# **ON Line Class February 8, 2016**

# **12. Electronic Mail** p. 119 fig 2.16 –asynchronous communications medium –usually a 2 tiered client server application

## a. Three major components:

## **user agents** : outlook, Mozilla thunderbird, apple Mail –popular Gui user agents

## **mail servers**

## **simple mail transfer protocol: smtp** –reliable

## 1)**User Agent**

## a.k.a. “mail reader”

## composing, editing, reading mail messages

## e.g., Eudora, Outlook, elm, Netscape Messenger

## outgoing, incoming messages stored on server

## **2)Mail Servers**

## mailbox contains incoming messages (yet to be read) for user

## message queue of outgoing (to be sent) mail messages

## smtp protocol runs between mail servers to send email messages

### client: sending mail server

### “server”: receiving mail server

# **3) smtp [RFC 821]-legacy system around since 1982—port 25**

## uses tcp to reliably transfer email message from client to server, port 25

## direct transfer: sending server to receiving server and from senders client to senders mail server.

## **b. three phases of transfer p. 121 e.g**

### **handshaking** (greeting)\*\*this is different from the tcp handshaking. TCP is already connected when this happens. This is one mail server saying hi to another.

### **transfer of messages**

### **closure**

## **c. command/response interaction**

### commands: **ASCII text**

### response: status code and phrase

## messages **must be in 7-bit ASCII 128 combos. Look up ASCII chart.**

1. Therefore, we have problems with multimedia messages.

# Sample smtp interaction p. 122 fig 2.17—p. 122 bottom eg

# try smtp interaction for yourself: manually sending to email without an email client. **DO THIS!**

# **telnet servername 25 p. 124 try this**

## see 220 reply from server

## enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

## above lets you send email without using email client (reader)

# **d. smtp: final words**

## smtp uses **persistent connections**—therefore can transfer multiple messages.

## smtp requires **that message (header & body) be in 7-bit ascii**

## **certain character strings are not permitted in message (e.g., CRLF.CRLF).** Thus message has to be encoded (usually into either base-64 or quoted printable)

## smtp server uses CRLF.CRLF to determine end of message.: A single period on a line followed by a blank line. **This is called a delimiter.**

## **Comparison with http p.122 webmail history**.

## http: **pull** –users pull information from a server

## smtp: **push**-users push information onto the mail server

## **both** have ASCII command/response interaction, status codes but smtp needs 7 bit ascii

## http: each object is encapsulated in its own response message

## smtp: multiple objects message sent in a multipart message

# **e. Mail message format protocols need to have formats (syntax) p. 125**

## smtp: protocol for exchanging email messages

## RFC 822: standard for text message format: within the message itself

* 2 parts:

## header lines, e.g.,

### To:

### From:

### Subject:

### *Header lines different from smtp commands that are used in handshake*! (blank line)

## body

### the “message”, ASCII characters only

# **f. Message format: multimedia extensions . What if you want to send more than just ASCII text?**

## **MIME**: multimedia mail extension, RFC 2045, 2056 multipurpose internet mail extensions

## additional lines in msg header declare MIME content type

# MIME types 2 headers

# **Content transfer encoding**—says message has been encoded into ascii and says what type of encoding was used to get it into 7 bit ascii—so that it knows what to use to get it back to its original form

# **Content-Type: type/subtype; parameters –what actions should be taken on the body of the message**

**HOW DOES A USER ACCESS HIS/HER MAIL?**

# **g. Mail access protocols** p. 126 fig 2.18 **Transfer from mailserver to the users** PC—when a user wants to access email, email client contacts his mail server sending an imap or pop3 packet that asks for the content of the mailbox. Remember that

## **SMTP**: delivery/storage to receiver’s server – **only push operation**

## **Mail access protocol: retrieval from server – when agent (local pc) is physically separate from mail server**

### **POP3:** Post Office Protocol [RFC 1939] p. 129

#### authorization (agent <-->server) and download

### **IMAP**: Internet Mail Access Protocol [RFC 1730] p. 131

#### more features (more complex)

#### manipulation of stored messages on remote server

### **HTTP**: Hotmail , Yahoo! Mail, etc. **email using web**-considered a three tiered architecture

# 1**) POP3** protocol on top of TCP so tcp connection is set up

i) **authorization phase**—handshake and authorize p. 128

**client commands**: **S: +OK POP3 server ready**

## **user:** declare username **C: user alice**

**pass:** password **S: +OK**

### **C: pass hungry**

## server responses

**+OK S: +OK user successfully logged on**

### **-ERR**

## ii) **transaction phase**, client: p.128

## **list:** list message numbers

## **retr:** retrieve message by number

## **dele:** delete

## quit

**C: list**

**S: 1 498**

**S: 2 912**

**S: .**

**C: retr 1**

**S: <message 1 contents>**

**S: .**

**C: dele 1**

**C: retr 2**

**S: <message 1 contents>**

**S: .**

**C: dele 2**

**C: quit**

**S: +OK POP3 server signing off**

# **IMAP-** more features and more complex than pop3

# -maintains user state information across IMAP sessions

- can get part of a message

# Web-based email: through http and browser p.129 where browser acts as the user agent. **E.g**. write a message in Yahoo and send to web server **inside an HTTP packet**. Web server runs a program that takes the info from the request and **builds an SMTP packet🡪 mailserver** which processes it as though it came from a client computer 🡪destination mailserver 🡪 user mailbox.

When the receiver checks mail through the web browser, an HTTP request goes to Web server. Program on web server processes the request and sends either an IMAP or POP3 packet to mail server. Mail server responds with IMAP or POP packet which program on web server converts to HTTP response and sends to client. Client then displays email in web browser. In this case no need for an email client on your machine, just use the web browser.

# 13. **DNS: Domain Name System** An application protocol where the user doesn’t interact directly. Provides a core internet function; **translation and mapping of names and addresses. Runs over UDP on port 53 (because they are short).** It is considered an application protocol because it runs between end systems using client server protocols and it relies on end to end transport protocol to transfer DNS messages between communicating end systems. P. 131

## a. People: many identifiers:

### SSN, name, Passport #

## b. Internet hosts, routers:

### **IP address (32 bit)** - used for addressing datagrams (packets)- **4 byte form x.x.x.x** where the period separates each byte in decimal notation 0-255. Preferred by routers

### “host name”, e.g., gaia.cs.umass.edu - used by humans

1. also used by other application layer protocols to translate user supplied host names with IP addresses.

## Q: **Why not centralize DNS? P. 133**

## single point of failure

## traffic volume

## distant centralized database

## maintenance

## **Q: How do we map between IP addresses and name?**

## **c. Domain Name System: 1982**

## *distributed database* implemented in hierarchy of many *name servers around the world*

## *application-layer protocol* host, routers, name servers communicate to *resolve* names (address/name translation)- to query the distributed database **p. 131 mid e.g.**

### note: **core Internet function implemented as application-layer protocol**

### **complexity at network’s “edge” structuring principle of the internet is to put complexity at the edge of the network so that you don’t need to access the core to implement new applications.**

1. **Other Services:**
2. **Host aliasing**: using an alias for a canonical hostname. DNS can be invoked to provide the canonical hostname given an alias.
3. **Mail server aliasing**: we want mnemonic email addresses for ease of use. DNS can get the canonical hostname as well as the IP address of the host
4. **Load distribution**: can replicate sites over multiple servers especially busy ones. Therefore a set of IP addresses is associated with a canonical hostname. Queries are rotated among the different hosts p. 132

## no server has all name-to-IP address mappings:

## **d. 3 types of servers p. 134 fig 2.19**

# 1) **Root name servers** p. 135 fig 2.20–keeps a cache of names it knows.

## contacted by local name server that cannot resolve name

## **root name server:**

### contacts authoritative name server if name mapping not known

### gets mapping

### returns mapping to local name server

## ~ 13 root name servers worldwide (actually 13 clusters of servers)—**ICANN** controls these root server names

* hierarchical setup based on the organization to which the machine belongs

2) **Top Level Domain servers:** com, org, net edu and gov servers

## **3) Authoritative name server:** every organization with accessible hosts on the Internet must provide publicly accessible records that map the names of those hosts to IP addresses. An organization’s authoritative name server houses these DNS records.

## **4) local DNS servers**: don’t belong to hierarchy but is central to DNS

### each ISP, company has *local (default) name server*

### host DNS query first goes to local name server and IP is passed back. On client side browser passes the hostname from the URL to the client side DNS that generates the query.

# **e. Simple DNS** e**xample** p. 137 fig 2.21

## host **cis.poly.edu** wants IP address of **gaia.cs.umass.edu**

## 1. Contacts its local DNS server, **dns.poly.edu**

## 2. **dns.poly.edu** contacts root name server, if necessary that responds with list of IP addresses for the TLD servers responsible for .edu.

## local DNS server sends to TLD server and it responds with address of the authoritative name server for umass that sends back the address for gaia.cs.umass.edu

## We needed 8 messages for 1 address here.

# DNS example

## Root name server:

## may not know authoritative name server

## may know *intermediate name server:* who to contact to find authoritative name server

* so I contact dns.umass .edu that is intermediate because dns.cs.umass.edu is acting as the administrative server for all hosts in cs dept.

# **f. DNS: recursive queries p. 138 fig 2.22**

## **recursive query:**

## puts burden of name resolution on contacted name server

## heavy load?

## **iterated query:**

## contacted server replies with name of server to contact

## “I don’t know this name, but ask this server”

# **g. DNS: caching and updating records p. 139 eg**

## once (any) name server learns mapping, it *caches* mapping locally

### cache entries timeout (disappear) after some time ~2 days

## update/notify mechanisms under design by IETF

### RFC 2136

### http://www.ietf.org/html.charters/dnsind-charter.html

# **h. DNS records p. 139**

## DNS: distributed db storing resource records (RR)

RR format: **(name, value, type,ttl)**

* **Type=A**

**name** is hostname

**value** is IP address

* **Type=CNAME**

**name** is an alias name for some “cannonical” (the real) name

**value** is cannonical name

* **Type=MX**

**value** is hostname of mailserver associated with **name**

## Type=NS

### **name** is domain (e.g. foo.com)

### **value** is IP address of authoritative name server for this domain

# **i. DNS protocol, messages p141. Fig 2.23**

## DNS protocol : *query* and *repy* messages, both with same *message format*

msg header

* identification: 16 bit # for query, reply to query uses same #
* flags:
  + query or reply
  + recursion desired
  + recursion available
  + reply is authoritative

That is all for applications: Now you have a feel for protocols: request/response; data for message; structure of messages with fields.

* + - Sending a query message from your host to a DNS server directly: use nslookup. You will do this in the DNS lab.

**B. Peer to Peer file sharing Applications p. 145**

* highest traffic application on the Internet
* do eg on p 145
* content is distributed directly among peers over a TCP connection
* **Bit Torrent**
  + Torrent= peers participating in the distribution of a file
  + Peers download equal sized chunks typically 256K from one another
  + Peers also upload chunks to other peers
  + Peers may leave the torrent at any time
  + Infrastructure
  + Tracker node in each torrent keeps track of who is in the torrent: nodes identify themselves to the tracker node
  + When a node joins, tracker selects a subset of peers and sends the IP addresses of these peers to the node
  + P. **150 fig 2.26** e.g. –tries to establish TCP connections with peers. Here she has 3 neighboring nodes. She asks these neighbors (who may fluctuate by moving in and out of the torrent) for chunks of the file that she needs.
  + Alice needs to decide who to ask for chunks and also has to decide who to send chunks to
    - Rarest first; ask for the ones that seem to have the fewest copies out there
    - Gives chunks to the neighbors that are supplying her at the highest rate—eliminates free riding e.g. p. 150 bottom –look at choked and unchecked peers.
  + Index- peers dynamically update an index that maintains information about the title, copy, IP address of peer, etc. Peers can search the index for information.
* How do we organize and search an index in a community of peers (P2P filesharing)
* How can a peer who wants a file determine the IP address of the peer who has it? This is a non-trivial problem because peers connect and disconnect all the time and the IP addresses are dynamically allocated.
* Use large distributed databases. If interested read section 2.6.2