Chapter7 Application Layer

王昊翔：hxwang@scut.edu.cn

School of Computer Science & Engineering ,SCUT

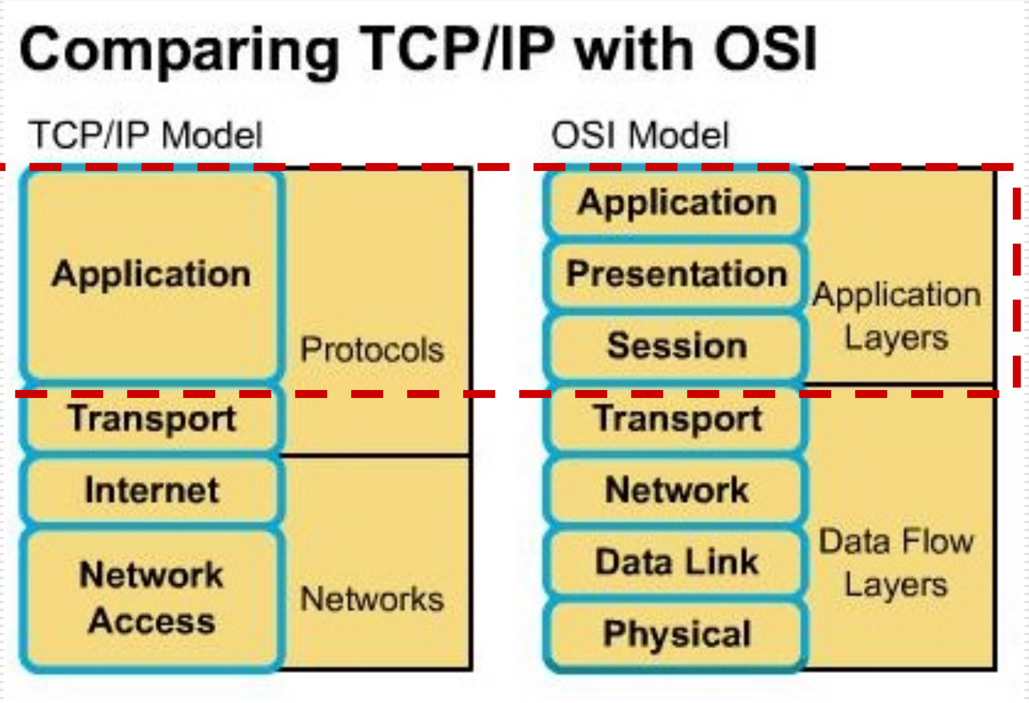
Communication & Computer Network key-Lab of GD

Review

- Transport layer
  - Functionalities/services provided
  - Distinguish from network layer
- UDP and TCP
  - Segment headers
  - Socket
  - Three-way hank shake
  - Two-army problem

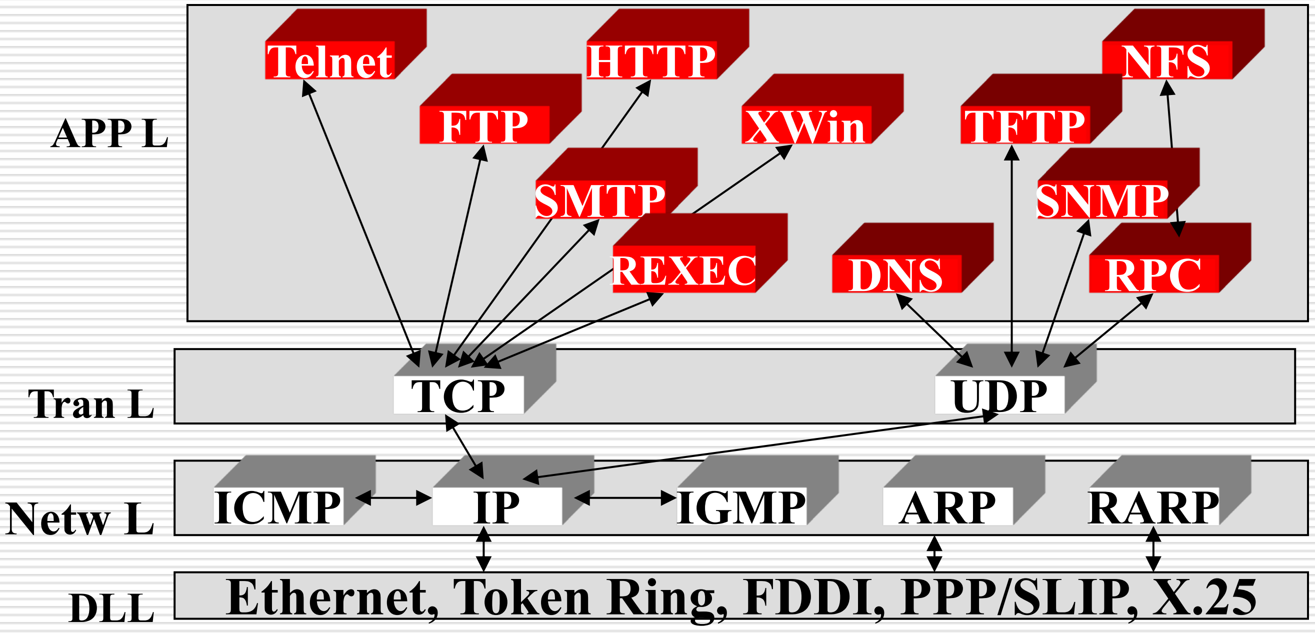
Outline

- Overview of application layer
- Domain Name System
- Important application
  - E-mail
  - World wide web
  - ftp
  - telnet
  - Multimedia



Main function of application layer

- Be nearest to user, provide application program the network communication.

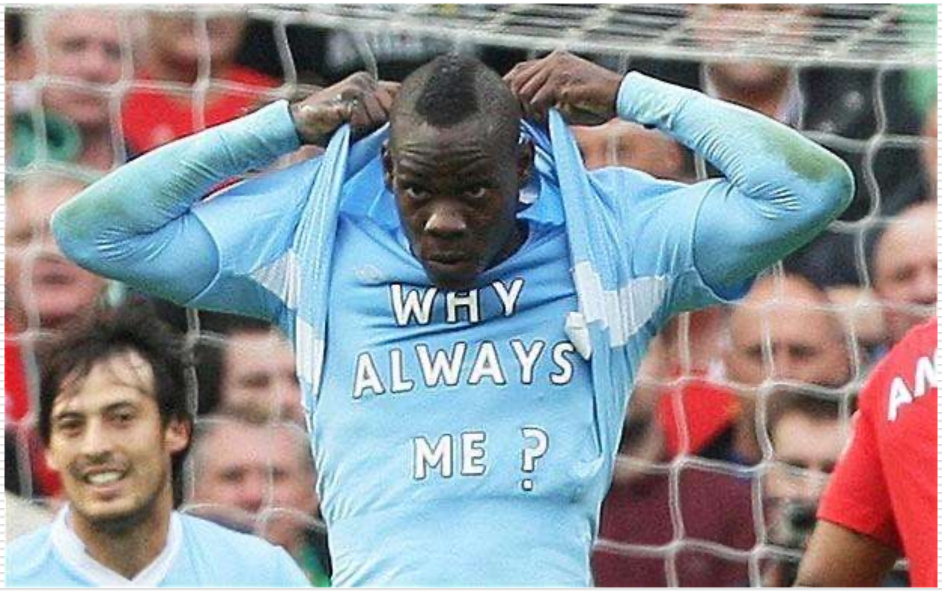


Property of application layer

- No application layer, no network communication support;
- The only layer does not provide service to it's upper layer;
- It provide service to application outside reference-model.
- Application program can be divided into:
  - Direct: Browser , e-mail ,FTP , Telnet
  - Indirect: Word , resource manager , (via redirector)

Domain Name System

- Already have IP and port number
  - Socket can identify a process on a particular host
- WHY still need DNS

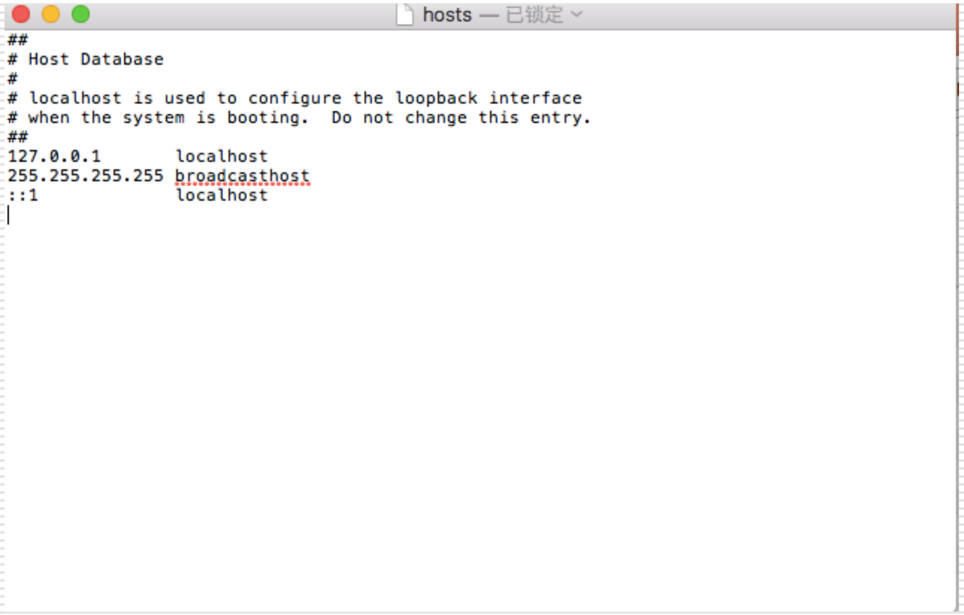


Domain Name System

- Using IP addresses as absolute machine addresses on the Internet is not very practical, three reasons:
  - Computers can frequently change IPs, rendering using an IP address to access the machine useless.
  - Be not easy to remember IP
  - Multiple servers doing the same thing
- A system of using a name to access the machine was devised to overcome this problem.

Domain Name System

- Hosts file /etc/hosts/ or "%WINDIR%\System32\drivers\etc"
- In the ARPANET, there was simply a file, **hosts**, that listed all the hosts and their IP addresses
- **small network**


hosts — 已锁定

```
##
# Host Database
#
# localhost is used to configure the loopback interface
# when the system is booting.  Do not change this entry.
##
127.0.0.1       localhost
255.255.255.255 broadcasthost
::1             localhost
```

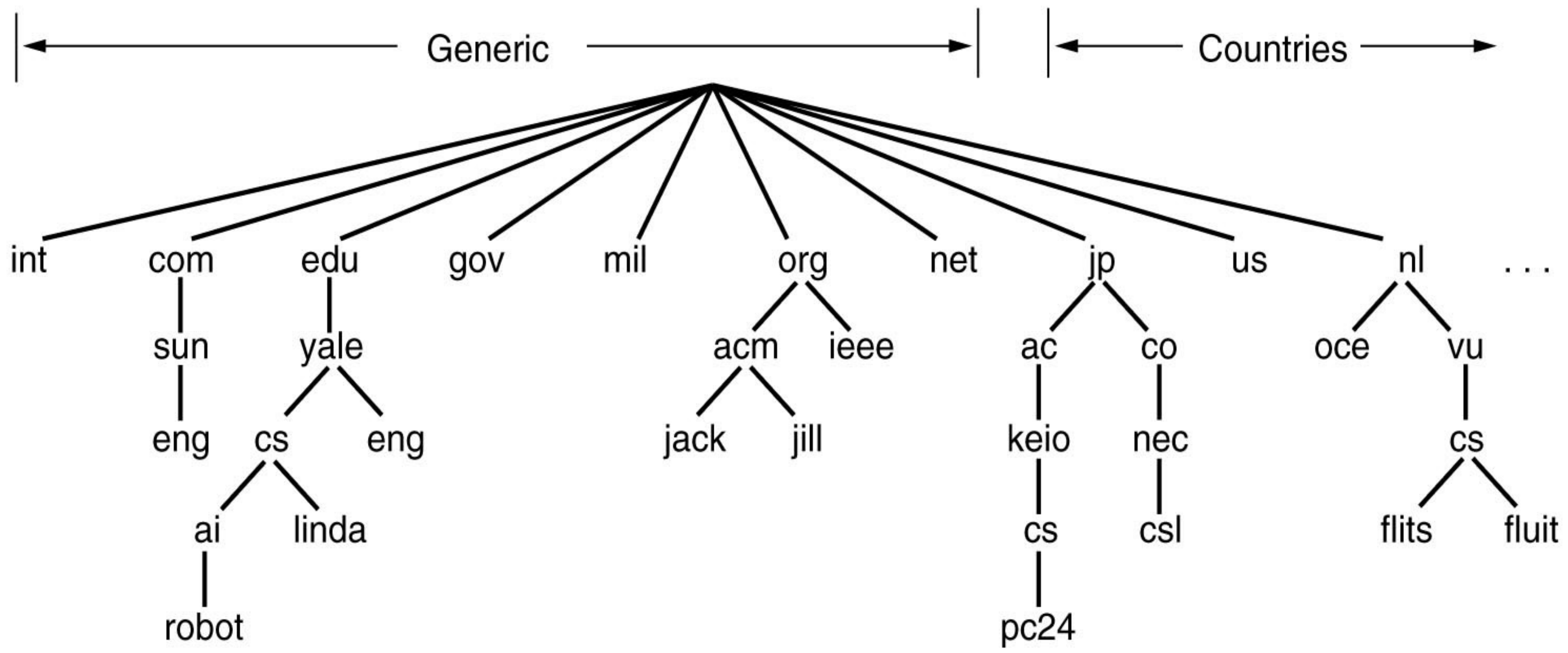
Domain Name System (cont'd)

- DNS is a hierarchical, domain-based naming scheme and a distributed database system for implementing this naming scheme
- Usage:
  - Map name onto an IP address, an application program calls an library procedure, called resolver, passing it the name as a parameter (i.e. gethostbyname() is an resolver)
  - The resolver sends UDP packet to a local DNS server which looks up the name and returns the IP address to the resolver
  - The resolver returns the IP address to the application

DNS Name Space

- Internet is divided into over 200 top level domains.
  - Each domain is divided into sub-domains, which are further partitioned, etc..
  - All domains can be represented by a tree.
    - The leaves of the tree represent domains that have no sub-domains (but contain machines)
    - A leaf domain may contain a single host or represent a company and contain thousands of hosts
- Top level domains could be generic and country domains.

DNS Name Space (cont'd)



Domain Names

- Each domain is named by the path upward from it to the (unnamed) root. The components are separated by periods (pronounced "dot").
- Can be either absolute (ends with a period i.e. eng.sun.com.) or relative (it doesn’t end with a dot)
  - Relative ones have to be interpreted in a context to find the true meaning
  - Both of them refers to a specific node in the tree and all the nodes under it
- Are case insensitive (edu, Edu, EDU are same thing)
- Components names can be up to 63 characters and full names should not exceed 255 characters.
- http://www.llanfairpwllgwyngyllgogerychwyrndrobwllllantysiliogogogoch.co.uk/

Domain Names (cont'd)

- Each domain controls how it allocates the domains under it (full control it's sub domain)
  - i.e. Japan makes a domains ac.jp and co.jp
  - Our country?
- To create a new domain, permission is required from the domain that will include it; once created, it can create sub-domains without having to ask permission from the higher up domains.
- Naming follows organizational boundaries, not physical networks.

Resource Records

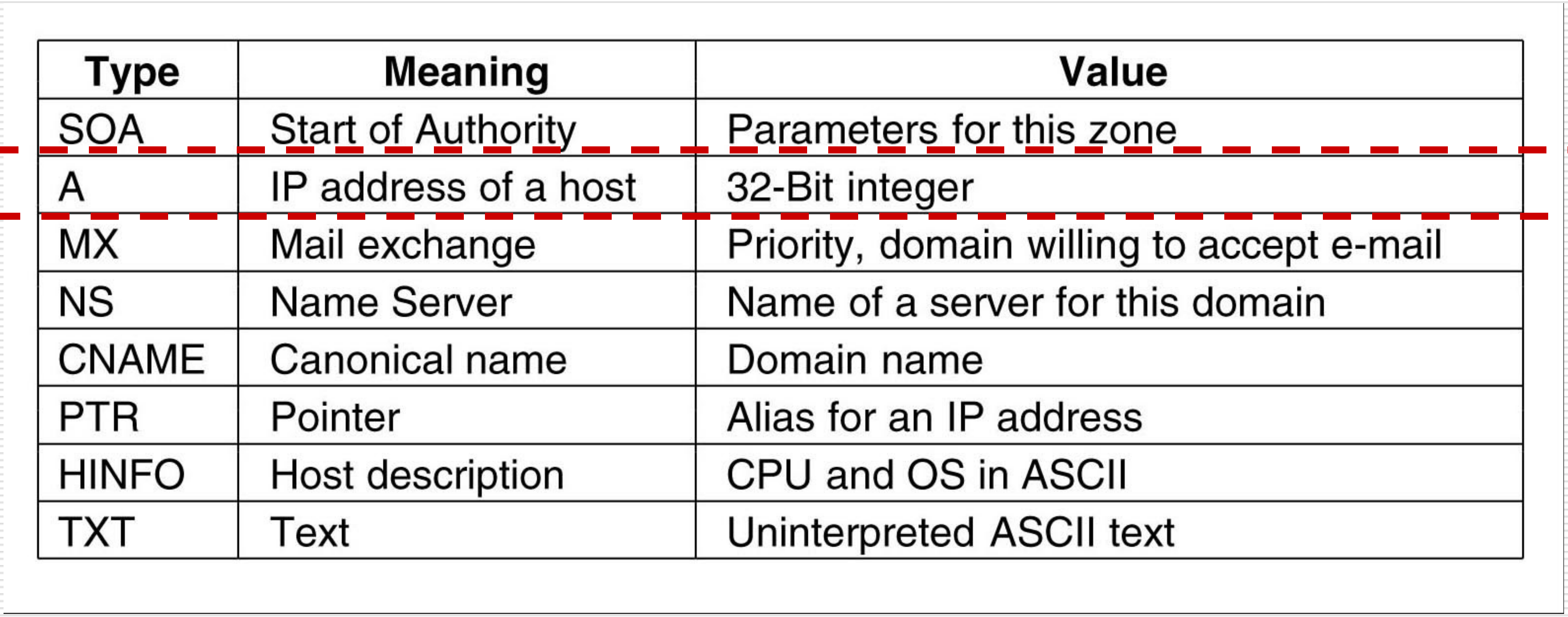
- Every domain, either single host or a top level domain, should have a set of resource records
- When a resolver gives a domain name to DNS, what it gets back are the resource records associated with that name; thus the primary function of DNS is to map domain names onto resource records
- A resource record has five parts:
  - Domain name
  - Time to Live
  - Class
  - Type
  - Value

Resource Records (cont'd)

- **Domain_name** - tells the domain to which this record applies
  - Normally many records exist for each domain and each copy of the database holds info about multiple domains
  - This field is the primary search key used to satisfy queries
  - The order of the records in the database is not important
- **Time_to_live** - gives an indication of how stable the record is
  - Info that is highly stable, is assigned a large value, such as 86400 (number of seconds in a day)
  - Info that is highly volatile is assigned a small value, such as 60 (1 minute)
- **Class field** - always IN for Internet information.

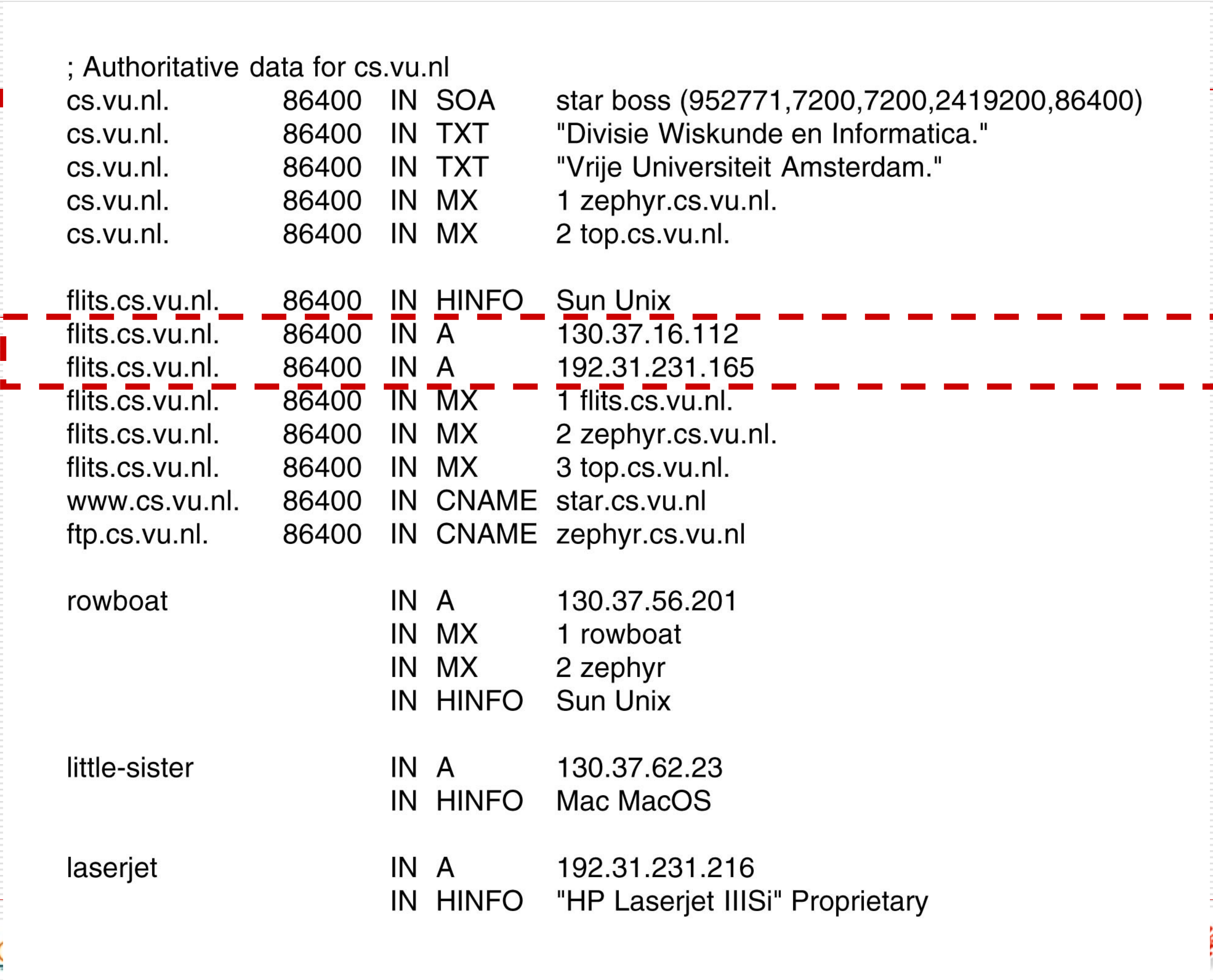
Resource Records (cont'd)

- **Type** - tells what kind of record this is.



Type	Meaning	Value
SOA	Start of Authority	Parameters for this zone
A	IP address of a host	32-Bit integer
MX	Mail exchange	Priority, domain willing to accept e-mail
NS	Name Server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
HINFO	Host description	CPU and OS in ASCII
TXT	Text	Uninterpreted ASCII text

Resource Records Example



```
; Authoritative data for cs.vu.nl
cs.vu.nl.        86400  IN  SOA     star boss (952771,7200,7200,2419200,86400)
cs.vu.nl.        86400  IN  TXT     "Divisie Wiskunde en Informatica."
cs.vu.nl.        86400  IN  TXT     "Vrije Universiteit Amsterdam."
cs.vu.nl.        86400  IN  MX      1 zephyr.cs.vu.nl.
cs.vu.nl.        86400  IN  MX      2 top.cs.vu.nl.

flits.cs.vu.nl.  86400  IN  HINFO   Sun Unix
flits.cs.vu.nl.  86400  IN  A       130.37.16.112
flits.cs.vu.nl.  86400  IN  A       192.31.231.165
flits.cs.vu.nl.  86400  IN  MX      1 flits.cs.vu.nl.
flits.cs.vu.nl.  86400  IN  MX      2 zephyr.cs.vu.nl.
flits.cs.vu.nl.  86400  IN  MX      3 top.cs.vu.nl.
www.cs.vu.nl.    86400  IN  CNAME   star.cs.vu.nl
ftp.cs.vu.nl.    86400  IN  CNAME   zephyr.cs.vu.nl

rowboat                 IN  A       130.37.56.201
                        IN  MX      1 rowboat
                        IN  MX      2 zephyr
                        IN  HINFO   Sun Unix

little-sister           IN  A       130.37.62.23
                        IN  HINFO   Mac MacOS

laserjet                IN  A       192.31.231.216
                        IN  HINFO   "HP Laserjet IIISi" Proprietary
```

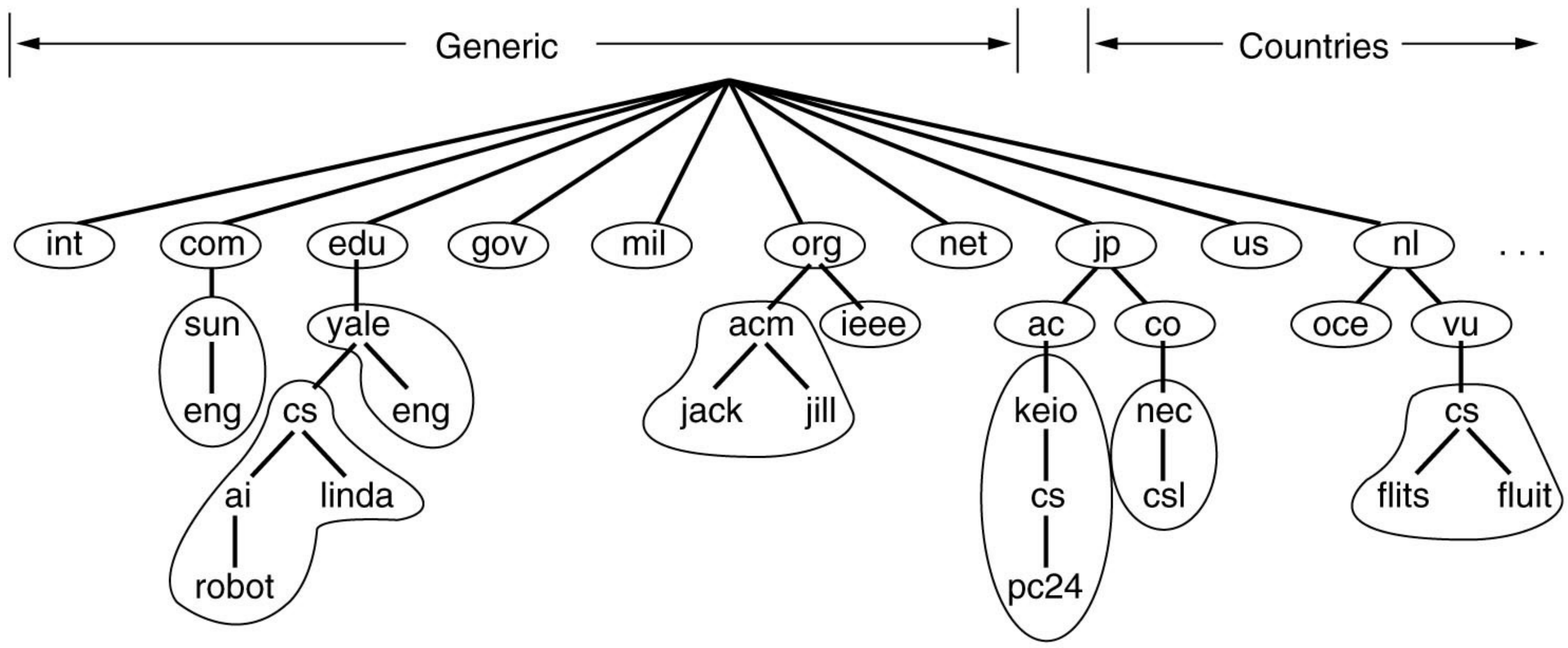
Resource Records (cont'd)

- Questions for you
  - How these records generated and synchronized
  - SOA and NS
  - DNS hijack and DNS (cache) pollution
- Discussion topics for next time

Name Servers

- We need more than just one name server for the whole Internet.
- The DNS name space is split into non-overlapping zones, each of which has a primary name server.
- The primary name server gets its information off of disk, and shares that information with secondary name servers.
- Some name servers for a zone can be located outside of the zone to improve reliability.
- Where the zone boundaries are placed within a zone it is up to the zone's administrator. This decision is made based on how many name servers are needed and where.

Part of the DNS Name Space Showing the Division Into Zones



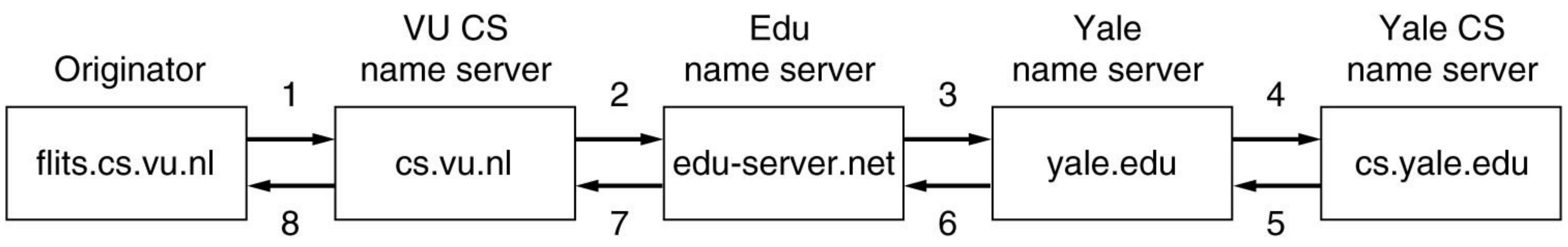
Resolving Domain Names

- When a resolver has a query about a domain name, it passes the query to one of the local name servers.
- If the domain being sought falls under the jurisdiction (权限)of the name server, it returns the authoritative resource records.
  - An authoritative record is one that comes from the authority that manages the record and is thus always correct. But cached records may be out of date.
- If the domain is remote and no information about the requested domain is available locally, the name server sends a query message to the top-level name server for the domain requested.

How a Resolver Looks Up a Remote Name in Eight Steps

- **Recursive query (递归查询、递归解析)**

Flits.cs.vu.nl wants to know IP of linda.cs.yale.edu



- **recursive query（迭代查询）**
  - **Return IP or the name of the next server to try.**

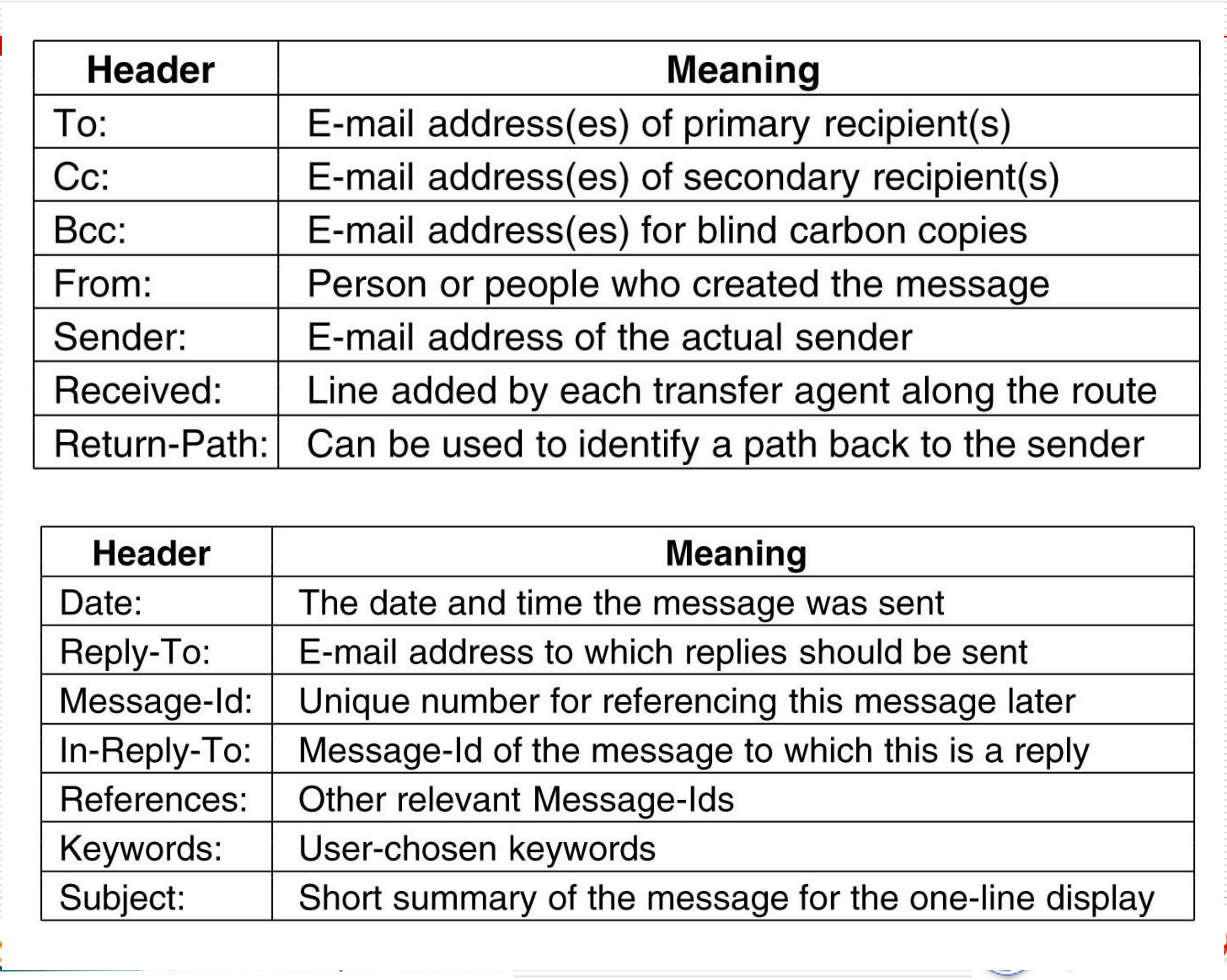
Electronic Mail

- E-mail system consists of **two parts**
  - User agents(UA), which allow people to read and send email
    - Local programs that provide a command based or graphical method for interacting with e-mail system
  - Message transfer agents (MTA), which move the messages from source to destination
    - Are typically system daemons or processes that run in background, having the job to move messages.

E-mail Message Format

- Basic ASCII e-mail message using RFC 822
  - Messages consists of a primitive envelope (described in RFC821), some number of header fields, a blank line and then the message body
  - Each header field (logically) consists of a single line of ASCII text, a colon（：） and, for most fields, a value
  - RFC822 was designed long ago and doesn’t clearly distinguish between the envelope fields and the header fields
    - This was revised in RFC 2822, however, wasn’t possible to completely redo it due to the widespread usage
  - Users are allowed to invent new headers for their own private use, provided that these headers start with the string X-.

RFC 822 Message Header



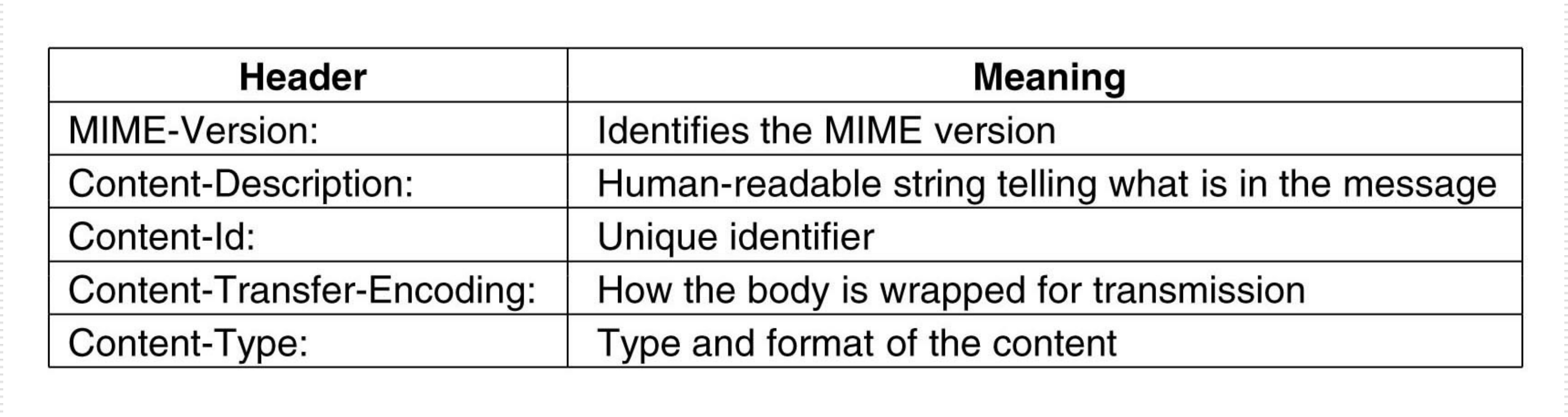
Header	Meaning
To:	E-mail address(es) of primary recipient(s)
Cc:	E-mail address(es) of secondary recipient(s)
Bcc:	E-mail address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	E-mail address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

Header	Meaning
Date:	The date and time the message was sent
Reply-To:	E-mail address to which replies should be sent
Message-Id:	Unique number for referencing this message later
In-Reply-To:	Message-Id of the message to which this is a reply
References:	Other relevant Message-Ids
Keywords:	User-chosen keywords
Subject:	Short summary of the message for the one-line display

E-mail Message Format (cont'd)

- MIME – the Multipurpose Internet Mail Extensions
  - Solves problems related to send and receive of messages in non Latin alphabets, languages with no alphabets (Chinese or Japanese), messages not containing text at all (video or audio)
  - RFC1341 and updated in RFCs 2045-2049.
- The basic idea of MIME is to continue to use RFC 822 format, but to add structure to the message body and define encoding rules for non-ASCII messages
  - By not deviating the RFC822, MIME messages can be sent using the existing mail programs and protocols
  - All it has to be changed is the receiving and sending programs, which each user can do it.

RFC 822 headers added by MIME



Header	Meaning
MIME-Version:	Identifies the MIME version
Content-Description:	Human-readable string telling what is in the message
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Type and format of the content

- **Content-Transfer-Encoding**
  - **tells how the body is wrapped for transmission through a network that may object to most characters other than letters, numbers, and punctuation marks.**

Email Message Transfer

- Message transfer mechanism is concerned with relaying messages from the originator to the destination
  - This can be done by establishing an transport level connection between the source and the destination and then just transfer the message
- SMTP – Simple Mail Transfer Protocol
  - Source machine establishes a TCP connection on port 25 on destination machine, where SMTP daemon listens.
  - If a message can't be delivered, an error report containing the first part of the undeliverable message is returned to the sender

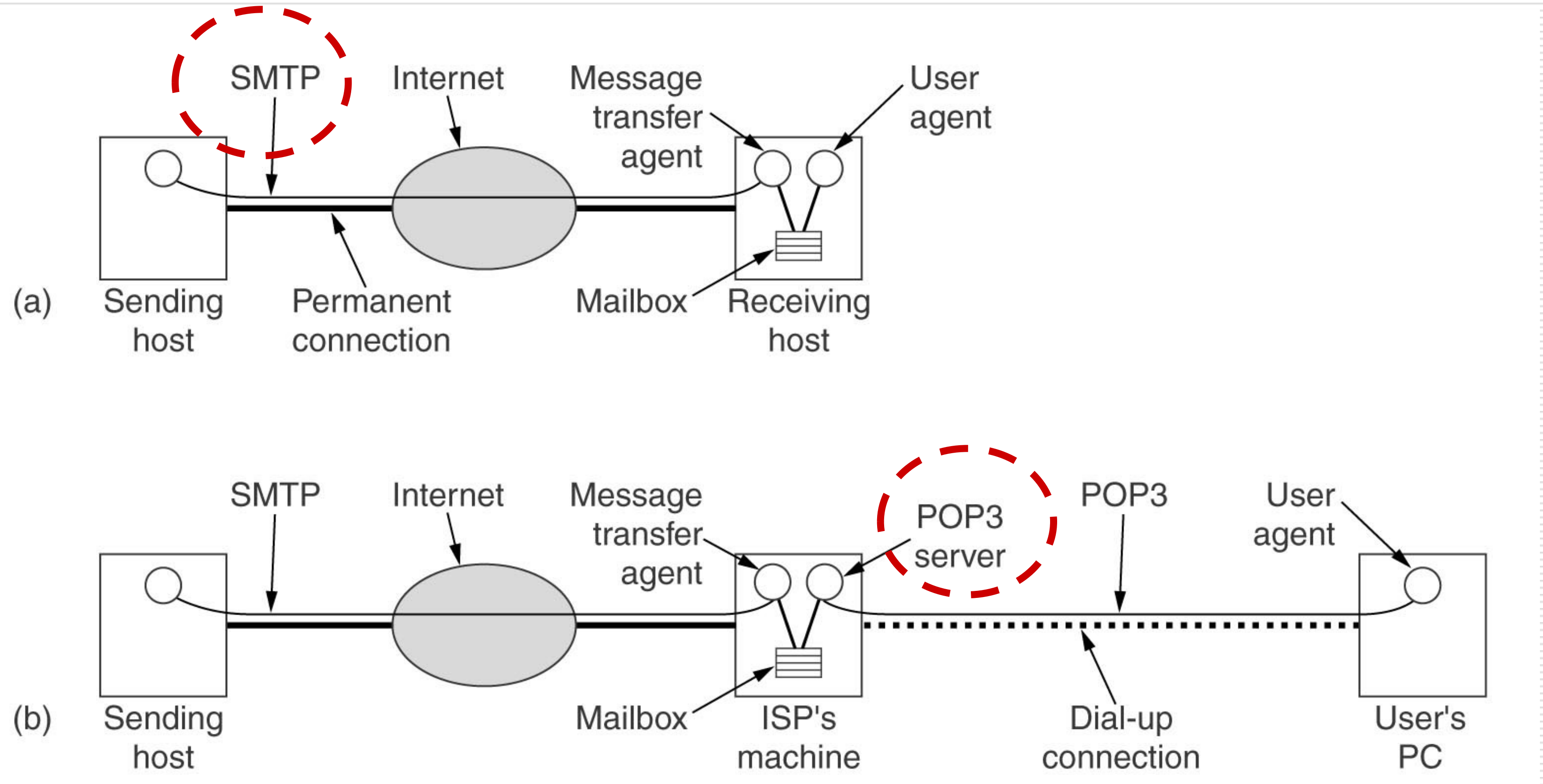
SMTP Protocol （three steps）

- Connection establishment (on port 25)
- Data exchange
  - the client machine (operating as a client) waits for the destination machine (operating as a server) to talk first;
  - the server begins by sending a line of text giving its identity and telling whether is prepared to receive mail;
    - if it is not, then the client releases the connection and tries again latter
  - If the server is willing to accept mail, then the client announces whom the e-mail is coming from and whom it is going to
  - If such recipient exists at the server end, then the client get the go-ahead to send the message
  - The client sends the message, server acknowledges it
- Connection is released

Final Delivery

- Assuming that all machines can send and receive mail all the time, the e-mail model so far works
- This model breaks for people accessing Internet over a dialup connection
  - What happens when John wants to send Mary e-mail and Mary is not currently online?
- One solution is to have a message transfer agent on ISP machine; since this transfer agent can be online all the time, e-mail can be sent 24 hours a day
  - This solution creates another problem: how does the user gets e-mail from ISP’s message transfer agent
    - Solution: create another protocol that allows user transfer agents (on client PCs) to contact the message transfer agent (on ISP’s machine) and allow e-mail to be copied from ISP to the user
    - One such protocol is POP3 (Post Office Protocol Version 3), RFC 1939

Final Delivery (cont'd)



POP3

- Starts when the user starts the mail reader
- Mail reader calls up the ISP (if there is no connection) and establishes a TCP connection with the message transfer agent on port 110;
- The POP3 protocol goes through three states in sequence:
  - Authorization
    - Having user logged in by sending its username and password
  - Transactions
    - User collecting the e-mails and marking them for deletion
  - Update
    - Causes the e-mails to be deleted

POP3 (cont'd)

- USER **userid**
  - **This must be the first command after the connect. Supply your e-mail userid (not the full e-mail address). Example: USER john.smith**
- PASS **password**
  - **This must be the next command after USER. Supply your e-mail password. The password may be case sensitive.**
- STAT
  - **The response to this is: +OK #msgs #bytes Where #msgs is the number of messages in the mail box and #bytes is the total bytes used by all messages. Sample response: +OK 3 345910**
- LIST
  - **The response to this lists a line for each message with its number and size in bytes, ending with a period on a line by itself.**
- RETR **msg#**
  - **This sends message number msg# to you (displays on the Telnet screen). You probably don't want to do this in Telnet (unless you have turned on Telnet logging). Example: RETR 2**

POP3 (cont'd)

- TOP **msg# #lines**
  - **This is an optional POP3 command. Not all POP3 servers support it. It lists the header for msg# and the first #lines of the message text. For example, TOP 1 0 would list just the headers for message 1, where as TOP 1 5 would list the headers and first 5 lines of the message text.**
- DELE **msg#**
  - **This marks message number msg# for deletion from the server. This is the way to get rid a problem causing message. It is not actually deleted until the QUIT command is issued. If you lose the connection to the mail server before issuing the QUIT command, the server should not delete any messages. Example: DELE 3**
- RSET
  - **This resets (unmarks) any messages previously marked for deletion in this session so that the QUIT command will not delete them.**
- QUIT
  - **This deletes any messages marked for deletion, and then logs you off of the mail server. This is the last command to use. This does not disconnect you from the ISP, just the mailbox.**

summary

- Function of application layer
- DNS
- Application
  - E-mail