

Services and HTTP

Lecture 4

cs193i – Internet Technologies
Summer 2004
Stanford University

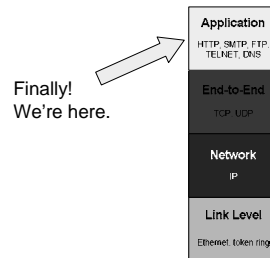
Administrative Stuff

- Lab #2 due July 14
- HW #1 due July 12
- Silas' review Perl review session
 - 7/13, 2:15-3:05
 - Skilling 193

Protocol Stack



Protocol Stack



End-to-End Argument

- Move functionality from lower layers to application specific layers
- Why?
 - Functionality may require application level info
 - Everyone pays for it when it's in lower layer
- BUT you may add functionality at lower levels for performance

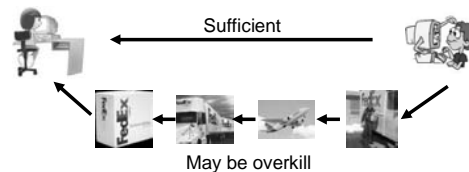


Worse performance/
Programmer Hassle

Redundant/
Relatively costly
Worse performance/
Programmer Hassle
Higher Performance/
Easy for Programmers Above

End-to-End Example

- Real Life Example: Mail package confirmation
 - Messenger to Messenger (Low Level)
 - Each scans package and confirms receipt
 - Sender to Receiver (High Level)
 - Receiver calls sender, "I got it"



End-to-End

- Acknowledge Receipt of Data (ACKs)
- Application/Service level, App <=> App
 - (e.g. FTP Client to Server)
 - harder for programmers
- TCP level, Computer <=> Computer
- Routing level, Hop <=> Hop
 - 10 router hops means 10x ACKs!

Service

- Mechanism for computers to interact (application layer)
- Term refers to overall solution
- Usually associated with IP port number
- Differs from protocol which describes the details of how interaction works
 - Ex) HTTP service builds on TCP/IP
- RFC used to define service standard

Applications

- Traditional PC applications
 - Everything done locally
 - Fast but sharing difficult
 - Word, Excel
- Client/server applications
 - Client local and responsive
 - Client provides interface
 - Server centralizes resources
 - Server performs some work



Thin vs. Thick Clients

- Web Apps are “Thin”
- Server does processing
- Client does presentation
 - + Simple! (Browser)
 - Limited GUI (HTML)



Thin vs. Thick Clients

- Software is “Thick” (AIM)
- Client does processing and presentation
 - + GUI not limited by HTML
 - + Snappy
 - (fewer Latency Problems)
 - People need to download & install client



Servers

- Hardware server
 - Computer on Internet, always running
- Software server (aka daemon)
 - Program running on server
 - Listening on port
 - Receives requests, processes them, makes outgoing calls
 - Daemon examples: sshd, lpd, inetd, httpd

Contact a Daemon Using Telnet

```
saga10:-> telnet www.google.com 80
Trying 64.233.167.104...
Connected to www.google.akadns.net (64.233.167.104).
Escape character is '^]'.
GET /index.html HTTP/1.0

HTTP/1.0 200 OK
Cache-Control: private
Content-Type: text/html
Set-Cookie:
PREF=ID=72459b575402fb39:TM=1089165164:LM=1089165164:S=U8m_gb0
hxi2SV
KLP; expires=Sun, 17-Jan-2038 19:14:07 GMT; path=/;
domain=.google.com
Server: GWS/2.1
Content-Length: 2096
Date: Wed, 07 Jul 2004 01:52:44 GMT
Connection: Keep-Alive

<html><head> ...
```

Example Standardized Services

DNS
FTP
SCP
Ping
Finger
Telnet, SSH
SMTP
POP (your HW#1)
IMAP
HTTP (the next several lectures)

Domain Name Service

- TCP/IP uses IP Addresses (171.64.123.12)
- DNS allows us to use URLs to refer to IP addresses
(e.g. www.yahoo.com)
- It's just a service built on top of TCP/IP!!!

DNS

- Benefits of indirection
 - Can move machine to new IP
(just update the DNS entry)
- Multiple DNS names map to single IP
 - www.foo.com,
movies.foo.com
- Multiple servers can service same domain name

```
saga10:-> nslookup www.google.com
Server: cici.Stanford.EDU
Address: 171.64.7.121

Non-authoritative answer:
Name: www.google.akadns.net
Addresses: 64.233.167.99,
64.233.167.104
Aliases: www.google.com
```

Sending an Email

- SMTP – Simple Mail Transfer Protocol
- Your email client talks to an SMTP server
 - SMTP server routes the mail to other servers... until it reaches destination
 - Destination server program (aka daemon)
 - Accepts mail, puts in mailbox of the user
 - If user doesn't exist, then bounce!

Receiving an Email

- Elm/Pine
 - Connect to account via telnet
 - All mail remains on server
- POP – Post Office Protocol
 - Copies mail from server to local PC
- IMAP – Internet Mail Access Protocol
 - Mail remains on server
 - GUI presents interface for interacting with server

Thick Email Client



Thin Email Client



Basic Security

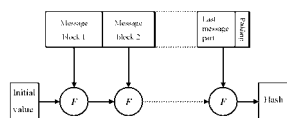
- Authentication (Prove who you are)
- Q: What are the three ways?
 - Something You Know **password: foobar**
 - Something You Are **retina**
 - Something You Have **car keys**

Traditional Authentication

- Shared Secret
 - Server & client both know password
- Password Demand (Server asks client for it)
 - Client presents it
 - Server checks against its own password DB

One-way Hash Function

- Combine, or “hash” bits of a string together to produce a “hash value”
- Function of the input
- Not invertible
- Hash’s should be kind of unique
 - Strings A & B should not have same hash



Sample Hash Functions

- Bad Hash: Add Up Byte Values
 - $FOOBAR = 70 + 79 + 79 + 66 + 65 + 82 = 441$
- OK Hash: Linear Hash
 - Mathematical Function of Bits %
 - `SOME_BIG_NUMBER`
- Good Hash: MD5 (128 bit hash values)
- Better: SHA-1 (160 bit values)

Replay Attack

- Snooper captures your message including your hashed password
- Snooper can now resend that message to server to pretend to be you!

Challenge / Response

- Server sends R (random number) as a challenge to client
- Client computes Hash(R + Password), sends to server
- Server verifies

Replay attacks are prevented!

Problem: People choose Bad Passwords!

- Words in the Dictionary
 - Dictionary Attack
- Short & Simple Passwords
 - Brute Force
 - 3 Alphabet letters => $26^3 = 17576$
 - 9 Alphabet letters => $26^9 = 5.4E12$
 - 9 Alphanumeric => $36^9 = 1.0E14$

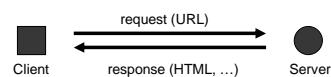
Five Minute Break

HTTP and HTML

- Hypertext Transfer Protocol (HTTP)
- Tim Berners-Lee, 1991
- Hypertext Markup Language
 - For creating web pages

Client and Server

- User uses HTTP client (Web Browser)
- It has a URL (e.g. <http://www.yahoo.com/>)
- Makes a request to the server
- Server sends back data (the response)
- User clicks on the client side...



HTTP Client (Browser)

- NCSA Mosaic (M. Andreessen)
- Netscape Navigator (M. Andreessen)
- Microsoft Internet Explorer
- Browser Wars of the 1990's
- Mozilla (Netscape Open Sourced)
- Now Mozilla Firefox
- Apple Safari (from Konqueror)
- Others (Opera, Lynx)

Universal Resource Location (URL)

`http://www.stanford.edu:80/class/cs193i/schedule.html`



Protocol (Scheme)

Universal Resource Location (URL)

`http://www.stanford.edu:80/class/cs193i/schedule.html`



Host Name

Universal Resource Location (URL)

`http://www.stanford.edu:80/class/cs193i/schedule.html`



Port

Universal Resource Location (URL)

`http://www.stanford.edu:80/class/cs193i/schedule.html`



Path

Request

- Just a string of ASCII text
- `GET /food/index.html HTTP/1.0\r\n\r\n`

HTTP Server

- Listens on port 80 (usually)
- Handles HTTP requests
- Sends back responses
- Document root is a directory in the file system
- Server maps path to file system file

URL Path = File System Path

- URL Path “/” maps to Document Root
- Let's say Document Root is C:\htdocs\
 - / => C:\htdocs\
 - /images/ => C:\htdocs\images\
 - /a/X.html => C:\htdocs\a\X.html

Response Example

HTTP Header	HTTP/1.1 200 OK Date: Fri, 16 Apr 2004 18:48:13 GMT Server: Apache/1.3.29 (Darwin) Last-Modified: Fri, 16 Apr 2004 10:15:59 GMT ETag: "58db37-89-407fb25f" Accept-Ranges: bytes Content-Length: 137 Connection: close Content-Type: text/html
Blank line	
Data	<html> <body> <p>Welcome</p> </body> </html>

Example Request / Response

- Client requests
http://solaria.stanford.edu/food/index.html
- Client sends
GET /food/index.html HTTP/1.0\r\n\r\n
- Server sees request with path /food/index.html
- Server maps onto Document Root
G:/webroot + /food/index.html
- Server sends back file over HTTP (e.g. HTML file)

HTTP 1.0 is Stateless

- Each request/response pair uses its own connection; doesn't know about other pairs
- "One-Shot"
 - Server Fulfills Request, and closes connection
 - + Simple
 - Hard to design pages that are "logically connected" (e.g. Amazon checkout)

Request

- Client sends a GET request
- GET path HTTP/1.0\r\n\r\n
- Note the two \r\n

What is the URL path?

- `http://foo.com:8080/a/b/bar.html?hello.there#binky`
- query begins with ?
 - `hello.there`
- fragment begins with #
 - `binky`
- So, path is between host and query/fragment
 - `/a/b/bar.html`
- But Request-Line includes Query

Query

- Starts with ?
- May contain name/value pairs
- May contain & to list multiple pairs
- `http://bob.com/subscribe.html?name=ron&uid=1234`

Fragment

- Used by client side to scroll to named anchors
- `...`
- `http://foo.com/b.html#Chapter1`

Request String

- The path & query part of the URL
- NOT the fragment part
- `http://foo.com/dir/b.html?info=extra&hello`
 - `/dir/b.html?info=extra&hello` is the Request String
 - GET request-string `HTTP/1.0\r\n\r\n`

Two Main Request Types

- GET
- POST
- PUT & DELETE are rarely used

Response

- Header
- `<Blank Line>`
- Document Data
(e.g. HTML, GIF, JPEG, SWF...)

HTTP Response Header

- Header Describes the Document
- VERSION / CODE / REASON
 - HTTP/1.0 200 OK
 - HTTP/1.1 404 Not Found
- Content-Length: size-in-bytes

HTTP Response Header

- Content-Type: MIME-type
 - text/html
 - text/plain
 - image/jpeg
 - image/gif

```
elaine30-> telnet cslibrary.stanford.edu 80
Trying 171.64.64.168...
Connected to cslibrary.Stanford.EDU (171.64.64.168).
Escape character is '^J'.
GET /test.html HTTP/1.0

HTTP/1.1 200 OK
Date: Wed, 07 Jul 2004 17:59:42 GMT
Server: Apache/1.3.26 (Darwin)
Last-Modified: Thu, 25 Apr 2002 00:50:34 GMT
ETag: "115b1-1cb-3cc752da"
Accept-Ranges: bytes
Content-Length: 459
Connection: close
Content-Type: text/html

<!doctype html public "-//w3c/dtd html 4.0 transitional//en">
<html>
<head>
<title>Test</title>
<meta http-equiv="nick-mode" content="high">
</head>
<body bgcolor="#FFFFFF">

<h1>
Test</h1>
<p>Just a little test doc.

</body>
</html>
```

HTML Characteristics

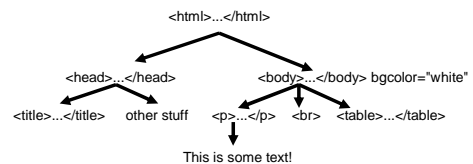
- Just a Text File!
 - + Portable
 - + Human Readable/Writable
- Defines the Structure (not Appearance) of the Document
 - Client (Browser) defines the appearance
 - + Portable
 - + Pours into Browser (PDAs, Bigger/Smaller)

Document Structure

```
<html>
<head><title>My First Web Page</title>
</head>
<body bgcolor="white">
<p>A Paragraph of Text.</p>
</body>
</html>
```

Nested Tags

- Like a tree, each element is contained inside a parent element
- Each element may have any number of attributes



Basic Tags

- `<hr>` horizontal rule
- `
` new line
- `...` bold
- `<i>...</i>` italicize text in between

Advanced Tags

- `First Item`
- `Second Item`
- Also, `...`
- ``

Image File Types

- JPEG
- GIF
- PNG
- SVG

Tables

- `<table>...</table>`
- `<tr>...</tr>` for each row
- `<td>...</td>` for each element in a row

Comments

- `<!-- This is a comment -->`
- `<!--`
This paragraph,
is also a
comment...
`-->`

Special HTML

- `<` → `<`
- `>` → `>`
- `&` → `&`
- ` ` → space