IT441: Network Servers & Infrastructure

CLASS 10 : 08 Nov 2004
13:30 – 16:15

Last Time:

- TCP
- some basic protocols for getting started
  - RARP
  - BOOTP
  - DHCP
- Domain Name Service
- sample service: mail
This Time

- a TCP vulnerability
- Internet service: VoIP
- Internet service: WWW

Address Services

- what’s in a name?
  - user@somenode.subdomain.domain
- where does “user” get its email?
- what about
  - user@someothernode.subdomain.domain
Address Services

- many organizations now use email gateway
  - node dedicated to receiving, forwarding, sending email
- lets users have email addresses like
  - username@domain
- gateway receives all inbound email
  - either forwards to particular node, or,
  - keeps local and user’s user-agent retrieves the email therefrom

Address Services

- gateway receives all inbound email
  - either forwards to particular node
- gateway needs list of entries associating
  - username@domain with
    username@node.subdomain.com
- must this association be 1:1?
Address Services

- must this association be 1:1?
  - no: can have one name: many names
    - so email gateway can match a listname with $\geq 1$ mailboxes and must generate individual copies of the message to each
    - such a gateway is hence known as a mail-exploder
    - see Comer fig. 32.5 for example
    - users must have their email address added to list in order to receive mailings

- adding name to list can be automated procedure:
  - usually simply send email message to list
  - follow syntax rules specified by list, e.g., send subject line as “add me”
  - no body to msg (not being read by human)
  - list manager on email gateway watches for these special msgs
    - performs add when discovers one to do
    - analogously for removal
Getting Your Mail

- today, email usually is delivered to some server whence you must retrieve it
  - rather than being delivered to ‘your’ local machine
  - i.e., your email user agent may be local, but the actual mailbox isn’t
- how to retrieve?

Post Office Protocol (POP)

- currently POP-3 as described in RFC1939
- POP is client–server pair
  - server runs on system where email physically is collected
  - client runs on system where user agent runs
- so mail server runs
  - software to collect email directed to it, including exploding function
  - software to support access to collected email via POP
Post Office Protocol (POP)

- so mail client runs
  - software to send email (via SMTP)
  - software to receive email (via POP)
- POP client interacts with server to,
  - copy accumulated email from mail server to user client
  - delete copied messages from email server, or leave intact
- service runs on port 110

Post Office Protocol (POP)

- example client server interaction in POP3:

  *connect to port 110 on server...*

  +OK POP3 server ready

  USER fred
  +OK

  PASS derf
  +OK user logged on

  LIST
  +OK 2 messages (320 octets)
  1 120
  2 200

  .
Post Office Protocol (POP)

- example client server interaction in POP3, cont’d:
  
  ```
  RETR 1
  +OK 120 octets
  <the POP3 server sends the entire message here>
  
  DELE 1
  +OK message 1 deleted
  DELE 1
  -ERR message 1 already deleted
  QUIT
  +OK POP3 server signing off
  ```

POP3

- available commands in the POP3 protocol include:

  ```
  APOP  PASS  STAT
  DELE  QUIT  TOP
  LIST  RETR  UIDL
  NOOP  RSET  USE
  ```
Another Mail Fetching Protocol

- instead of POP, a user agent may use Internet Mail Access Protocol (IMAP) RFC3501
- particular advantage: allows management of messages on a server
  - e.g., can organize messages into folders
- also provides ability to retrieve only parts of a message
  - e.g., subject lines
  - e.g., specific part of MIME multipart message

TCP Vulnerability

- news reports ‘newly-discovered’ “major” flaw in TCP
- released as “Vulnerability Issues in TCP” by UK’s National Information Security Co-ordination Center
- details being presented at CanWest Security 2004 (Vancouver)
- reports claim can affect/disrupt large parts of Internet
- not really a “new” problem
  - e.g., CERT advisory from 3 years ago
TCP Vulnerability

- may achieve DOS or DDOS
- 3 kinds of attack possible:
  1. **RST-attack**: premature termination of a TCP connection
  2. **SYN-attack**: premature termination of a TCP connection
  3. **data injection attack**: insert false data into a TCP connection

Basis for Attack

- attacker generates fake packets that receiving end acts upon
  - e.g., by performing a RST on connection (DOS)
- packets need to fake:
  - src IP and port number
  - sequence number
- vulnerability increases with duration of connection
  - BGP connections particular concern
Sequence Numbers

- $0 \leq SN \leq 2^{32} - 1$ so attacker needs to guess
  - expected value: $2^{31} - 1$ or $2,147,483,648$ attempts
- TCP receiver accepts an SN within window
- window size established in 3-way handshake at connection start
- typical sizes today: 32 to 64 Kbytes
- reduces difficulty of SN guess to 1 in 64K or 32K
- at DSL speeds, 1 in 64 K guessable in $< 200$ seconds

Countermeasure: RST attack

- IETF suggests modifying behaviour on RST from:
  
  ```java
  if (RST bit set)
      if (SN outside expected range) silently drop seg
      else immediately perform RST on connection
  else
      send ACK
  ```

- to:
  
  ```java
  if (RST bit set)
      if (SN outside expected range) silently drop seg
      else if (SN == expected SN value) immediately RST
      else send ACK
  ```

  only reset if SN is exactly expected value

- provides “some” protection (not a complete solution)
Countermeasure: SYN attack

- IETF suggests modifying behaviour on SYN from:
  
  ```java
  if (SYN bit set)
    if (SN outside expected range) send ACK to peer
    else send RST to peer
  ```

- to:
  
  ```java
  if (RST bit set)
    if (SN outside expected range) send ACK to peer
    else if (SN == expected SN value) {
      send ACK to peer with ACK value - 1
    }
    else send ACK
  ```

- provides complete protection

Countermeasure: Data Injection Attack

- ACK value of any segment is valid so long as doesn’t ack data ahead of next segment to send
- attacker needs to guess 2 ACK values for each guessed SN
- IETF suggests:
  - additional test on incoming seg: accept ACK only if:
    
    `(SND.UNA - MAX.SND.WND) <= SEG.ACK < SND.NXT`
  - must guess SN in window and ACK value from restricted range
  - complete solution: deploy RFC2385 between endpoints
Other Suggested Countermeasures

- general:
  - use IPSEC network layer encryption
  - reduce TCP window size
  - avoid publishing TCP source port info
- BGP:
  - use MD5 signature option
  - use ingres (RFC2827) and egres filtering

Servers

- provide a service
- provide content
  - static
  - dynamic
    - generated on-demand
  - “streamed”
    - on-going flow of data
    - often real-time sensitive
    - may be pre-computed (e.g., mp3)
    - real-time, not pre-computable (e.g., telephony)
Internet Telephony

- real-time
- full duplex
- cannot be pre-computed, pre-compressed
- provide usual telephony services
  - call forwarding
  - call waiting
  - caller ID
  - PBX services
- interoperability with existing PSTN

Telephones

- what is a telephone?
  - any combination of hardware and software that performs these functions can be a telephone
  - activation control switch
  - audio capture device (microphone)
  - audio reproduction device (speaker)
  - activity alert device (ringer)
  - address selection device (keypad)
Conventional Phone Network

- End office
- Connecting trunk
- Interoffice trunk
- Subscriber loop
- Long-distance office
- Subscriber loop
- Digital PBX

Figure 9.2 Example Connection Over a Public Circuit-Switching Network

Voice over IP

- end points of calls:
  - computer (with mic and spkr)
Voice over IP

- end points of calls:
  - computer (with mic and spkr)
  - conventional telephone
- a ‘phone call’ can originate or end at either
- need to have Internet interoperate with PSTN
  - signal standards
  - addressing
    - phone numbers
    - IP addresses

The New Layout

- need to use telephony stds for PSTN work
- need to use IP stds for Internet work
Special Gateways

- **media gateway:**
  - audio data xlate between IP and PSTN
- **media gateway controller:**
  - ‘overall’ coordination between IP phones
  - mediates access to callee info in PSTN for IP side
- **signaling gateway:**
  - signaling operation translation (e.g., between SIP and SS7)
  - xlates call requests from either ‘side’ into corresponding representation for other ‘side’

The Standards

- VoIP standards proposed by
  - IETF
  - ITU-T
It Started Off Well…

- ITU-T and IETF agree on the following common ground:
  1. audio is digitized using PCM
     - 8 bits per sample
     - 8000 samples per second
     - 64 kbps
  2. use RTP to move data (Real-Time Transport Protocol) [RFC3550]
     - not a transport-layer protocol
  3. RTP message carried via standard UDP
     - further encapsulated as IP

Why UDP?

- UDP provides
  - out of order datagram delivery
  - duplicates, missing data
  - little or possibly no checking on data
Why UDP?

- UDP provides
  - out of order datagram delivery
  - duplicates, missing data
  - little or possibly no checking on data
- overhead of TCP too high for real-time needs
  - better to have a gap from dropped or error pkt than delay waiting for re-xmit
- introduce real-time clock values into datagrams
  - can eliminate duplicates
  - can reassemble in correct sequence

Time Sensitive (Multimedia) Data

- voice data is carried in UDP datagrams
- needs (at least) timestamps
  - may need other information
- ⇒ need for a protocol that can carry time sensitive data via UDP
- what other requirements do we have?
Time Sensitive (Multimedia) Data

- some additional requirements:
  - notification of missing packets
  - respond to congestion
  - control messages
    - e.g., so recipient can notify sender of congestion
  - frame boundary notification
  - strings for identification

RTP

- Real-time Transport Protocol (RTP) [RFC3550]
  - really a transport protocol?
- Real-time Transport Control Protocol (RTCP)
- supports wide range of ‘real-time’ data
  - audio, video
- defines, for each class of real-time data:
  - profile: guides understanding of header data
  - ≥ 1 format: how data after hdr is represented
RTP Message

```
+----------------+-----------------+------------------+
|V=2|P|X|  CC   |M|     PT      |       sequence number         |
+----------------+-----------------+------------------+
|                           timestamp                           |
+----------------+-----------------+------------------+
|           synchronization source (SSRC) identifier            |
|+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=+=|
|            contributing source (CSRC) identifiers             |
|                             ....                              |
+----------------+-----------------+------------------+
```

Padding [1]
eXtension header [1]
Payload Type [7]

```
version = 2
```

sequence num [16]: count-by-1 of packets
RTP Message

<table>
<thead>
<tr>
<th>V=2</th>
<th>P</th>
<th>X</th>
<th>CC</th>
<th>M</th>
<th>PT</th>
<th>sequence number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- timestamp [32]
- SSRC ident [32]
- CSRC ids [32 x n]

RTCP

- main functions:
  - app performance, network feedback
    - for adaptive rate applications
  - synchronize different media streams from same sender
    - e.g., audio and video may be sent separately
    - streams may have different clocks, different drift
  - represents identity of sender for display in UI
RTP: an example of ALF

- ALF: application layer framing
  - use to design new protocols for multimedia content
  - notion: only the application really best knows its requirements

Moving Datagrams Is The Easy Part

- once endpoints are known, moving datagrams from src to dest is ‘easy’
- **signaling**: work done to manage infrastructure:
  - set-up a call
  - tear-down a call
- current PSTN uses “Signaling System 7” (SS7)
- any IP elements in VoIP must interoperate with SS7 to use any PSTN elements
Standards, Again

- IETF proposes:
  - Session Initiation Protocol (SIP)
- ITU-T proposes:
  - H.323 group
- jointly have proposed:
  - Megaco, MGCP

SIP Terminology

- **user agent**: entity originating or receiving a call
  - user agent client: originates a call (outgoing)
  - user agent server: receives a call (incoming)
- **location server**: provides user-specific info, e.g.,
  - services user subscribes to
  - user preferences
- **proxy server**: handle routing, enforce policy
- **redirect server**: forwarding, toll-free connections
- **registrar server**: applies authenticated updates to db used by location server
The SIP Method(s)

- SIP defines *methods* (message types):

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INVITE</td>
<td>endpoint invited to join a session</td>
</tr>
<tr>
<td>ACK</td>
<td>acknowledge INVITE request</td>
</tr>
<tr>
<td>BYE</td>
<td>terminate session; end call</td>
</tr>
<tr>
<td>CANCEL</td>
<td>cancel pending request</td>
</tr>
<tr>
<td>REGISTRAR</td>
<td>URL where user can be reached</td>
</tr>
<tr>
<td>OPTIONS</td>
<td>query callee’s capabilities</td>
</tr>
</tbody>
</table>

Sample SIP Session
Who You Gonna Call?

- IP targets identified by IP addresses
- telephones ‘identified’ by telephone numbers
  - ITU-T E.164 telephone number standard
  - different formats in different places
- how do you identify callee
  - by phone number?
  - by IP address?

Mapping Phone Numbers

- ENUM protocol [RFC 2916]
  - reverse string holding phone number
  - treat each digit as segment of domain name
  - in special domain e164.arpa
  - e.g., phone number 1-800-555-1244 becomes: 4.4.2.1.5.5.5.0.0.8.1.e164.arpa
  - is 1:many mapping...DNS server replies with:
    - list of hosts with that phone number
    - protocol needed to reach each
Finding How to Reach Users

- TRIP: Telephone Routing over IP [RFC 3219]
  - location servers use to advertise known ‘routes’ to each other

H.323 Terminology

- terminal: provides IP telephone function
  - for voice
  - for video and other data as well
- gatekeeper: location and signaling functions + coordinates gateway access to PSTN
- gateway: interconnect IP to PSTN networks
- MCU: services including multipoint conferencing
H.323 Family

- H.323 is a collection of protocols that collectively perform all telephone operation;
- uses both TCP and UDP for transport;
- not ASCII-based; uses BER to represent ASN.1 binary messages;

Comer, Fig. 33.4 shows the family:

<table>
<thead>
<tr>
<th>Layer</th>
<th>Signaling</th>
<th>Registration</th>
<th>Audio</th>
<th>Video</th>
<th>Data</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>H.225.0-Q.931</td>
<td>H.225.9-RAS</td>
<td>G.711</td>
<td>N.261</td>
<td>H.235</td>
<td></td>
</tr>
<tr>
<td></td>
<td>H.250-Annex G</td>
<td>H.230</td>
<td>H.263</td>
<td>H.323</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>H.245</td>
<td>H.245</td>
<td>G.722</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>H.250</td>
<td>H.250</td>
<td>G.723</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>G.729</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>TCP, UDP</td>
<td>UDP</td>
<td>RTP, RTCP</td>
<td>TCP, UDP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>IP, RSVP, and IGMP</td>
<td>UDP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
International Packet Communications Consortium

- formerly ISC: International Softswitch Consortium
- industry-formed group
- define standards that can be implemented and put into service
- attempt to embrace all situations...

IPCC View

- **MGC–F**: keeps state info in endpoints, provides call logic and call control
  - **CA–F**: maintains call state (e.g., SIP, H.323, Q.931)
  - **IW–F**: signaling between different nets (e.g., SS7 → SIP)
  - **RF/AF**: Routing of calls/Accounting info re. calls
- **SG–F**: signaling IP network ↔ PSTN
- **AGS–F**: signaling IP network ↔ circuit-sw access net (e.g., ISDN)
- **MG–F**: digital audio xlate
IPCC View

- **AS–F**: application services (e.g., voicemail)
  - **SC–F**: invoked when AS–F controls a service
  - **MS–F**: handles media pkt stream for AS–F app
- **MG–F**: digital audio xlate
  - may also recognize DTMF tones and phone device events (like OH)

The World Wide Web

- a content–oriented Internet service
- ubiquity of clients makes content access and display nearly platform neutral
- grew from simple desire to cross–reference scientific articles at CERN
- has only existed since 1989
- Google currently lists 4,285,199,774 pages
  - recognized as a small percentage of total pages
WWW Client Server Model

- client, a.k.a. web browser
  - proprietary: Microsoft Internet Explorer
  - open source: Mozilla
  - text-only, e.g., lynx
- clients available for almost every platform known to mankind
- client:
  - renders documents it receives in a mark-up language, HTML
  - can initiate requests to servers for documents

Web Documents

- all delivered as HTML via HTTP from server to client
- server may offer:
  - static pages: content is generated once and delivered as fixed block of data
  - dynamic pages: content is generated once but may include instructions that generate content when the server goes to deliver the page
  - active pages: content is a program delivered to client who runs it to display output (often interacting with user)
WWW Client

- HTML documents
  - specify certain formatting instructions
  - leave others unspecified or unenforced
    - e.g., line length is determined by user’s window size, cannot be enforced *per se* in HTML document
- contain hypertext links (HREFs)
  - user model: mouse click on HREF causes current page to be abandoned in favour of newly requested page
  - links specified as Uniform Resource Locators

Uniform Resource Locators (URLs)

- see RFC 2396 (Aug 96)
- uniform addressing scheme for accessing resources from anywhere on Internet, using various protocols
- flexible, extensible, arcane-looking
- meant to identify static, persistent objects; not well suited to:
  - mobile
  - interactive
  - frequently varying resources
- gives specific location of the resource
  - so you have to know this in order to use it
  - resource must not ‘move’
URLs, cont’d:

- use 7-bit ASCII coding
  - excluding characters: { } | \ ^ ~ [ ] ` \\
  - reserved characters: ; / ? : @ = &

- related concept, uniform resource name (URN), identifies resource by name, independent of location
  - so can work where URLs fail
  - requires lookup to map URN to current URL
  - see RFC2141

URL Syntax

```
scheme  scheme specific part
        
one of: ftp  
       http  
gopher  
mailto  
news  
nntp  
telnet  
waist  
file prospero
```
URL Syntax, cont’d:

- common Internet scheme syntax:

```
scheme scheme specific part
```

```
http://<host_id>:port/pathname?stuff
```

- http scheme syntax (↑)

- relative URLs: resource location specified relative to a current URL (portability issues)
WWW Client/Server Interaction

HTTP Client 1 HTTP Server

HTTP Server 2 page files

Web Server Machine
WWW Client/Server Interaction

- HTTP Client
- HTTP Server
- Web Server Machine

Hypertext Markup Language (HTML)

- a language defined using Standard Generalized Markup Language (SGML)
- same base from which we get XML
- embed tags in text that describe how the text should be rendered when displayed
- browsers control many elements of rendering
  - e.g., window size, fonts
  - creator of document cannot enforce a layout
Sample HTML Tags and Their Effect

<TITLE>Don Quixote's Home Page</TITLE>
<I>This is in italics</I>
<TT>This it in typewriter font</TT>
<A HREF="http://www.w3.org">World Wide Web</A>
<br>
<IMG SRC="images/justborris.gif">

Hypertext Transfer Protocol (HTTP)

- see RFC2616 (June 99) for ‘current’ version 1.1
  - updated by RFC 2817
- primary protocol for accessing web pages
- simple, flexible, client-server transaction protocol
- 4 steps:
  - client establishes TCP connection to server
  - client issues request for a particular URL
  - server sends response containing
    - status code
    - any data associated with the URL
  - either client or server disconnects the TCP connection
HTTP, cont’d

- http is a **stateless** protocol, 1 transaction per TCP connection
  - but browser may open multiple connections for multi-part Web pages
- for HTTP 1.0 transaction types include:
  ```
  DELETE   HEAD   POST   UNLINK
  GET      LINK   PUT
  ```
- uses MIME to identify type of content for transactions

Statelessness: Cookies to the Rescue

- **cookie**: data record given to your web client by a server it has contacted
- thereafter: client contacting server sends cookies it previously received from that server
- client matches anything from complete URL to just domain
  - loose enough that server can set cookies for other servers to see
  - e.g., “.gov.au” lets all Australian government sites receive cookie (and so, e.g., track user activity)
### Cookies

- cookies contain:
  - name: (required) text field
    - not ';', ' ', or whitespace
  - expires: do not store or give out cookie after this date
  - path: prefix matched against URL (match all if absent)
    - often is '/'
  - domain: tail–match on site to which being sent
  - secure: if yes, send only if connection is secure
  - information: data server wants to see from client
    - usually cryptic, e.g., FCGKEJAEKNCBOKNLKJELAG

### Sample Cookie

```
>virginiadot.org
TRUE
/
FALSE
205122650
SITESERVER
ID=7f28bdcc9f7d2cd54ee168887db739a8.
```
HTTP Requests

- **GET URL**
  - requests object associated with URL
  - doesn’t have scheme name, host or port ID; why?
- **HEAD URL**
  - request meta-information for obj associated with URL
- **POST**
  - transfers data from web client to web server
- **PUT**
  - transfers data to server to be stored and associated with specific URL
- **DELETE**
  - asks server to remove specified object

HTTP Example

```
somehost% telnet cs.gmu.edu 80
Trying 129.174.87.2...
Connected to cs.gmu.edu.
Escape character is '^]'.
GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 22 Apr 2002 17:46:21 GMT
Server: Apache/1.3.12 (Unix)
Last-Modified: Wed, 30 Jan 2002 17:29:56 GMT
ETag: "470f-33e8-3c582d94"
Accept-Ranges: bytes
Content-Length: 13288
Connection: close
Content-Type: text/html

<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<link rel="stylesheet" href="textstyle.css" type="text/css">
</head>
<body>
<title>Computer Science Dept.</title>
...</body>
</html>
```
Server Takes Note

- every access to server creates log entry with:
  - date and time of access
  - IP address of request origin
  - the actual request string
  - client information (name, version)
  - client OS information (name, version)

- e.g., (all one long line):
  127.0.0.1 - - [01/Apr/2004:08:19:52 -0500] "GET /~charles HTTP/1.1" 301 339 "-" "Mozilla/5.0 (X11; U; Linux i686; en-US; rv:1.5a) Gecko/20030728 Mozilla Firebird/0.6.1"
  192.168.0.18 - - [03/Apr/2004:23:15:21 -0500] "GET /~charles HTTP/1.1" 301 339 "-" "Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0; (R1 1.3))"

Common Gateway Interface (CGI)

- some web sites are generated dynamically in response to specific client queries
- CGI protocol defines interface HTTP servers and applications use to exchange data
- such applications may be:
  - compiled executable
  - sequence of (shell) commands ("cgi–script") added to filesystem space under HTTP server 'control'
- common language choices for scripts include:
  - perl
  - Tcl
  - python
**CGI Architecture**

1. client connects to HTTP server, issues request

2. server sets environment variables from request, runs CGI script
CGI Architecture

1. client connects to HTTP server, issues request
2. server sets environment variables from request, runs CGI script
3. upon exit, output from CGI script sent to server
4. server sends reply to client
Processing Requests: Environment Variables

GET url HTTP 1.0 CR LF CR
/courses/summer/csa3?sect=1&dy=tj&tm=am

REQUEST METHOD
CONTENT_LENGTH
QUERY_STRING
CONTENT_TYPE set to reflect url-encoded text
REMOTE_HOST hostname of requesting system
REMOTE_ADDR IP address of requesting system
SERVER_SOFTWARE name/version of responding server software
SERVER_NAME server name or IP addr as appears on self-ref. URLS
GATEWAY_INTERFACE CGI/rev with which is compliant

Inside a CGI Script

What is REQUEST_METHOD?

GET

POST or PUT

Read CONTENT_LENGTH bytes from STDIN

Perform Processing
Format Output as http Header, followed by a blank line and then an http Entity-Body
Write the lines out to STDOUT

OUTPUT BYTES
Dynamic Pages

- some parts of web page static, others dynamic
  - e.g., company name static, date dynamic
- use specially marked-up html pages that include tags to insert contents when delivering page
  - e.g., <H3>OZME Manufacturing <getdate></H3>
- can insert almost anything, including complete db searches
- some commercial packages that do this:
  - ColdFusion (.cfm)
  - Microsoft Active Server Pages (.asp)
  - PHP (.php)
  - Java Server Page (.jsp)

Java Applets

- small programs that are downloaded as part of a web page, run on client system (*platform independent*)
- requires client system to provide necessary interpreter and run-time environment
- a link in a page may refer to an applet
  - just another file to the HTTP server
  - server replies by returning the applet
  - client system runs the applet
- security restrictions on Java applets
  - cannot perform file operations
  - cannot open other network connections
Java Applet Architecture

- Java program has been compiled to byte code and left on server
  1. client connects to HTTP server requests resource

```
HTTP Client
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Interpreter and Run-time System</td>
</tr>
</tbody>
</table>

HTTP Server Machine
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. HTTP Server</td>
</tr>
<tr>
<td>2. Java Byte Codes</td>
</tr>
</tbody>
</table>
```

Java Applet Architecture

- Java program has been compiled to byte code and left on server
  1. client connects to HTTP server requests resource
  2. server sends requested resource to client

```
HTTP Client
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Interpreter and Run-time System</td>
</tr>
</tbody>
</table>

HTTP Server Machine
<table>
<thead>
<tr>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. HTTP Server</td>
</tr>
<tr>
<td>2. Java Byte Codes</td>
</tr>
</tbody>
</table>
```
Java Applet Architecture

- Java program has been compiled to byte code and left on server
  1. client connects to HTTP server requests resource
  2. server sends requested resource to client
  3. client loads byte code and invokes local Java interpreter

Java Applet Lifecycle

- each phase in lifecycle corresponds to a method invocation in an applet
- once loaded, applet stays in client page cache until page containing applet is discarded
- return to cached page causes applet to restart
Applet References and Java Tools

- `javac` – Java compiler creates byte codes image from a Java program
- `java` – Java interpreter executes byte code images
- `jdb` – Java debugger
- `appletviewer` – Java applet viewer

- what applets look like in a web page:

```html
<applet code=ImageLoopItem width=80 height=90>
  <param name=nimgs value=10>
  <param name=img value=duke>
  <param name=pause value=1000>
</applet>
```

JavaScript

- scripting language embedded in html sent to client
  - `<script language="javascript"> ... </script>`
  - defines functions for actions during display of page
    - e.g., if mouse enters a region, pop-up a dialogue box
    - e.g., update part of a page as result of changes to another part
- not really related to programming language Java
- lighter-weight to process than Java applet
  - so user sees ‘instantaneous’ response
TCP: Getting a Web Page

14:50:12.452265 babylon5.int.lexiconix.com:57483 > jiju.gmu.edu:http: S
2018466819:2018466819(0) win 5840 <mss 1460,sackOK,timestamp 204923450
0,nop,wscale 0> (DF)
14:50:12.487105 jiju.gmu.edu:http > babylon5.int.lexiconix.com:57483: S
310748728:310748728(0) ack 2018466820 win 65160 <nop,nop,timestamp
197332264 204923450,nop,wscale 0,nop,nop,sackOK,mss 1460> (DF)
14:50:12.540360 babylon5.int.lexiconix.com:57483 > jiju.gmu.edu:http: P
1:439(438) ack 1 win 5840 <nop,nop,timestamp 204923459 197332264> (DF)
0x0000  4500 01ea be3b 4000 4006 3656 c0a8 00f2        E....;@.@.6V....
0x0010  81ae 0134 e08b 0050 784f 5c04 b96a 4239        ...4...Pox\.jB9
0x0020  8018 16d0 1bde 0000 0101 080a 0c36 e243        .............6.C
0x0030  0bc3 0228 4745 5420 00f2 4854 5450 312e        (GET./.HTTP/1
0x0040  0e204 00da 486f 7374 3a20 7777 772e 676d        u:.www.gm
0x0050  752e 6564 750d 0a55 7365 722d 4167 6e6e        u.edu..User-Agen
0x0060  743a 204d 6f7a 696c 6c61 2f35 2e30 2028        t:.Mozilla/5.0.
0x0070  5831 313b 2055 3b20 4c69 6e75 7820 6936        X11;.U;.Linux.i6
0x0080  3836 3b20 656e 2d55 533b 2072 6669 72652        86;.en-US;.rv:1.
0x0090  3561 2920 4765 636f2f32303037322e5b7b6943        5a).Gecko/200307
0x00a0  3233 204d 67672046 7372 6562 6573206469722        32.Mozilla.Fireb
0x00b0  6765 7320 6d61696c652f322e30372c6170706c69636c        ication/xml,appl
0x00c0  743a 2074 6570 7465 7374 3a20 73797374656d6967682        t:.text/xml,appl
0x00d0  666f20 666f 6669 6365737320 6c696e656420 616e6420 6e616d652        ion/xml,application/vnd.ms-excel
0x00e0  73657373696f6e64696e6775726e7320 666f7220 616e6420 6e616d652        ion+xml

© Charles M Snow 48