisations

- Mechanism to cross layer boundaries
- Knobs influence network behaviour from above
- Dials provide network characteristics to application

EECS 882 and 983
Networked Applications

AL.7 Social Networking

AL.1 Services, interfaces, and functions
AL.2 Application characteristics and types
AL.3 Information access applications
AL.4 Telepresence applications
AL.5 Distributed computing and networked storage
AL.6 Application adaptation and network interaction
AL.7 Social networking
Social Networking
Hybrid Layer/Plane Cube

“Layer 8”: social interactions over social network services implemented at layer 7
Social Networking
Layer 8

- Layer 7 is conventionally the top of the stack
  - standardised in the OSI model
  - conventionally used to describe application layer in Internet
- Higher layer numbers have occasionally been used
  - sometimes as a joke: financial, political, religious
  - sometimes seriously for organisation structures using the net
- In these notes, “layer 8” indicates social organisation
  - not the applications themselves
  - virtual communities have become very significant
    - 100’s of million users
Social Networking
Prehistory: BBSs

• Network and Internet subcultures date from 1970s
  – communities developed on Usenet, BBSs, and mailing lists

• BBS: bulletin board systems
  – dial-up modem banks on a server
    • generally consumer PC
    • initially 110 or 300 b/s
  – 1976: first BBS in Chicago

• 1985: The WELL (Whole Earth ‘Lectronic Link)
  – migrated from BBS to ISP-based virtual community
  – (still) a paid subscription
Social Networking
Prehistory: Usenet

- 1980: began at Duke over UUCP
  - distributed BBS over pre-Internet
  - developed news servers, client: rn, NNTP protocol
- 1987: great renaming into current structure
  - initially “big 7” with limits on content and group creation:
    comp.*, misc.*, news.*, rec.*, sci.*, soc.*, talk.*
  - alt.* added to open content (including binary files)
- Interfaces evolve over time
  - CLI (rn) → full-screen threaded (trn, pine, tin)
  - graphical interfaces (e.g. Netscape news reader)
  - Web-based (e.g. Google groups)
Social Networking
Prehistory: Usenet Netiquette

• First social mores develop in context of Usenet:
  netiquette (network ettiquite)
  – response to explosion of Usenet in early 1990s
  – RFC 1855: Netiquette Guidelines 1995

• Guidelines for acceptable interactions
  – signature length limited to 4 lines
  – tone of communication
    • case (UPPER CASE is shouting), emoticons, flamewars
  – on-topic posting and limits to cross-posting
    • lead to early definitions of what became spam

• Became basis for email netiquette
Social Networking
Prehistory: LISTSERVs

• Earlier email lists were manually administered
  – sysadmin would add individuals to file
  – not scalable for large dynamic lists

• 1986: LISTSERV software automates process
  – users can self-subscribe, optionally with list owner approval
  – permitted the emergence of *large discussion groups*
  – was the “Usenet of BITNET” for some years

• Unix-based software follows
  – 1992: Majordomo
  – 1999: GNU Mailman
Social Networking

Prehistory: Web-Based Communities

- 1994: Web-based communities
  - Tripod: 1995 – now owned by Lycos
  - Angelfire 1995 – into Tripod now Lycos

- personal web pages for the masses
  - first significant *content generated by the masses*
  - online tools for web page authoring

- communities of interest, chat rooms

- combination of ad-sponsored and paid subscriptions
Social Networking
Slashdot and Reddit

- **Slashdot**  [www.slashdot.org](http://www.slashdot.org)
  - “News for Nerds. Stuff that Matters”
  - 1997 founded by Rob Malda (Commander Taco)
  - news items are submitted by the user community
    - with users responding in a dialogue thread
  - complex moderation and rating scheme

- **Reddit**
  - 2005 founded by Huffman and Ohanian
  - discussion and social interactions
  - topic area divided by subreddits

- Both have developed their own subculture
Social Networking

Wikipedia

- Wikipedia wikipedia.org
  - free online encyclopedia anyone can edit
- Significant subculture and protocols for editors
  - look at these policies and on page discussions
Social Networking

Blogosphere

• Blog: contraction of web log
  – running commentary published on the web
  – currently several hundred million blogs
  – blog services store and publish with Web-based interfaces
    • Blogspot (now blogger)
      – e.g. http://jpgsterbenz.blogspot.com/
    • Wordpress
      – e.g. https://jpgsterbenz.wordpress.com/

• Blogosphere: blogs and their interrelationships
  – wide variety of opinions, including many kooks
  – the blogosphere can have a “buzz”
    • but no netiquette
Social Networking Overview

- **Social networking**
  - community of users enabled by a Web GUI
    - each user has online presence defined by profile
    - users become “friends” of one another
      - located by searches or friend-of-friend relationships
    - users are individuals or corporate entities
      - users can create groups of common interest

- **Social networking**
  - users update their status and interact in with one another
  - post multimedia content such as images
  - interact with 3rd party applications
Social Networking
Recent History

- Selected social networks
  - 2002: Friendster
  - 2003: MySpace
  - 2003: LinkedIn
  - 2004: Facebook
  - 2006: Twitter
Social Networking

Friendster

- Friendster established in 2002
  - a way to meet new people
  - currently ~115 M users
  - no longer significant market share in SU
    - has become Asia-centric: 90% of traffic
Social Networking

MySpace

- MySpace: 2003 by former Friendster employees
  - bought in 2005 by News Corp. (Fox)
  - revenue model: free supported by advertising
- Most popular social networking site in mid 2000s
  - overtaken by Facebook in 2008; new features to compete
  - laid off 30% employees in 2009
  - shifted to younger demographics; now music emphasis
- User profile pages highly customisable using HTML
  - username based URIs, e.g. http://www.myspace.com/jpgsterbenz
- Famous Megan Meier suicide Lori Drew cyber-bullying
Social Networking

LinkedIn

• LinkedIn formed 2003 for professionals
  – tracks professional and business connections
  – ~50 M users

• Use
  – users invite contacts and recommend one another
    • supervisor recommendations inappropriate, but common

• Revenue model
  – basic service free
  – encourages upgrade to fee-based premium service

• Privacy concerns
  – ability to data-mine business relationships by for-profit
Social Networking

Facebook

- Facebook established 2004 at Harvard
  - Facemash as online directory for students to meet opened up to other colleges; high schools in 2005
  - 2006: open access to all over 13 years old

- Most popular social networking service (2008)
  - now the most popular social net site: > 350 M users
  - 2nd ranked for Web traffic (after Google, before YouTube)
  - independent but investment from VCs, Microsoft, etc.

- Revenue model (claimed profitability in 2009)
  - free supported by advertising
  - controversy over giving advertisers private information
Social Networking
Facebook Environment

• Relatively structured environment: user walls
  – limited customization capabilities
  – followed MySpace with usernames
    e.g. http://www.facebook.com/jpgsterbenz

• 3rd party applications
  – Facebook development platform
  – content delivery, games (e.g. FarmVille, MafiaWars), etc.
  – significant privacy risk in leaking profile information
  – restrict behaviour they don’t like (e.g. SocialFixer)

• Concerns typical to social networking services
  – privacy, cyber-stalking, etc.
Social Networking
Facebook Demographics

- Facebook demographics
  - become choice for older more educated users

- Significant use in academia
  - EECS 780: http://www.facebook.com/group.php?gid=237076787361
  - KU EECS: http://www.facebook.com/KU.EECS
  - KU ITTC: http://www.facebook.com/KU.ITTC
  - KU Engineering: http://www.facebook.com/kuengineering
  - KU: http://www.facebook.com/KU
Social Networking

Twitter

• Twitter launched in 2006
  – text-based messaging among multiple platforms
  – essentially pub/sub multicasting text messaging
  – micro-blogging and news distribution service

• Publish/subscribe model
  – users *tweet* 140 character messages
  – delivered to followers that have subscribed to user feed

• Revenue model
  – independent; funded by VC

• Privacy issues
  – significant exposures for those that don’t restrict followers
Google Wave launched late 2009
- personal collaboration and communication tool
- merges document collaboration and social networking
- general release mid-2010 never caught on
- Google suspended development late 2010
  - development taken over by Apache Foundation

Google Buzz launched early 2010
- Gmail integrated social networking enhancements
- shut down late 2011 in favour of Google+
Google+ launched mid 2010
  - initially by invitation only
  - each user had a few tokens to invite others
    • typical Google ramp-up and scarcity-induced buzz
  - currently ~ 500M users; ~ half of which are active
    • has attracted users unwilling to joining Facebook

New features
  - finer-grained posting control: circles
    • Facebook responded with lists
  - multimedia hangouts
Social Networking

Google+

- Growth considerably slowed since 2012
  - clearly not a threat to Facebook
  - registered users spending far less time on G+ than FB
- Google attempting to increase market share
  - e.g. terminating Google Latitude
Social Networking

Netiquette

• Social networking in its current form is immature
  – netiquette and other mores have not yet been established
  – and majority of people unfamiliar with netiquette

• Social network friend $\neq$ real-life friend
  – real life friend
  – acquaintance
  – professional contact
  – friend-of-a-friend
  – anyone who requests friending (more friends are better?)

• Many people are unaware of who is seeing postings
  – e.g. employers, schools, law enforcement, parents
Social Networking Concerns

- Current social networking sites commercial entities
- Facebook is notorious for antisocial behavior
  - mine personal information for profit
  - regularly change UIs
    - include privacy settings to be less strict by default
  - seem to try new things before thinking through them
    - then retreat if user outcry is large enough
  - third party apps harvest and relay private information
- Users don’t understand consequences of actions
  - privacy
  - lack of contexts (e.g. co-workers see party photos)
Social Networking

Diaspora

• Diaspora (Greek διασπορά for dispersion)
  – distributed social networking service
  – Facebook-like functionality
  – information and privacy under control of users

• Servers organised as pods
  – hosted by user community (e.g. diaspora.ku.gpeni.net)
  – public servers

• Alpha release late 2010
  – accounts available now on diasp.org alpha server
  – accounts available soon on diaspora.ku.gpeni.net
    • to KU and GpENI community
Internet Currency

Bitcoin

- Decentralised virtual currency
  - Nakamoto 2008
  - open-source software 2009
Networked Applications

Additional References


• David Gourley and Brian Totty *HTTP: The Definitive Guide*, O’Reilly, Sebastopol CA, 2002

Networked Applications

Acknowledgements

Some material in these foils is derived from the textbook supplementary materials:

- Kurose & Ross, *Computer Networking: A Top-Down Approach Featuring the Internet*
- Sterbenz & Touch, *High-Speed Networking: A Systematic Approach to High-Bandwidth Low-Latency Communication*  
  http://hsn-book.sterbenz.org