CS 372 WEEK 3 NOTES

CS 372 Lecture #10 The Application Layer

* The application layer:
  + Example Application
    - Hypertext Transfer Protocol (HTTP)
* Web and HTTP (RFC 2616)
  + Web page consists of a basic HTML file which includes several referenced objects
    - Object can be HTML file, JPEG image, Java applet, audio file,….
  + Each object is addressable by a Uniform Resource Locator (URL)
  + Example URL
    - [www.someschool.edu/somedept/pic.gif](http://www.someschool.edu/somedept/pic.gif)
    - host name/ path name in host directory structure
* HTTP overview
  + Web’s application layer protocol
  + client/server model
    - client: browser that requests, receives “displays” web objects
    - server: Web Server sends objects in response to requests
* HTTP overview (continued)
  + uses TCP:
    - client initiates TCP connection (creates socket) to server, port 80
    - server accepts TCP connection from client
    - HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web Server (HTTP server)
    - TCP connection closed
  + HTTP is “stateless”
    - server maintains no information about past client requests
  + Protocols that maintain “state” are complex!
    - past history (state) must be maintained
    - if server/client crashes, their views of “state” may be inconsistent, must be reconciled
* HTTP connections
  + Non-persistent HTTP
    - at most one object is sent over a TCP connection
    - downloading multiple objects requires multiple connections
  + Persistent HTTP
    - multiple objects can be sent over single TCP connection between client and server
* Non-persistent HTTP
  + suppose user enters URL
    - [www.someSchool.edu/someDepratment/home.index](http://www.someSchool.edu/someDepratment/home.index) (contains text, references to 10 jpeg pages)
  + 1a. HTTP client initiates TCP connection to HTTP server (process) on port 80 at [www.someSchool.edu](http://www.someSchool.edu)
  + 1b. HTTP server at host [www.someSchool.edu](http://www.someSchool.edu) waiting from TCP connection at port 80. “accepts” connection, notifying client
  + 2. HTTP client send HTTP request message (containing URL) into TCP connection socket. Message indicates that client wants object someDepartment/home.index
  + 3. HTTP server receives request message, forms response message containing requested object, and sends message into its socket and closes connection.
  + 4. HTTP client receives response message containing HTML file, displays HTML. Parsing html file, finds 10 referenced jpeg objects
  + 5. Steps 1-4 repeated for each of 10 jpeg objects
* Non-Persistent HTTP: Response time
  + Definition of RTT (round-trip time)
    - time for a small packet to travel to server from client and back
  + Response time
    - one RTT to initiate TCP connections
    - one RTT for HTTP request and first few bytes of HTTP response to return
    - file transmission time
  + total = 2RTT + transmit time (non-persistent HTTP)
* Persistent HTTP
  + Non-persistent HTTP issues:
    - requires 2 RTTs per object
    - operating system overhead for each TCP connection
    - browsers often open parallel TCP connections to fetch referenced objects
  + Persistent HTTP
    - server leaves connection open after sending response
    - subsequent HTTP messages between same client/server sent over open connection
    - client sends requests as soon as it encounters a referenced object
    - as little as one RTT for all referenced objects
* HTTP request message
  + two types of HTTP messages: request, response
  + HTTP request message:
    - ASCII (human-readable format)
    - request line
    - header lines
    - extra carriage return line at start indicates end of header lines
  + The web browser is an application that includes and implements the client side of HTTP
* HTTP request message: general format
  + sp = space
  + cr/lf = carriage return/line feed (enter key)
  + request line
    - method sp URL sp version cr lf
  + header lines
    - header field name value cr lf
    - header field name value cr lf
    - cr lf
  + entity body (data)
* Uploading form input
  + Web page often includes form input
  + URL method
    - uses GET method
    - input is uploaded in UTL field of request line
    - Example:
      * ww.somesite/animalsearch?monkeys&banana
  + POST method
    - input is unloaded to server in entity body
* HTTP response message
  + status line
    - protocol
    - status code
    - status phrase
  + header lines
  + extra cr/lf
  + data, e.g., requested HTML file
* HTTP response status codes
  + In first line of server to client response message
  + A few sample codes:
    - 200 ok
      * request succeeded, requested object later in this message
    - 301 moved permanently
      * requested object moved, new location specified later in this message (location)
    - 400 bad request
      * request message not understood by server
    - 404 not found
      * requested document not found on this server
    - 505 HTTP version not supported
* Try out HTTP (client side) for yourself
  + 1. Telnet to your favorite web server:
    - telnet web.engr.oregonstate.edu 80
      * opens TCP connection to port 80
      * (default HTTP server port) at web.engr.oregonstate.edu
      * Anything you type will be sent to port 80 at web.engr.oregonstate.edu and the connection will close (default behavior)
  + 2. Type in a GET HTTP request:
    - GET /~paulson/HTTP/1.1
    - Host: engr.oregonstate.edu
      * By typing this in (hit carriage return twice) you send this minimal (but complete) GET request to HTTP server
  + 3. Look at response message sent by HTTP server (or use Wireshark to look at captured HTTP request/response)
* Summary LECTURE 10
  + Definitions
    - URL
    - RTT
    - response time
  + HTTP
    - Non-persistent, persistent
    - Request, response
    - Form input

CS372 LECTURE #11 THE APPLICATION LAYER

* The Application layer
  + more HTTP
  + Cookies
  + Caching
* Client-server state: cookies
  + Many major web sites use cookies
    - Four components
      * cookie header line of HTTP response message
      * cookie header line in HTTP request message
      * cookie file kept on user’s host, managed by user’s browser
      * back end database at web site
    - Example
      * User visits specific e-commerce site for first time
      * when initial HTTP requests arrives at site, site creates
        + unique ID
        + entry in backend database for ID
* Cookies: keeping “state” --diagram
  + client
  + server
* Cookies (cont.)
  + What cookies can provide
    - authorization
    - shopping carts
    - recommendations
    - user session state (Web e-mail)
  + Cookies and privacy
    - cookies permit sites to learn a lot about you
    - you may be giving your name and email to sites
* Web caches (proxy server)
  + Goal: satisfy client request without involving origin server
    - User’s browser sends all HTTP requests to cache
      * if object in cache: cache returns object
      * else cache requests object from origin server, then returns object to client
* More about web caching
  + - Cache acts as both client and server
    - Typically cache is installed by ISP (university, company, residential ISP)
    - Cached objects have “expiration” date/time
  + Why web caching?
    - reduce response time for client request
    - reduce traffic on an institution’s access link
    - enables “poor” providers to effectively deliver content
* Example (no caching)
  + Assumptions
    - average object size = 100K bits
    - average request rate from institution’s browsers to origin servers = 15 requests per second
    - delay from institutional router to any origin server and back to router = 2 seconds
  + Consequences
    - utilization on LAN = 15%
    - utilization on access link = -100%
  + total average delay = internet delay + access delay + LAN delay = 2 seconds + minute + milliseconds
* Example (caching)
  + Same assumptions
    - but with caching
    - suppose cache hit rate is .4
      * i.e. only 60% of requests go to origin servers
  + Consequences
    - 40% of requests will be satisfied almost immediately
    - utilization of access link is reduced to 60% resulting in negligible delays (say 10 ms) because of no congestion
  + total average delay = internet delay + access delay + LAN delay = .6 \* 2.0 seconds + .4 \*10ms < 1.4 seconds
* Conditional GET
  + fixes problem with origin server and proxy server delay
  + Goal: don’t send object if cache has up to date cached version
  + cache: specify date of cached copy in HTTP request
    - if-modified-since<date>
  + server: response contains no object if cached copy is up to date
    - HTTP/1.1 304 Not modified
  + cache/server diagram
* Summary LECTURE 11
  + Definitions
    - cookie
    - caching
  + HTTP
    - conditional GET

LECTURE 12 APPLICATION LAYER

* The application layer
  + File transfer protocol
  + email protocols
* FTP the file transfer protocol (RFC 959)
  + Diagram
    - user at host 🡨🡪 FTP user interface 🡨🡪local file system 🡨🡪 FTP client 🡨🡪FTP server 🡨🡪 remote file system
  + transfer file to/from remote host
  + client/server model
    - client: that initiates transfer (to or from remote)
    - server: remote host
* FTP: separate control, data connections
  + FTP client contacts FTP server to open TCP control connection
  + Client browses remote directory by sending commands over the TCP control connection
  + When server receives file transfer command, server opens 2nd TCP data connection (for file) to client
  + After transferring one file, server closes TCP data connection. TCP control stays on
  + FTP Reserved Ports
    - Data Transfer: 20
    - Control : 21
  + FTP server maintains “state”:
    - current directory, earlier authentication
    - limit on concurrent connections
* FTP commands, responses
  + A few sample commands
    - sent as ASCII text over control channel:
      * ls
        + lists files in remote directory
      * get filename
        + retrieves file from remote site to current local directory
      * put filename
        + stores file onto remote host in current dir
      * cd directory
        + changes remote dir
      * help
        + lists all ftp commands
  + A few sample return codes
    - status code and phrase
      * 200 command successful
      * 331 Username OK, password required
      * 425 Can’t open data connection
      * 452 Error writing file
      * 550 Failed to open file
* Electronic Mail
  + Three major components
    - User Agent
      * email client: compose, read, edit, send
        + e.g., Eudora, Outlook, elm, Mozilla Thunderbird
      * Outgoing, incoming messages stored on server
    - Mail Server
      * mailbox contains incoming messages for user
      * message queue of outgoing mail messages
    - SMTP [RFC 2821]
      * Simple Mail Transfer Protocol
      * defines message transfer rules and formats b/w mail servers
        + client: sending mail
        + server: receiving mail
* Electronic Mail: SMTP [RFC 2821]
  + Application layer protocol uses TCP
    - Client server application for sending mail
    - reliable transfer
    - default port is 25
  + Command/response interaction
    - commands: plain text
    - response: status code and phrase
  + Messages must be in 7-bit ASCII
* SAMPLE SMTP INTERACTION FOR THE LAB?
* Scenario: Alice sends message to Bob
  + Alice uses her user agent to compose message and sent to bob@someSchool.edu
  + Alice’s user agent send’s message to her mail server, message placed in message queue
  + Client side of SMTP opens TCP connection with Bob’s mail server
  + SMTP client sends Alice’s message over the TCP connection
  + Bob’s mail server places the message in Bob’s mailbox
  + Bob uses his user agent to read message
* Mail message format
  + SMTP: protocol for exchanging email messages
  + RFC 822: standard for text message format:
    - header lines (entered by user, formatted by local user agent)
      * To:
      * From:
      * Subject:
    - body
      * the “message”, ASCII characters only
* Message format: Multimedia extensions:
  + MIME: Multipurpose Internet Mail Extension, RFC 2045, 2056
    - Allows SMTP to handle foreign characters and images
    - additional lines in message header declare MIME content type
    - Structure
      * MIME version
      * method used to encode data
      * multimedia data type, subtype, parameter declaration
      * encoded data
* Mail Access Protocols
  + Q: why do we need mail servers?
    - why can’t Alice’s user agent push the message to Bob’s user agent without going via mail servers?
  + A: Potential Problems
    - first attempt fails (who tries again?)
    - Bob’s user agent is off (who stores the incoming messages for him?)
* Mail Access Protocols
  + Now because messages are to be stored at the server, Bob’s user agent must be able to retrieve them
  + Q: Why can’t Bob use SMTP to retrieve messages?
  + A: SMTP is a “push” protocol. Hence we need “pull” protocols: called “mail access protocols”
    - POP3
    - IMAP
    - HTTP (web mail)
* Mail Access Protocols
  + POP: Post Office Protocol [RFC 1939]
    - authorization (agent 🡨🡪 server) and download
  + IMAP: Internet Mail Access Protocol [RFC 1730]
    - Manipulation of stored messages on server, more complex
  + HTTP: gmail, Hotmail, Yahoo! Mail, etc. (web mail)
* POP3 and IMAP
  + POP3
    - uses download and delete mode
    - Bob cannot re-read email if he changes the client
    - Download and Keep copies of messages on different clients
    - POP3 is stateless across sessions
  + IMAP
    - Keep all messages in one place: the server
    - Allows user to organize messages in folders
    - IMAP keeps user state across sessions
* Summary Lecture #12
  + FTP
    - control connection, data connections
  + SMTP
    - MIME
  + Mail Access protocols
    - POP3
    - IMAP
    - HTTP
  + Definitions
    - User agent
    - push protocol
    - pull protocol

CS 372 LECTURE 13 THE APPLICATION LAYER

* The Application Layer
  + Domain Names Services (DNS)
* Addresses in an internet
  + Every host that is connected to any network has a unique hardware address
    - 12 hexadecimal digits, assigned by the manufacturer of the network interface device
      * e.g, 90e612a73d
        + each position has 16 possible alphanumeric digits so there are
    - used at the physical layer
    - not practical for internet addressing
  + IANA assigns internet protocol (IP) network addresses to service providers, etc, to be managed hierarchically
    - e.g., 128.192.0.0 (IP version 4)
  + Service providers assign IP addresses to individual subscribers and bind these IP addresses to subscriber hardware addresses
    - e.g., 128.192.35.172 IPv4 dotted-decimal notation
  + Much more about IP addressing later
* The need for naming
  + All applications use IP addresses through the TCP/IP protocol software
  + Numeric addresses are easy for computers to manage
  + but difficult for humans to remember:
    - e.g., 128.192.35.172 (dotted decimal)
    - binary form is 10000000110000000010001110101100
  + The computer needs a binary address
  + Humans “need” mnemonics
  + Domain Name System (DNS) provides translation between symbolic names and IP addresses
* Structure of DNS names
  + each name consists of a sequence of alphanumeric components separated by periods
  + examples
    - Comcast.com
    - [www.oregonstate.edu](http://www.oregonstate.edu)
    - [www.cnn.com](http://www.cnn.com)
  + Note: There is not a correspondence between the DNS name’s individual components and the fields of an IP address in dotted-decimal notation
* Structure of DNS names
  + names are hierarchical with most significant component on the right
    - Top-Level Domain (TLD)
  + Second from right is the domain name within the TLD
    - approved by a global authority
* Structure of DNS names
  + Other names may be added by the organization that owns the name
    - hierarchical structure
  + Left-most component is computer name
  + NOTE: www does not necessarily imply web services
    - It’s just a computer name in a domain
    - usually that is what’s used however
* Structure of DNS names
  + Organizations apply for names in a TLD. E.G.:
    - oregonstate.edu
    - mozilla.com
  + Organizations determine own internal structure. E.G.:
    - eecs.oregonstate.edu
    - classes.eecs.oregonstate.edu
    - [www.mozilla.com](http://www.mozilla.com)
    - en-US.www.mozilla.com
* IP/DNS authority
  + IP addresses and “root zone” TLDs coordinated by IANA (internet assigned numbers authority)
    - <http://www.iana.org>
  + Information database (whois), dispute resolution, etc., managed by InterNIC (Internet Network Information Center)
    - <http://www.internic.net>
  + High-level management of Internet names and addresses handled by ICANN (Internet Corporation for Assigned Names and Numbers)
    - [www.icann.org](http://www.icann.org)
  + .com TLD managed by network soutions
    - [www.networksolutions.com](http://www.networksolutions.com)
  + .edu TLD managed by Educause
    - <http://www.educause.edu>
  + Traditional top-level domains (TLD)
* New gTLD (generic TLD)
  + In 2012, ICANN started taking applications for new TLD names
    - Application fee US$185,000.00
      * additional fees may apply
    - Approval process 9-20 months
    - 1930 applications submitted
      * mostly company names (e.g., canon, .progressive,…)
      * …and professions (e.g., .attorney, .doctor,…)
  + These should be showing up soon
* Geographic structure
  + [http://www.iana.org/domains/root/db#](http://www.iana.org/domains/root/db)
  + TLDs are USA-centric
  + Geographic TLDs (ccTLD) are used for organizations in other countries
  + Note: some countries sell their ccTLDs
* Geographic structure
  + Countries define their own internal heirachy:
  + .ac.jp and .edu.au are used for academic organizations   
    in Japan and Australia, respectively
  + DNS domains are logical concepts and need not correspond to physical location of organizations
    - E.G., chinatoday.com is hosted partly in Beijing, partly in San Francisco
* DNS: Domain name system
  + Distributed database
    - Implemented in hierarchy of many name servers
  + Application -layer protocol:
    - running at host, routers, & name servers to resolve names (address/name translation)
  + DNS services
    - hostname to IP address translation
    - Web server aliasing
      * canonical, alias names
      * mail server aliasing
      * load distribution
    - Discussion questions: Lookups would be faster if there one massive central database instead of having to search a hierarchical distributed database, Why not centralize DNS?
* DNS: a distributed, hierarchical database
  + Root DNS servers
    - com DNS servers
      * yahoo.com DNS servers
      * amazon.com DNS servers
    - org DNS servers
      * pbs.org DNS servers
    - edu DNS servers
      * poly.edu DNS servers
      * umass.edu DNS servers
  + Example: Client wants IP for [www.amazon.com](http://www.amazon.com)
    - 1st approximation
      * client queries root to find .com DNS server
      * client queries .com DNS server to get amazon.com DNS server
      * client queries amazon.com DNS server to get IP address for amazon.com
* DNS: root name servers
  + Contacted by local name server that cannot resolve name
  + Root name server
    - gets mapping from authoritative name server if name mapping not known, and returns mapping to local name server
    - 13 root name “servers” worldwide
* DNS records
  + DNS: distributed database storing resource records (RR)
    - RR format (name, value, type, ttl)
  + type = A
    - name is hostname
    - value is IP address
  + type = NS
    - name is domain
    - value is hostname of authoritative name server for this domain
  + type=CNAME
    - name is alias name for some canonical (the real) name
    - www.ibm .com is really servereast.backup2.ibm.com
    - value is canonical name
  + type =MX
    - value is name of mailserver associated with name
  + ttl= <seconds to remain in cache>
    - set to 24hrs expressed in seconds
* DNS protocol, messages
  + query and reply messages, both with same message format
    - message header
      * identification: 16 bit # for query, reply to query uses the same #
      * flags
        + 16 bit field
        + query or reply
        + recursion desired
        + recursion available
        + reply is authoritative
      * Questions
        + 16 bit as well
      * Answers
      * # of authoritative RR
      * # of additional RR
    - Note: all multi-byte numeric values must be in big-endian order (network order).
    - Why does DNS use UDP instead of reliable TCP?
* Domain name 🡨🡪 IP address lookup, reverse lookup
  + <http://remote.12dt.com/rns/>
  + <http://www.dnsstuff.com/>
    - and many others
* Summary Lecture 13
  + Defintions
    - IP Address
    - Domain name
    - TLD
  + DNS
    - structure
    - management
    - lookups
    - protocol, messages
  + This concludes our coverage of the Application layer
    - next: the transport layer