Computer Networks Basic Protocols

Assoc. Prof. Dr. Berk CANBERK

29 November 2017

-Application Layer-

References:

-*Data and Computer Communications*, William Stallings, Pearson-Prentice Hall, 9th Edition, 2010.

-*Computer Networking, A Top-Down Approach Featuring the Internet*, James F.Kurose, Keith W.Ross, Pearson-Addison Wesley, 6th Edition, 2012.

Some network applications

- E-mail
- Web
- Instant messaging
- Remote login
- P2P file sharing
- Multi-user network games
- Streaming stored video clips
- Internet telephone
- Real-time video conference
- Massive parallel computing

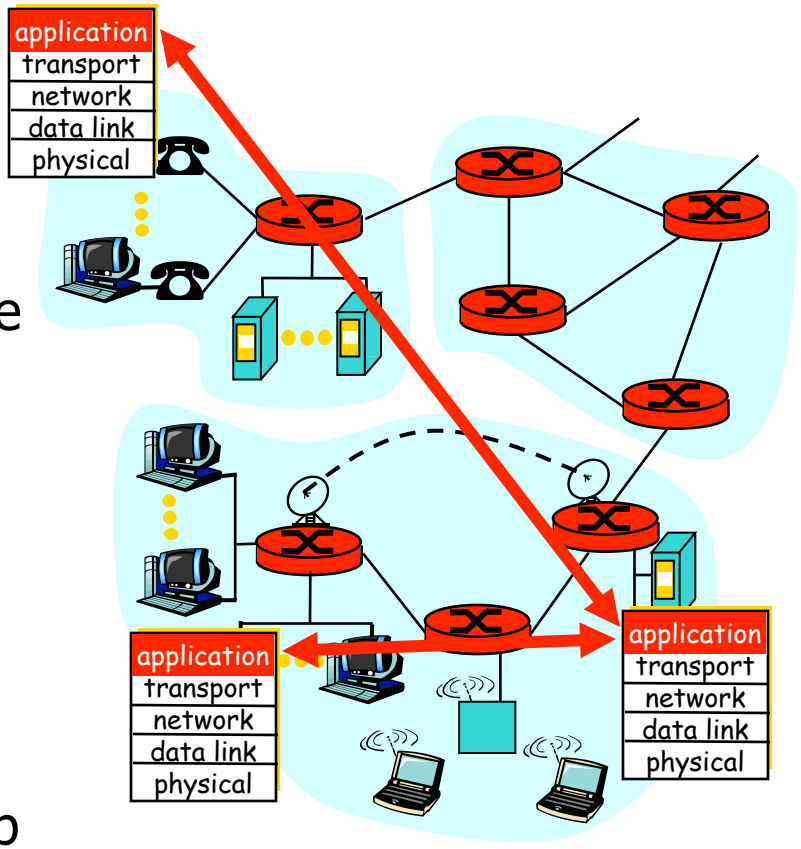
Creating a network application

Write programs that

- run on different end systems and
- communicate over a network.
- e.g., Web: Web server software communicates with browser software

No software written for devices in network core

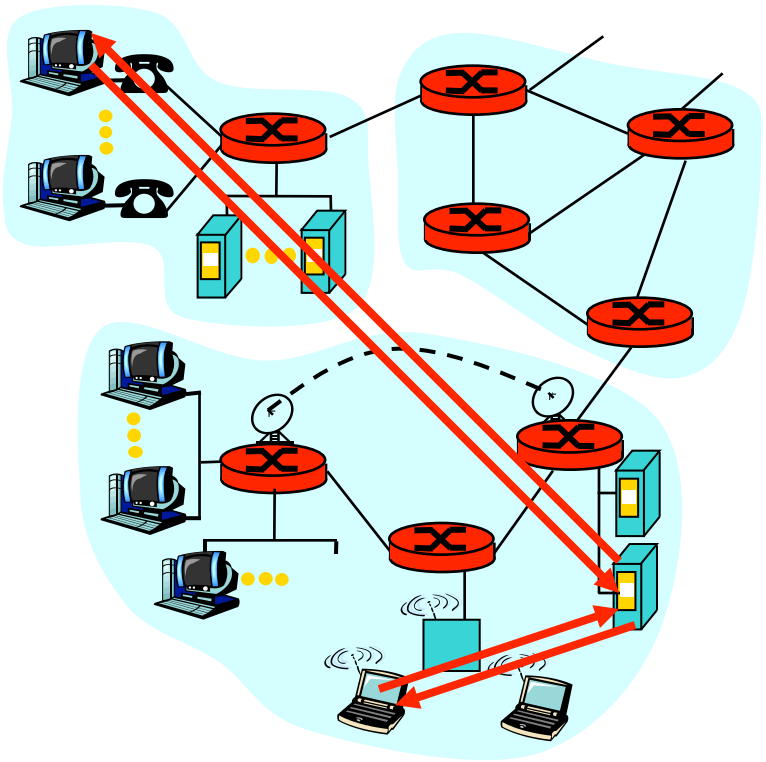
- Network core devices do not function at app layer
- This design allows for rapid app development



Application architectures

- Client-server
- Peer-to-peer (P2P)
- Hybrid of client-server and P2P

Client-server archicture



server:

- always-on host
- permanent IP address
- server farms for scaling

clients:

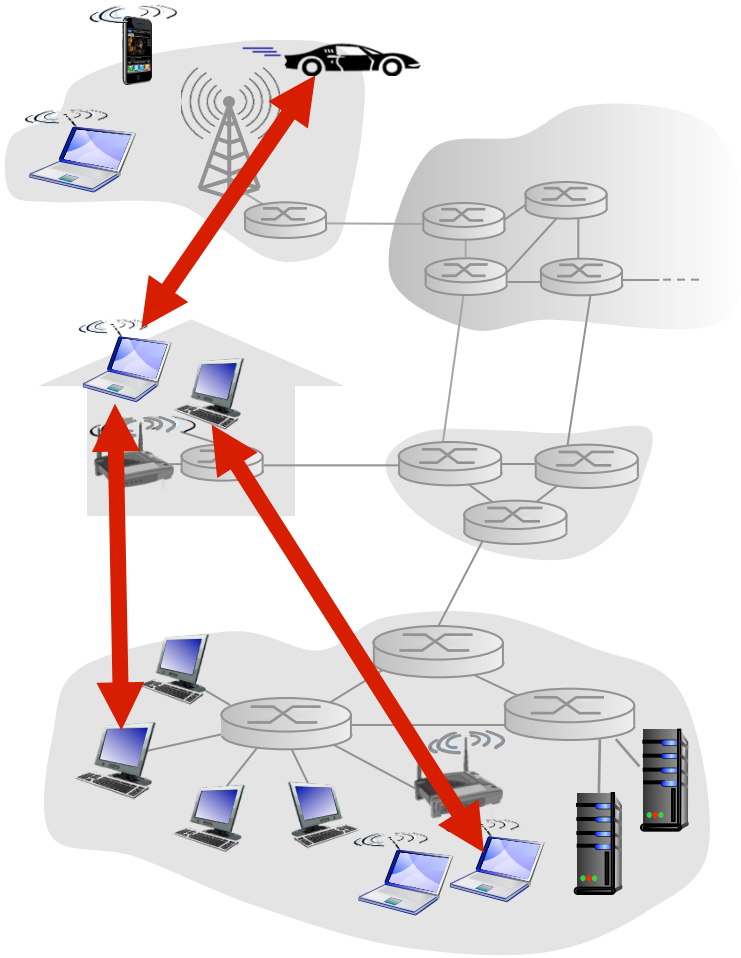
- communicate with server
- may be intermittently connected
- may have dynamic IP addresses
- do not communicate directly with each other

Pure P2P architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses

examples:

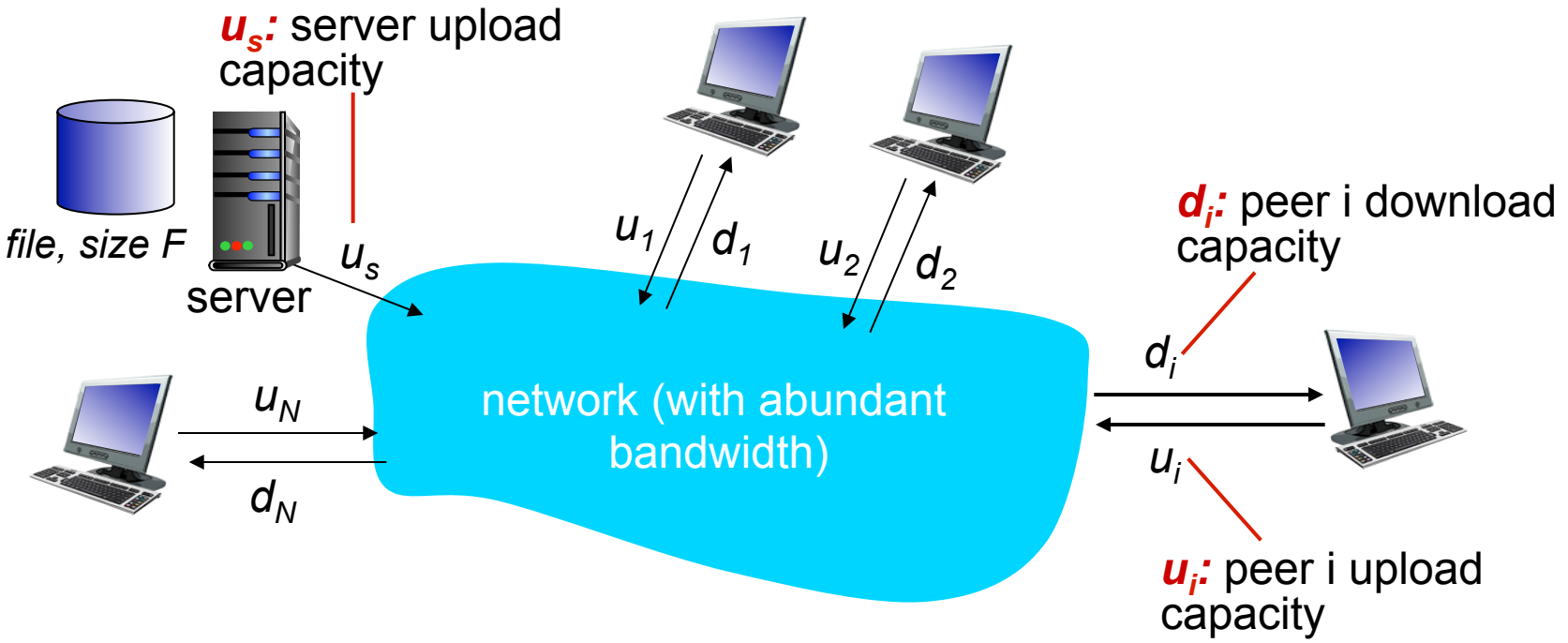
- file distribution (BitTorrent)
- Streaming (KanKan)
- VoIP (Skype)



File distribution: client-server vs P2P

Question: how much time to distribute file (size *F*) from one server to *N* peers?

- peer upload/download capacity is limited resource

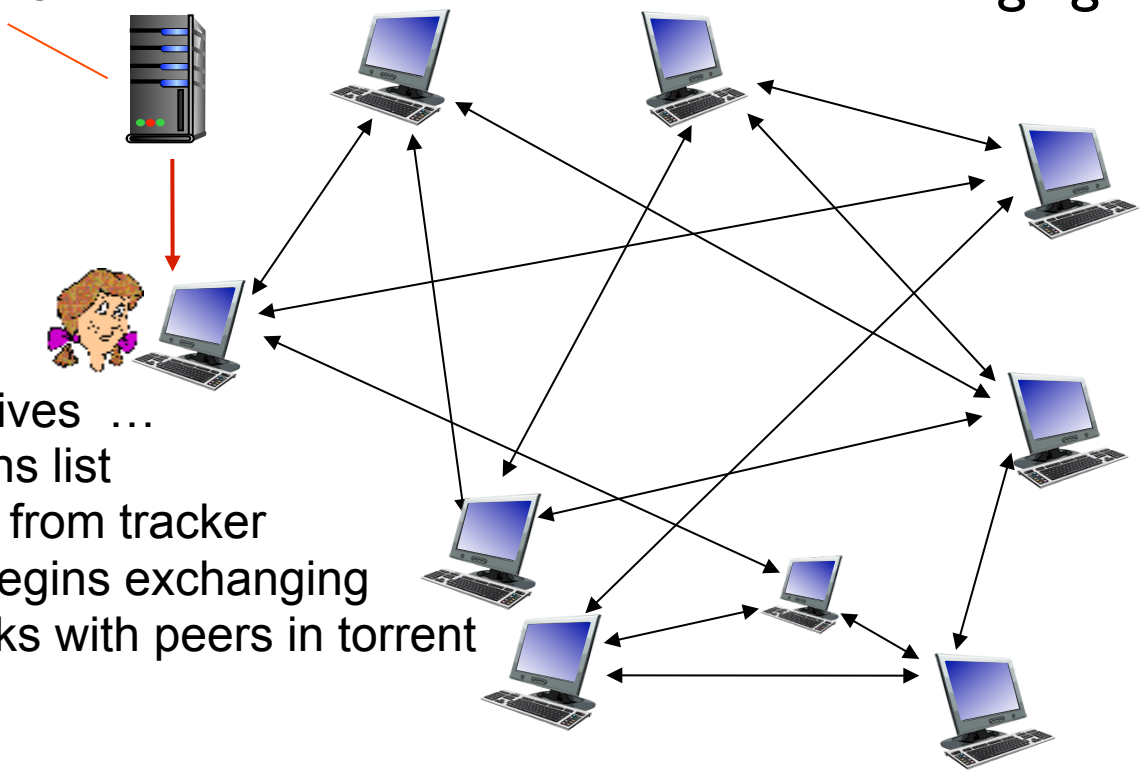


P2P file distribution: BitTorrent

- file divided into 256Kb chunks
- peers in torrent send/receive file chunks

tracker: tracks peers participating in torrent

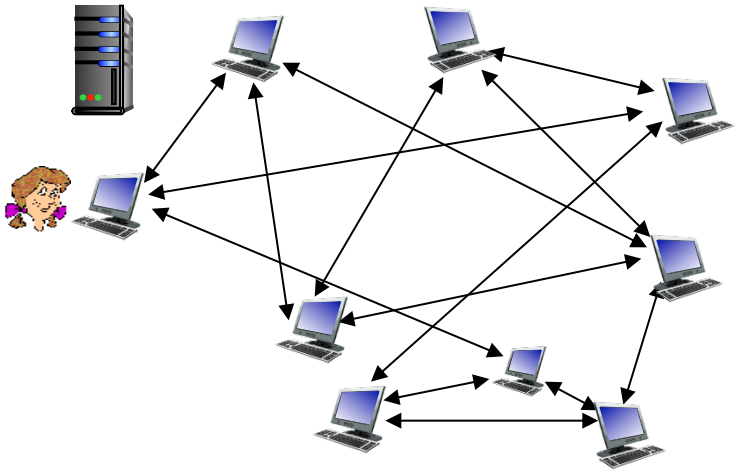
torrent: group of peers exchanging chunks of a file



Alice arrives …
… obtains list
of peers from tracker
… and begins exchanging
file chunks with peers in torrent

P2P file distribution: BitTorrent

- peer joining torrent:
  - has no chunks, but will accumulate them over time from other peers
  - registers with tracker to get list of peers, connects to subset of peers ("neighbors")



- while downloading, peer uploads chunks to other peers
- peer may change peers with whom it exchanges chunks
- *churn:* peers may come and go
- once peer has entire file, it may (selfishly) leave or (altruistically) remain in torrent

Hybrid of client-server and P2P

- File transfer P2P
- File search centralized:
  - Peers register content at central server
  - Peers query same central server to locate content

Instant messaging

- Chatting between two users is P2P
- Presence detection/location centralized:
  - User registers its IP address with central server when it comes online
  - User contacts central server to find IP addresses of buddies

Processes communicating

Process: program running within a host.

- within same host, two processes communicate using inter-process communication (defined by OS).
- processes in different hosts communicate by exchanging messages

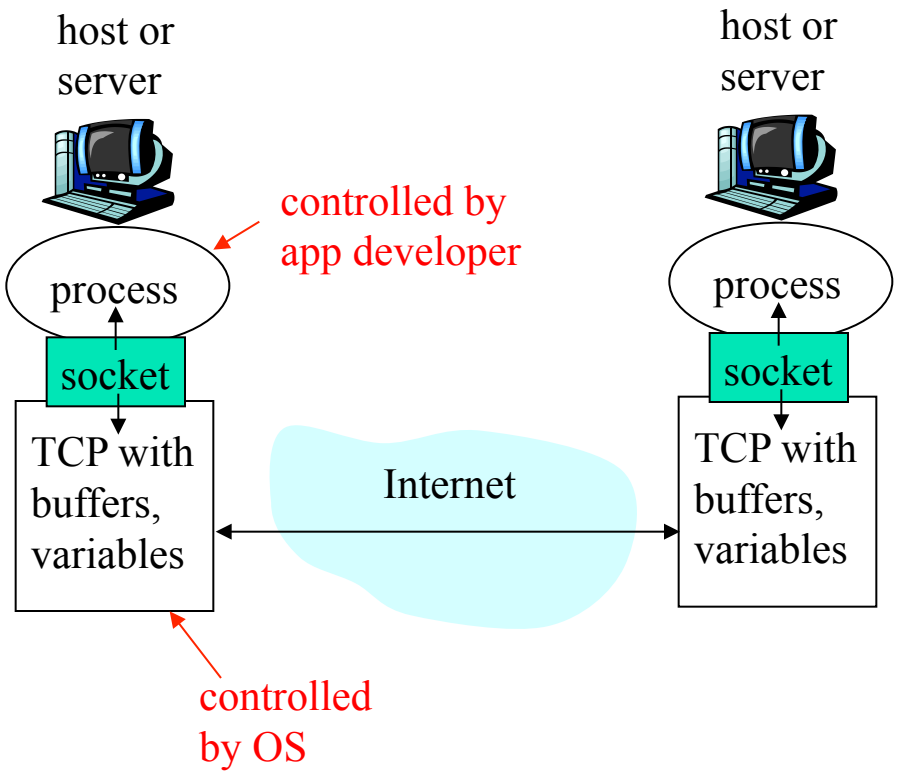
Client process: process that initiates communication

Server process: process that waits to be contacted

- Note: applications with P2P architectures have client processes & server processes

Sockets

- process sends/receives messages to/from its socket
- socket analogous to door
  - sending process shoves message out door
  - sending process relies on transport infrastructure on other side of door which brings message to socket at receiving process



Addressing processes

- For a process to receive messages, it must have an identifier
- A host has a unique 32-bit IP address
- Q: does the IP address of the host on which the process runs suffice for identifying the process?
- Answer: No, many processes can be running on same host

- Identifier includes both the IP address and port numbers associated with the process on the host.
- Example port numbers:
  - HTTP server: 80
  - Mail server: 25

What transport service does an app need?

Data loss

- some apps (e.g., audio) can tolerate some loss
- other apps (e.g., file transfer, telnet) require 100% reliable data transfer

Timing

- some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"

Bandwidth

- some apps (e.g., multimedia) require minimum amount of bandwidth to be "effective"
- other apps ("elastic apps") make use of whatever bandwidth they get

Transport service requirements of common apps

Application	Data loss	Bandwidth	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video:10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
instant messaging	no loss	elastic	yes and no

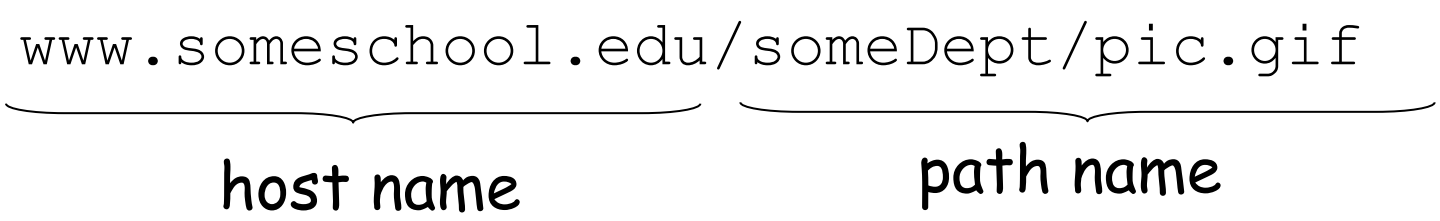
Internet apps: application, transport protocols

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	proprietary (e.g. RealNetworks)	TCP or UDP
Internet telephony	proprietary (e.g., Dialpad)	typically UDP

Web and HTTP

First some jargon

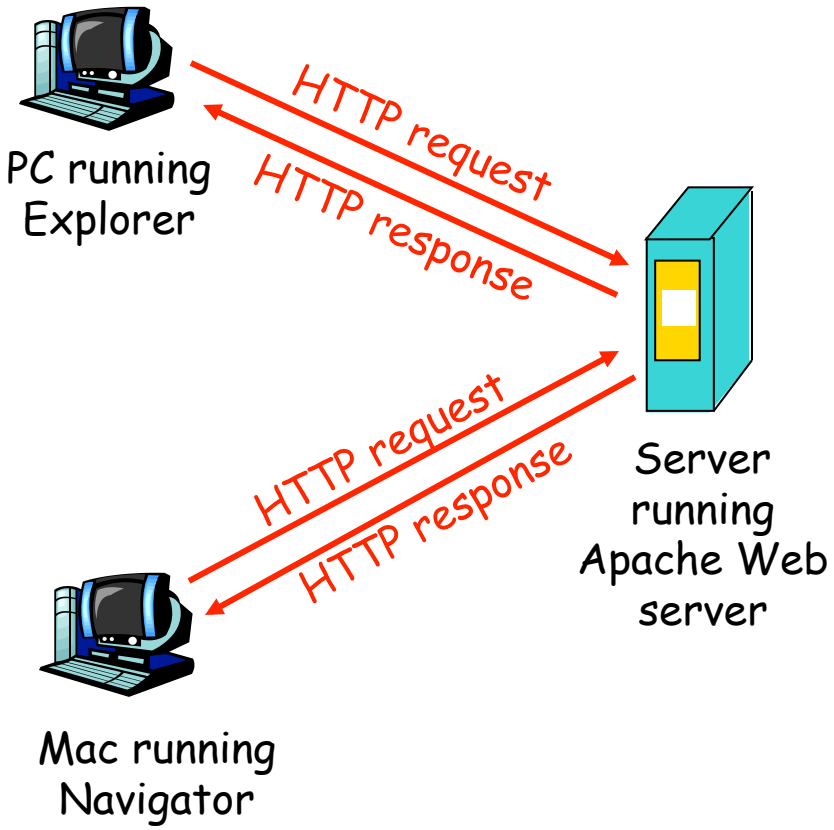
- Web page consists of objects
- Object can be HTML file, JPEG image, Java applet, audio file,…
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL
- Example URL:



HTTP overview

HTTP: hypertext transfer protocol

- 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 1.0: RFC 1945
- HTTP 1.1: RFC 2068



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

aside

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

Nonpersistent HTTP

- At most one object is sent over a TCP connection.
- HTTP/1.0 uses nonpersistent HTTP

Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server.
- HTTP/1.1 uses persistent connections in default mode

HTTP request message

- ASCII (human-readable format)

request line (GET, POST, HEAD commands)

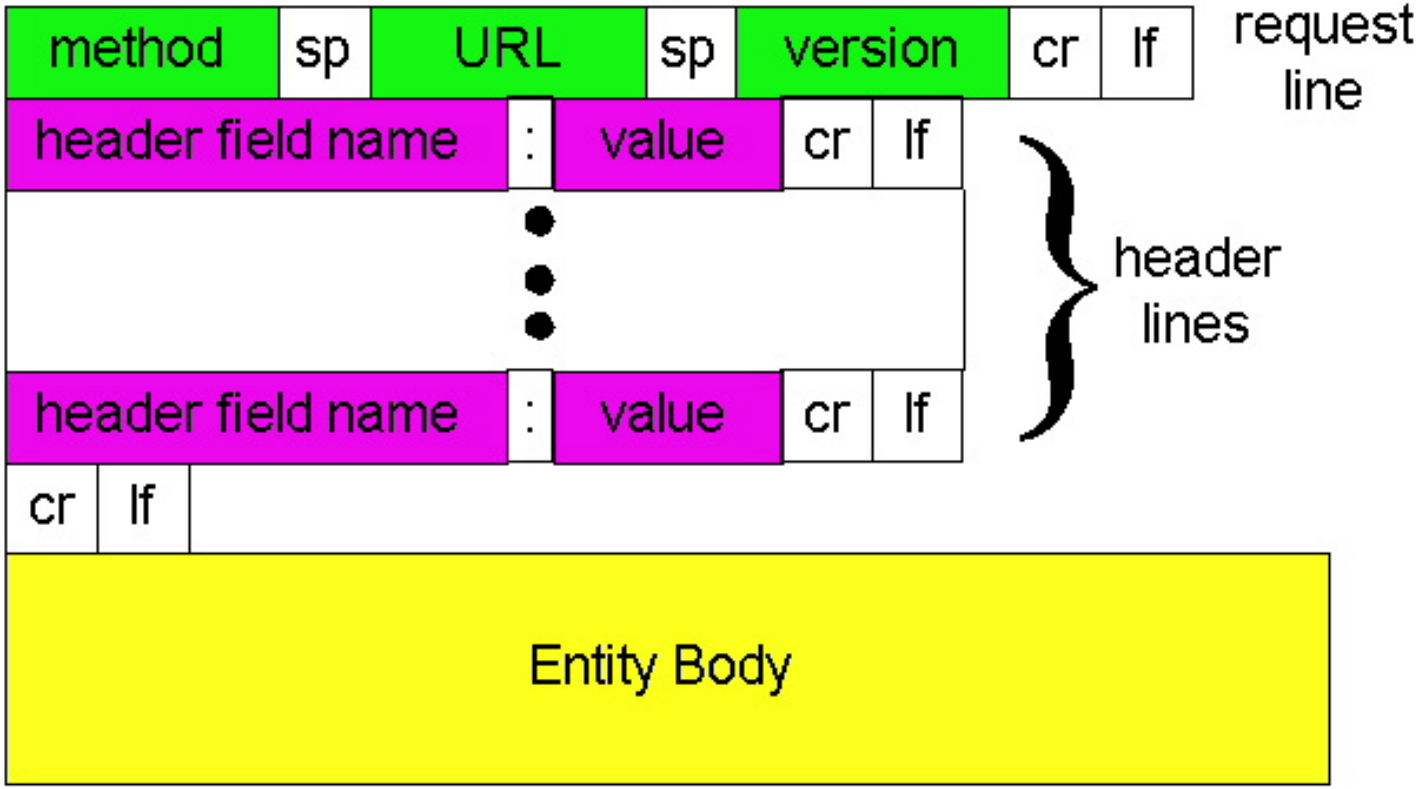
header lines

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

Carriage return, line feed indicates end of message

(extra carriage return, line feed)

HTTP request message: general format



Method types

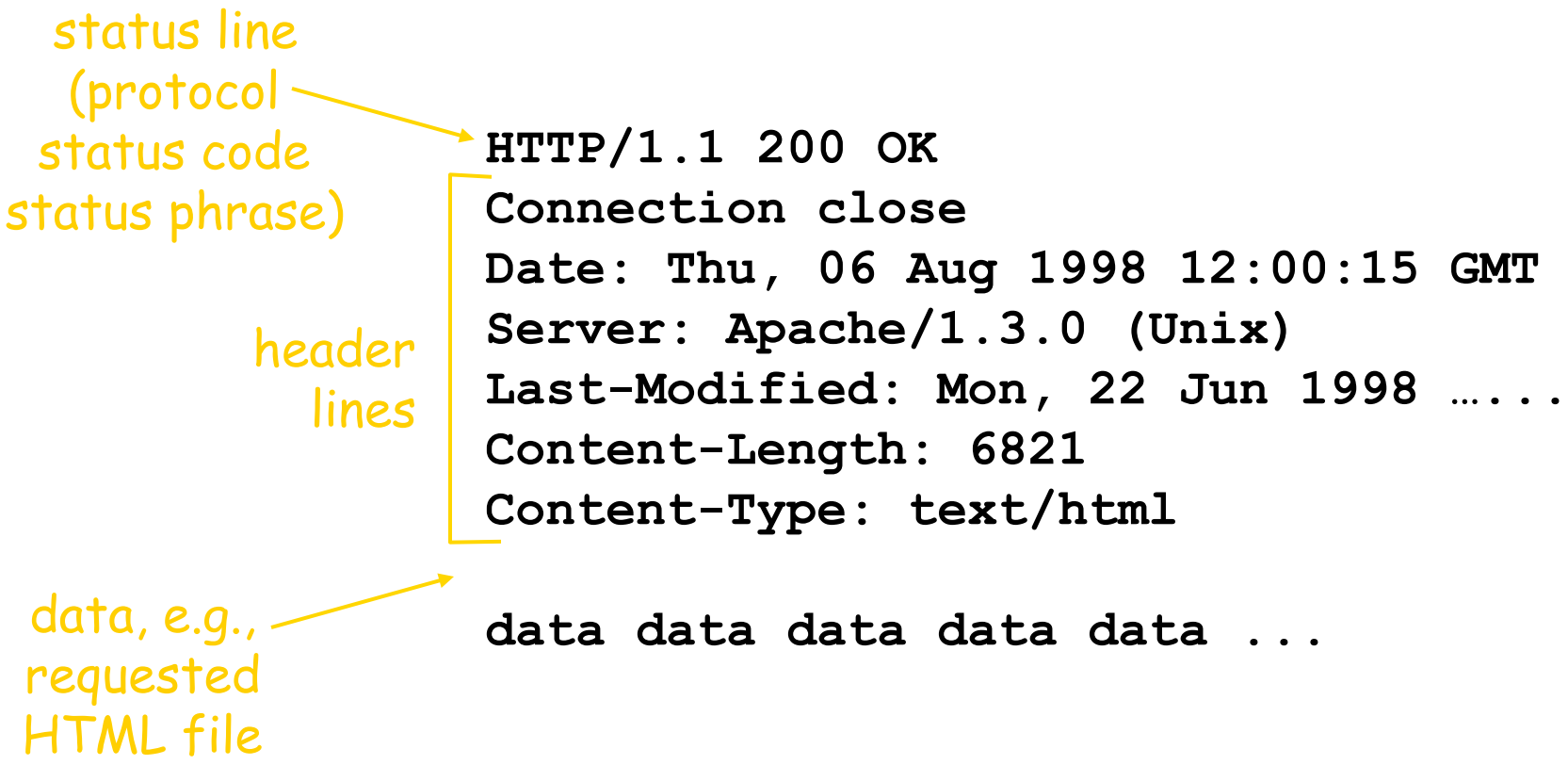
HTTP/1.0

- GET
- POST
- HEAD
  - asks server to leave requested object out of response

HTTP/1.1

- GET, POST, HEAD
- PUT
  - uploads file in entity body to path specified in URL field
- DELETE
  - deletes file specified in the URL field

HTTP response message



HTTP response status codes

In first line in server->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

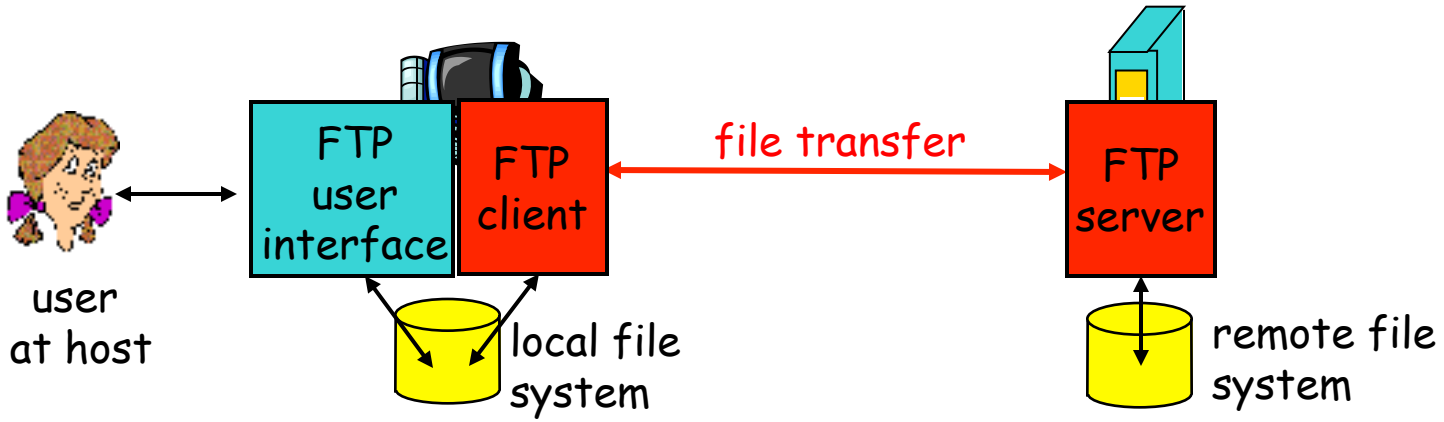
- request message not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

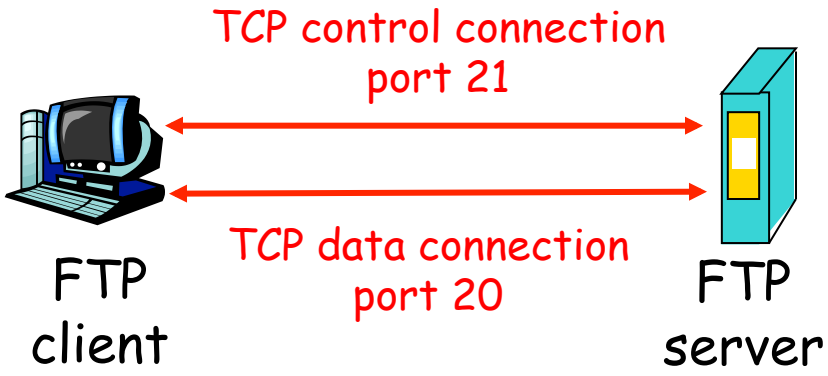
FTP: the file transfer protocol



- transfer file to/from remote host
- client/server model
  - *client:* side that initiates transfer (either to/from remote)
  - *server:* remote host
- ftp: RFC 959
- ftp server: port 21

FTP: separate control, data connections

- FTP client contacts FTP server at port 21, specifying TCP as transport protocol
- Client obtains authorization over control connection
- Client browses remote directory by sending commands over control connection.
- When server receives a command for a file transfer, the server opens a TCP data connection to client
- After transferring one file, server closes connection.



- Server opens a second TCP data connection to transfer another file.
- Control connection: "out of band"
- FTP server maintains "state": current directory, earlier authentication

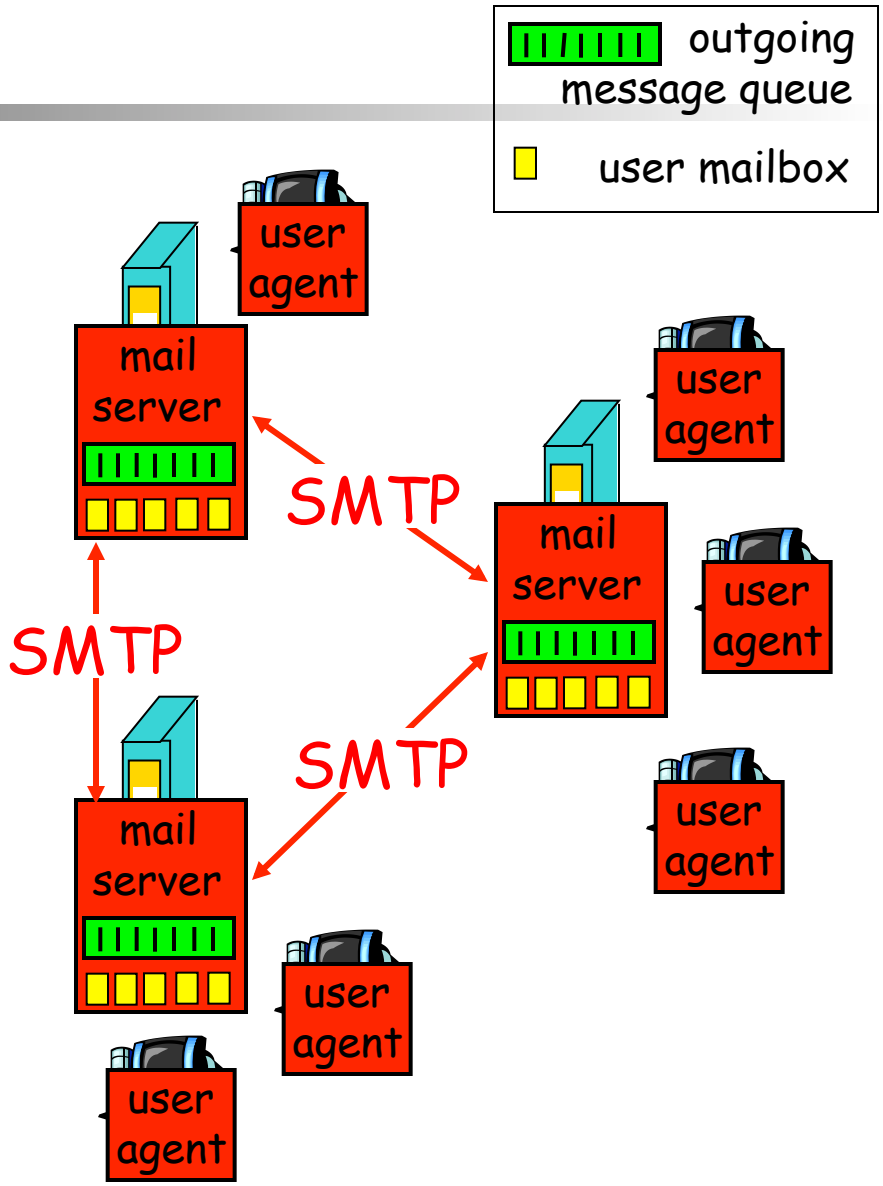
Electronic Mail

Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User Agent

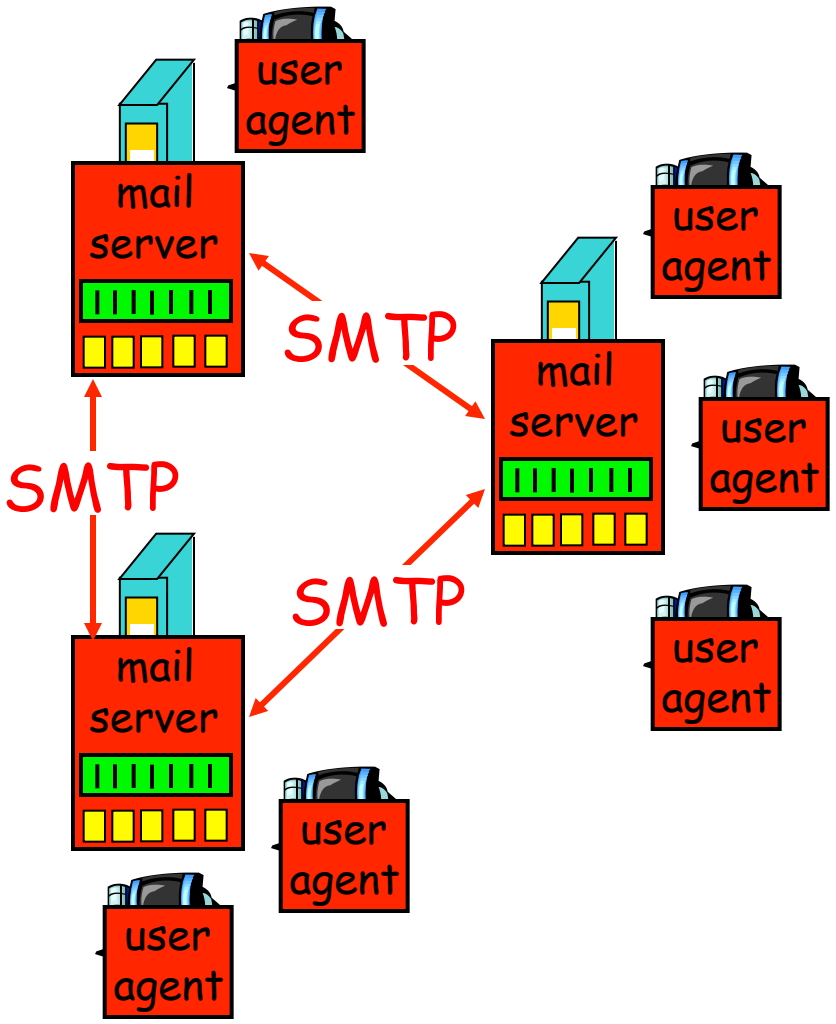
- "mail reader"
- composing, editing, reading mail messages
- e.g., Eudora, Outlook, elm, Netscape Messenger
- outgoing, incoming messages stored on server



Electronic Mail: mail servers

Mail Servers

- mailbox contains incoming messages for user
- message queue of outgoing (to be sent) mail messages
- SMTP protocol between mail servers to send email messages
  - client: sending mail server
  - "server": receiving mail server

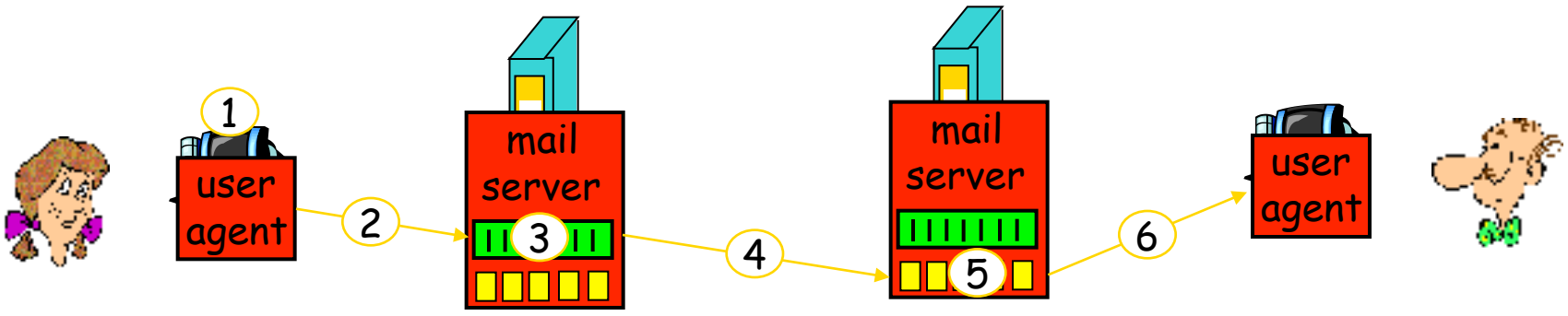


Electronic Mail: SMTP [RFC 2821]

- uses TCP to reliably transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer
  - handshaking (greeting)
  - transfer of messages
  - closure
- command/response interaction
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII

Scenario: Alice sends message to Bob

1) Alice uses UA to compose message and "to" `bob@someschool.edu`
2) Alice's UA sends message to her mail server; message placed in message queue
3) Client side of SMTP opens TCP connection with Bob's mail server
4) SMTP client sends Alice's message over the TCP connection
5) Bob's mail server places the message in Bob's mailbox
6) Bob invokes his user agent to read message



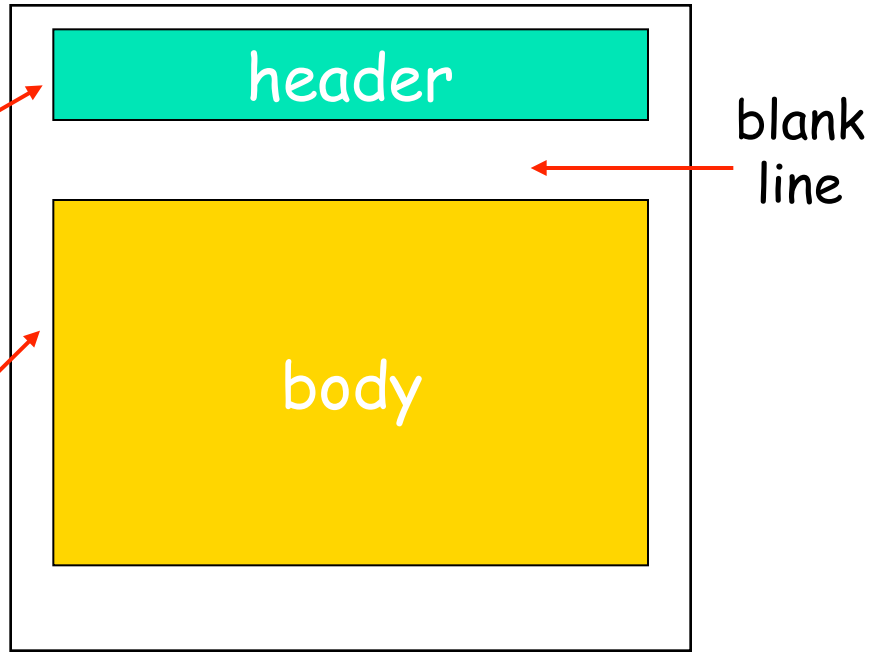
Mail message format

SMTP: protocol for exchanging email msgs

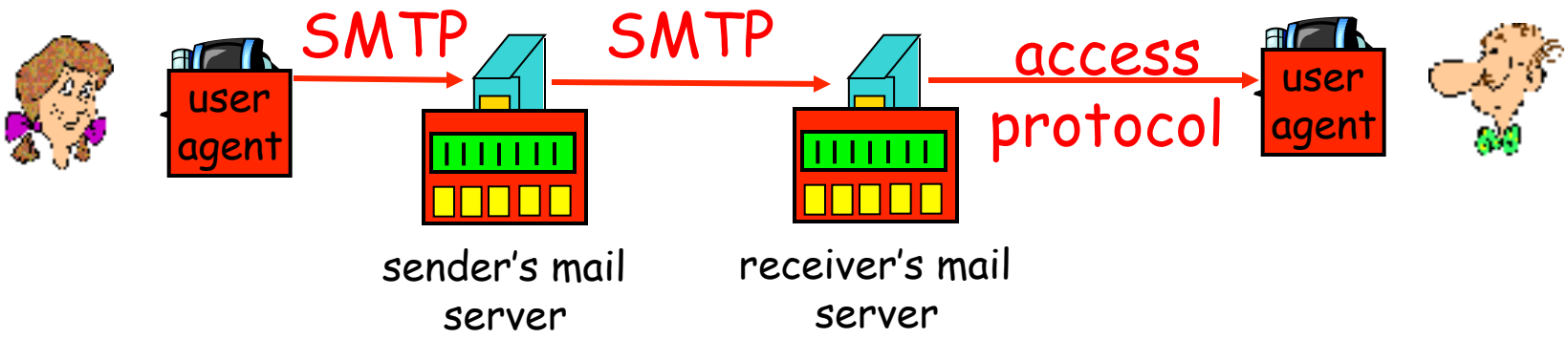
RFC 822: standard for text message format:

- header lines, e.g.,
  - To:
  - From:
  - Subject:

  different from SMTP commands!
- body
  - the "message", ASCII characters only



Mail access protocols



- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: Post Office Protocol [RFC 1939]
    - authorization (agent <-->server) and download
  - IMAP: Internet Mail Access Protocol [RFC 1730]
    - more features (more complex)
    - manipulation of stored msgs on server
  - HTTP: Hotmail , Yahoo! Mail, etc.

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name", e.g., ww.yahoo.com - used by humans

Q: map between IP addresses and name ?

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

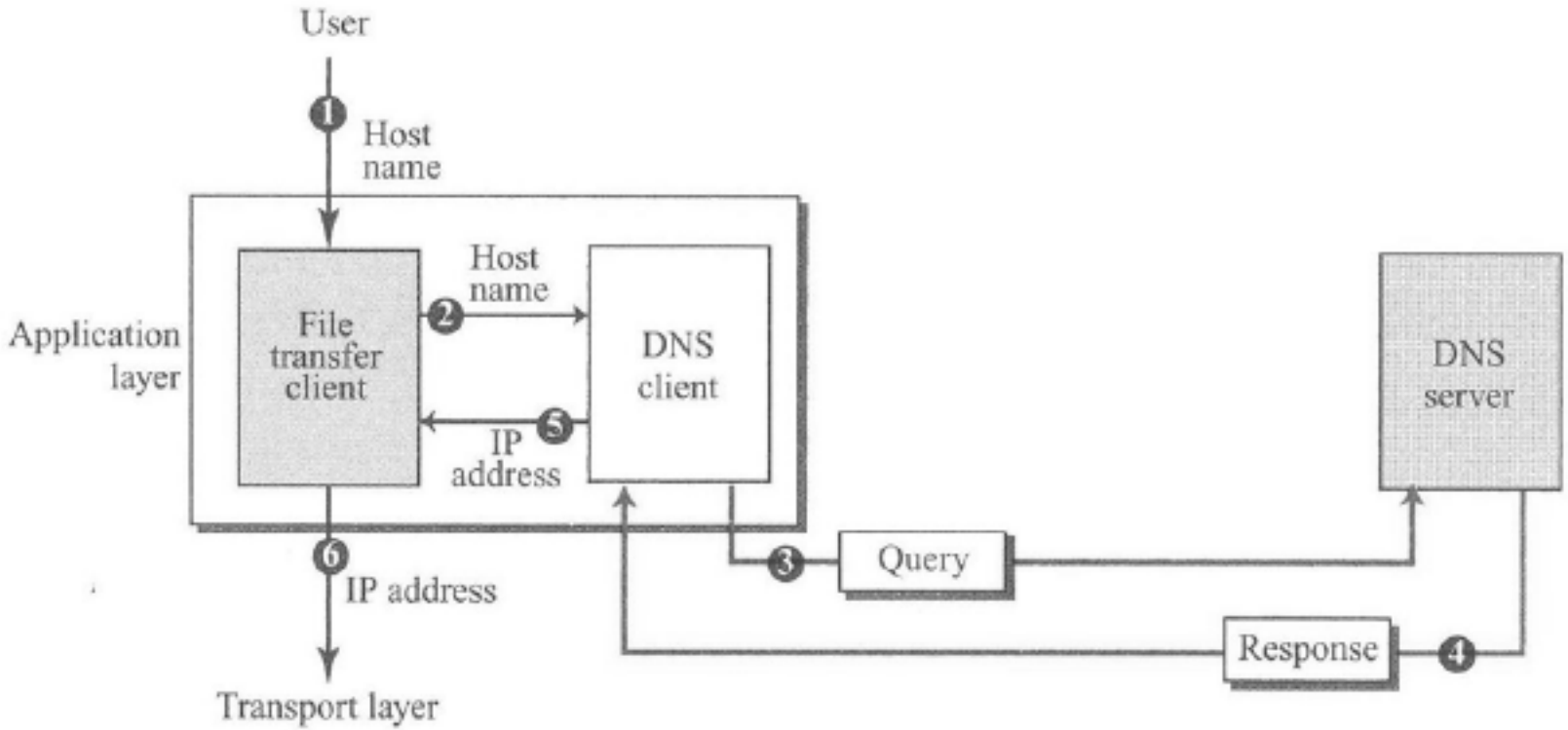
- IP address (32 bit) - used for addressing datagrams
- "name", e.g., ww.yahoo.com - used by humans

Q: map between IP addresses and name ?

Domain Name System:

- *distributed database* implemented in hierarchy of many *name servers*
- *application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

DNS Client


User
1
Host name
Application layer
File transfer client
Host name
2
DNS client
5
IP address
DNS server
6
IP address
Transport layer
3
Query
Response
4

DNS Client and Server

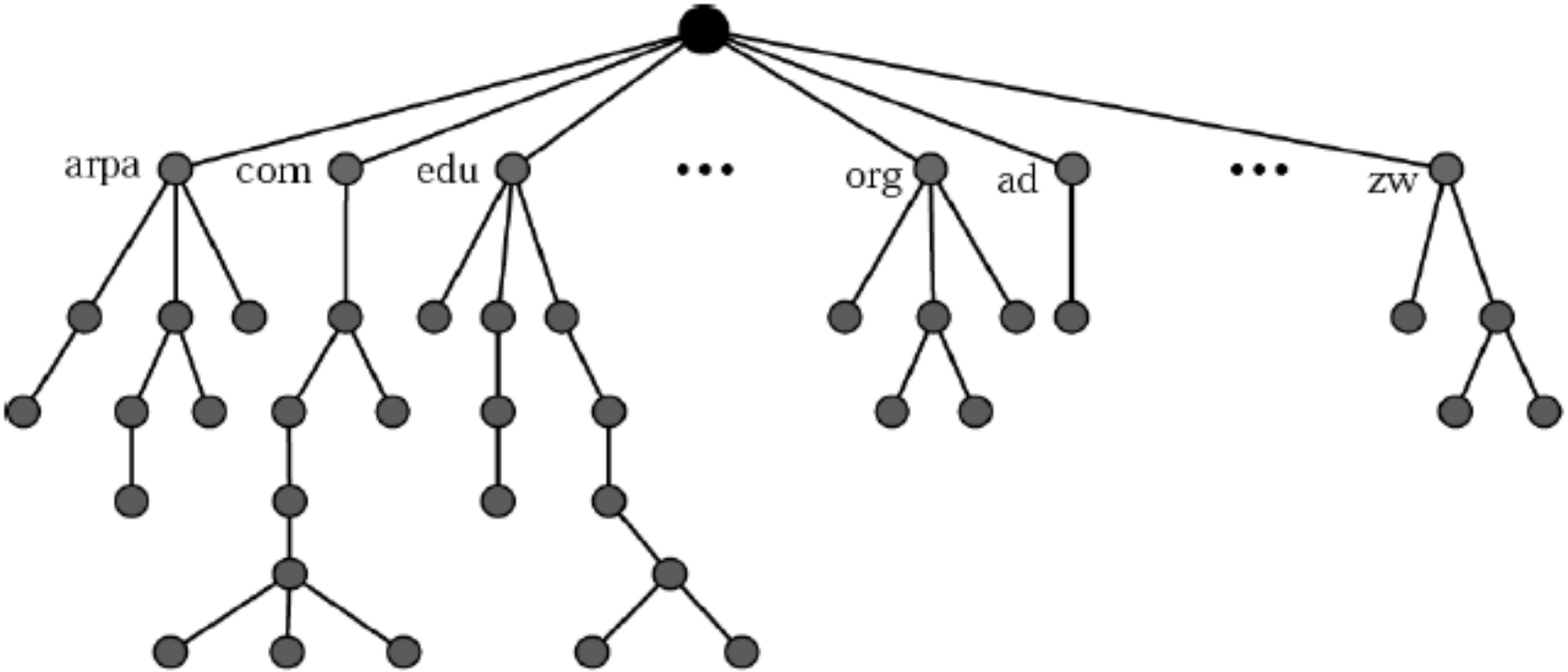
- In the figure:
  1. The user passes the host name to the file transfer client
  2. The file transfer client passes the host name to the DNS client
  3. The DNS client sends a message to a DNS server with a query.
  4. The DNS server responds with the IP address of the desired file transfer server
  5. The DNS client passes the IP address to the file transfer client
  6. The file transfer client uses the received address to access the file transfer server.

Name Space

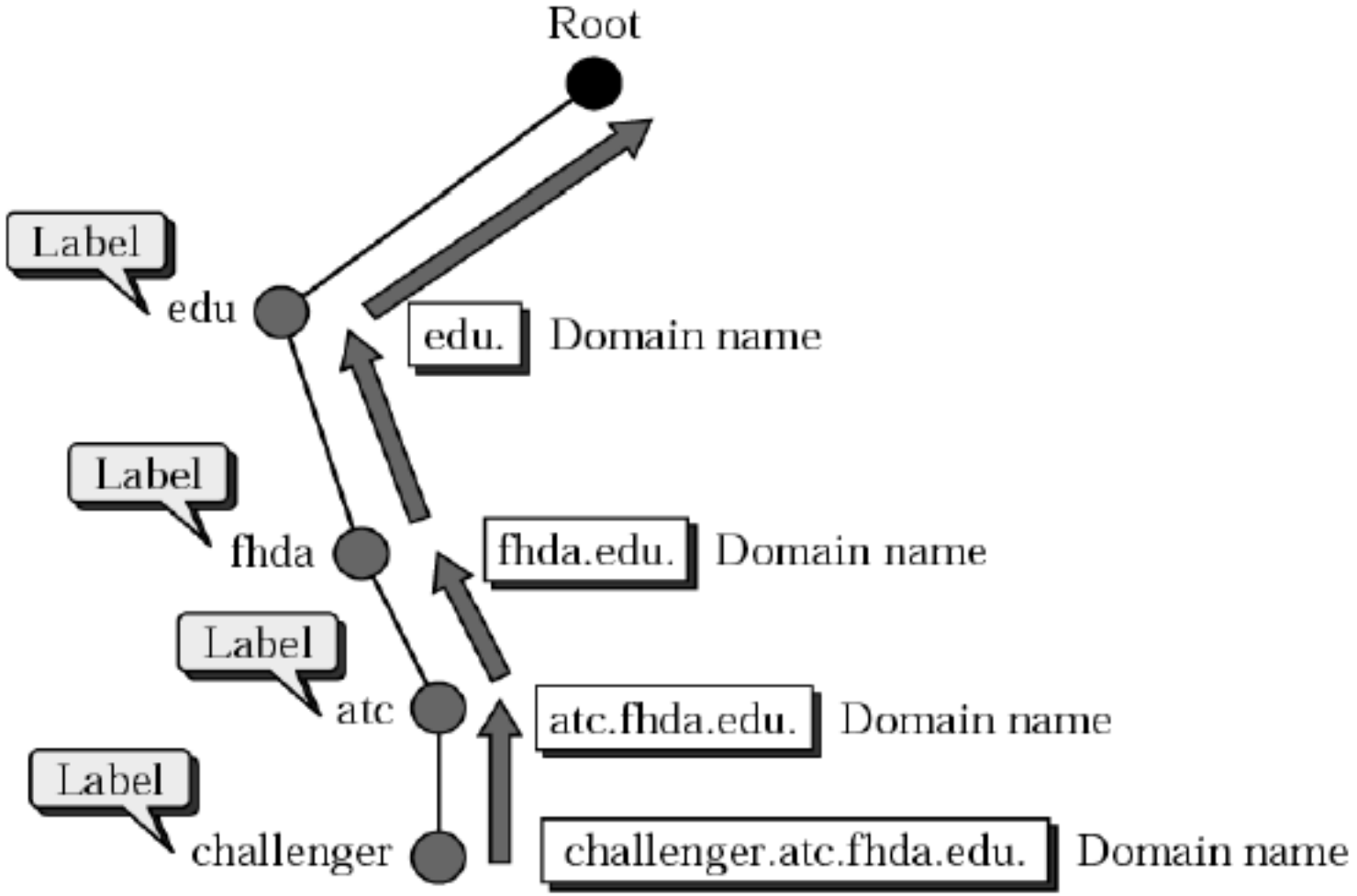
- The names assigned to machines must be unique.
- A name space can be organized in 2 ways:
  - Flat name space
    - A name is a sequence of characters without a structure
    - Common sections between names have no meaning
    - It cannot be used in a large system because it is not possible to centrally control the whole Internet
  - Hierarchical name space
    - Names are made of several parts
    - The authority to assign and control the name spaces can be decentralized
    - The central authority controls only part of the name
    - The organizations can assign prefixes to their hosts

Domain Name Space

- The names are defined in a tree structure.
- The tree can have 128 levels (root: level 0)

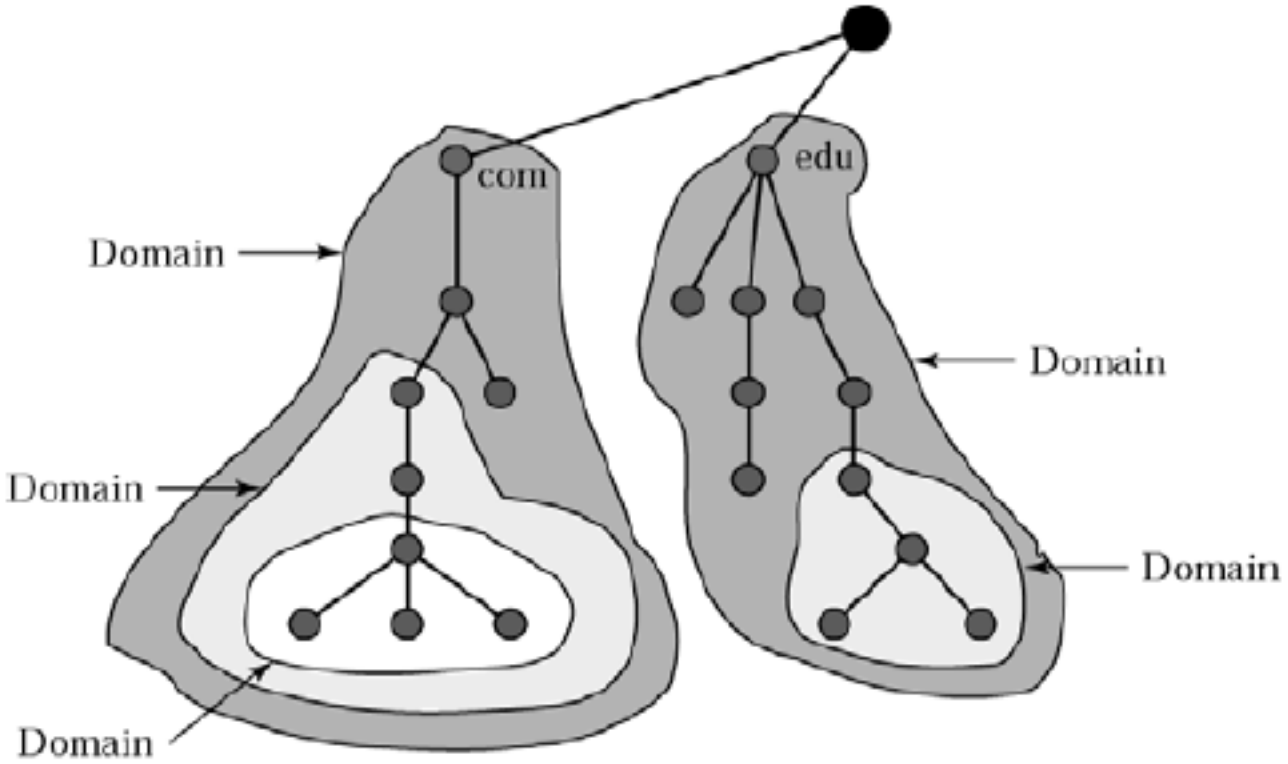


Domain Names and Labels

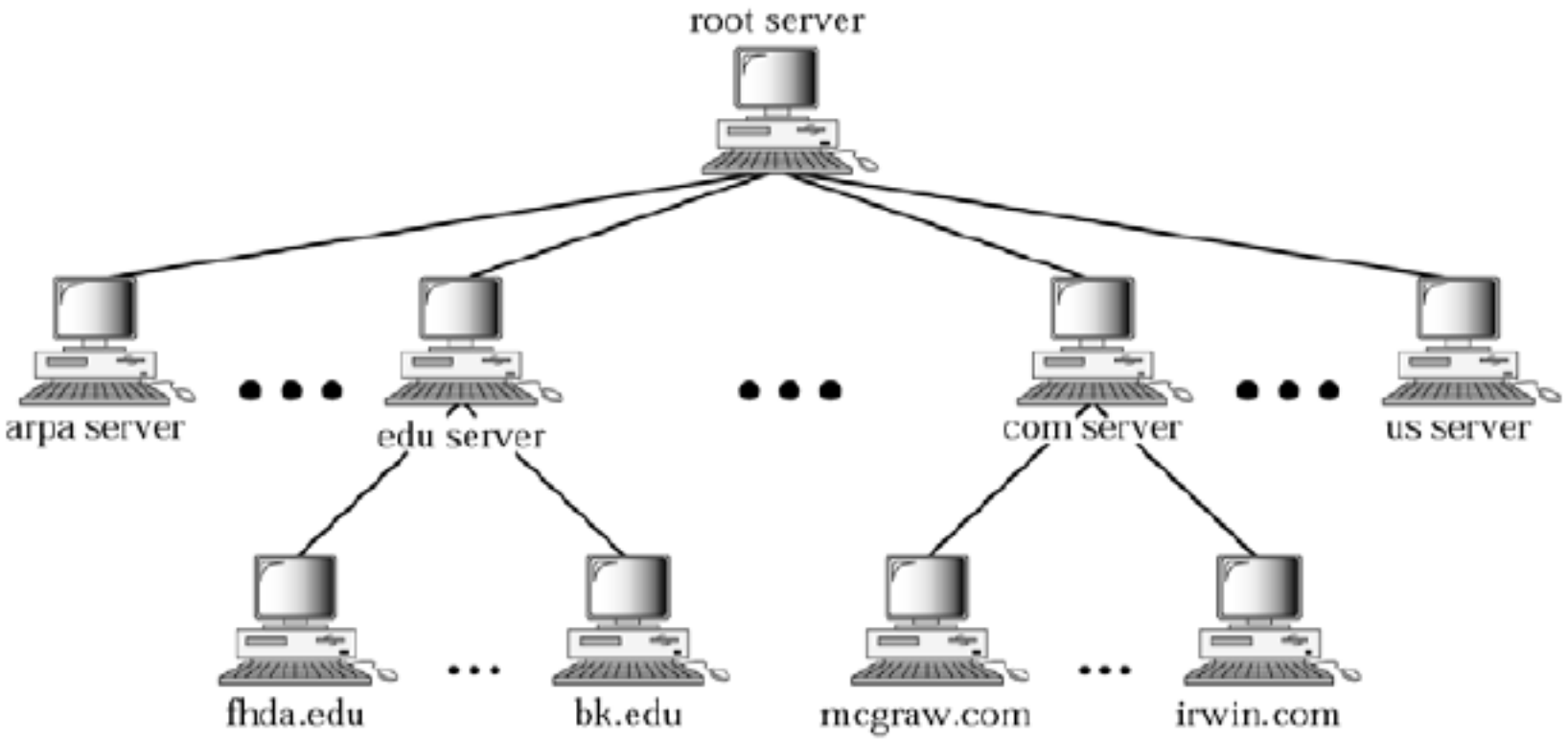


DOMAIN

- A domain is a subtree of the domain name space
  - The name of the domain is the name of the root node of the subtree

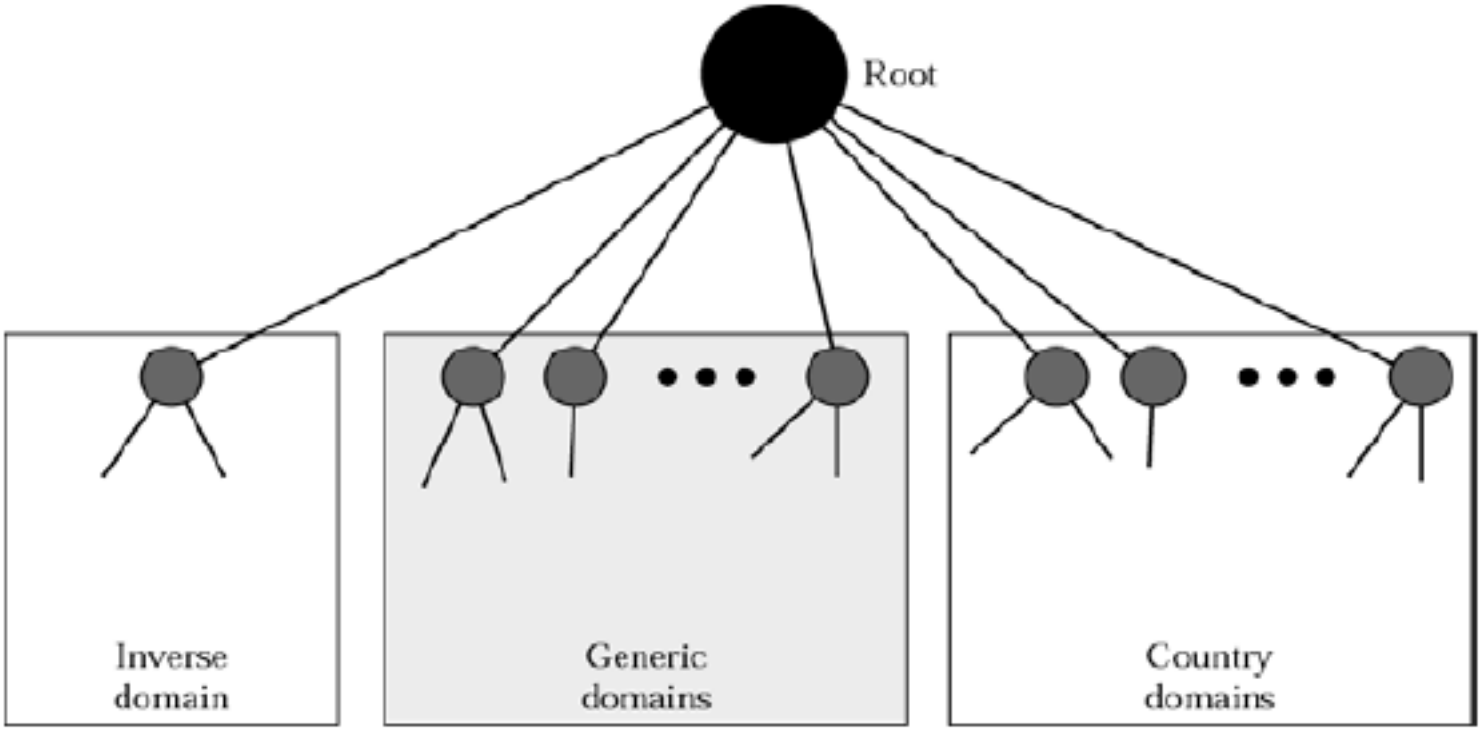


Distribution of Name Space



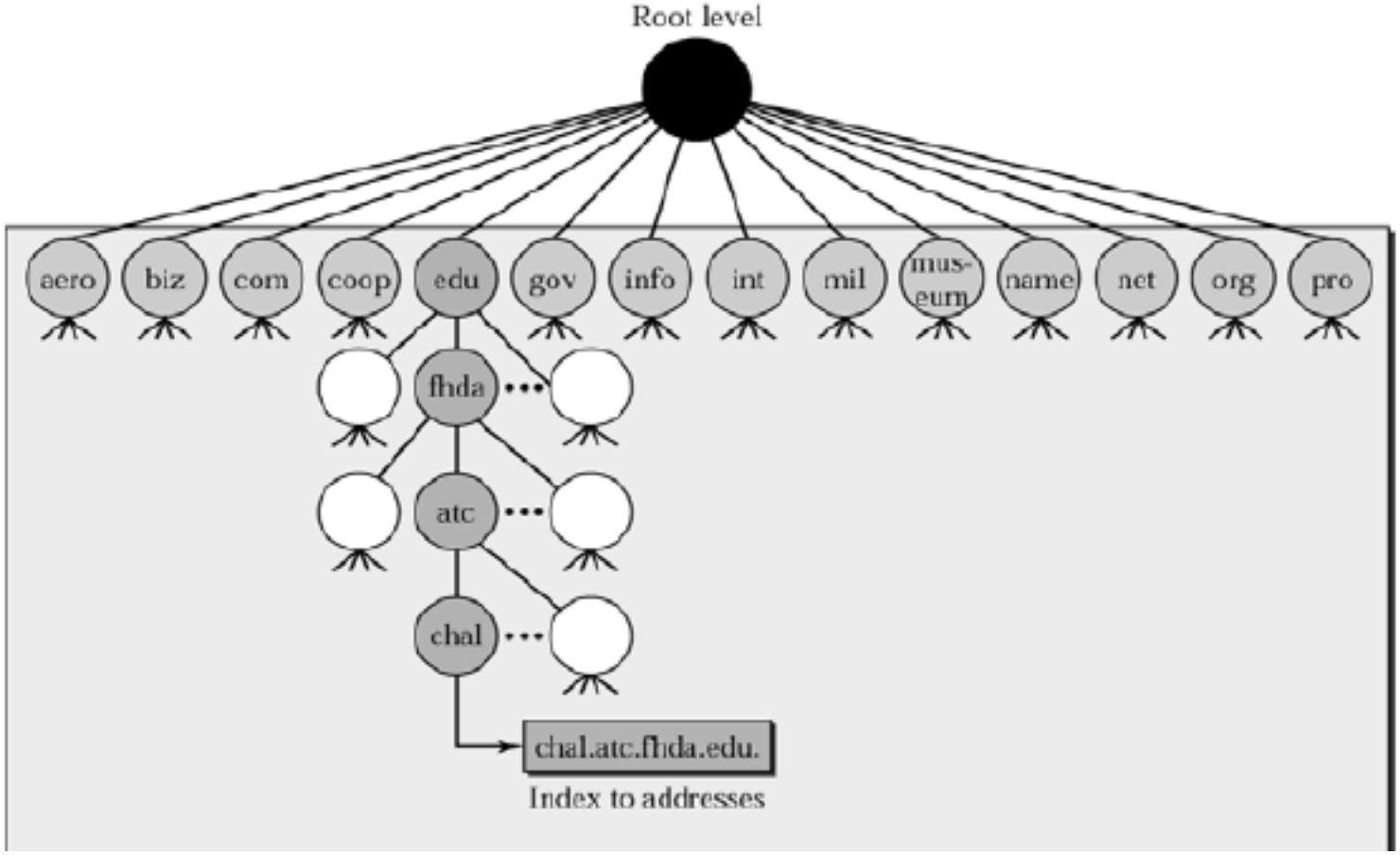
DNS in the Internet

- In the Internet the domain name space is divided into three sections.
  - Generic domains
  - Country domains
  - The Inverse domain



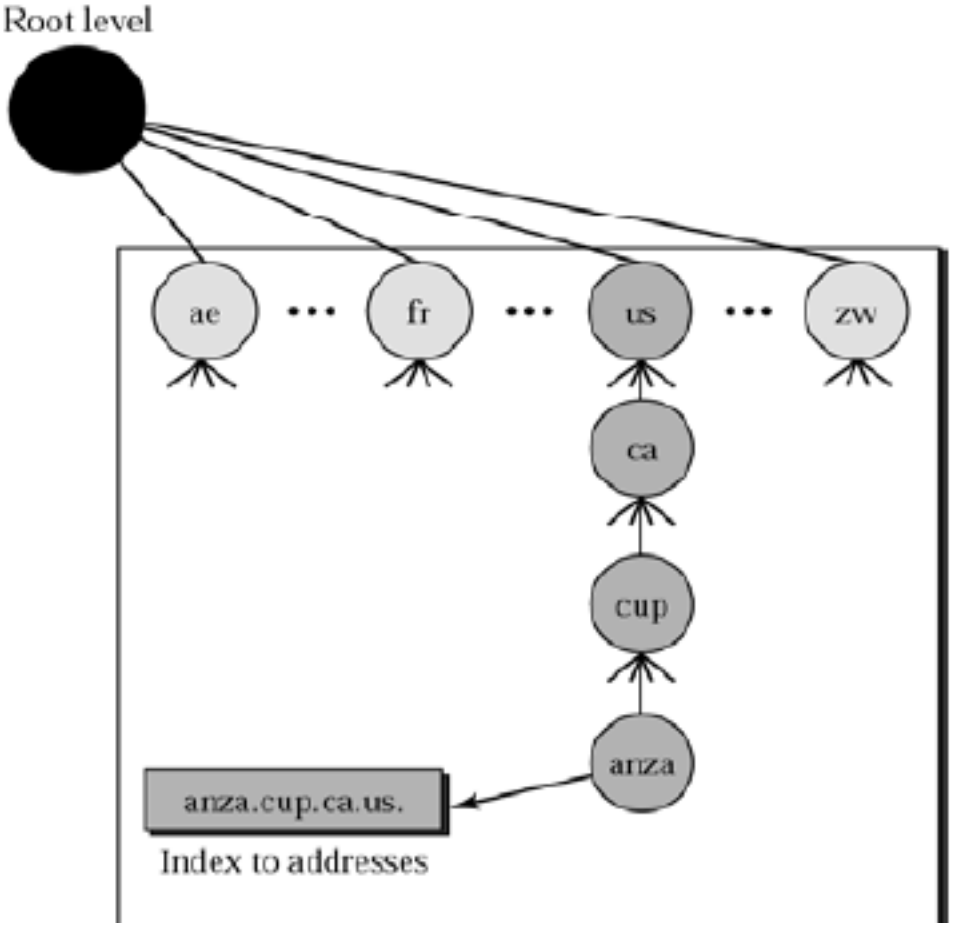
Generic Domains

- There are 14 possible labels in the first level.



Country Domains

- Uses two character country abbreviations
- The second labels can be organizational or more specific, national designations



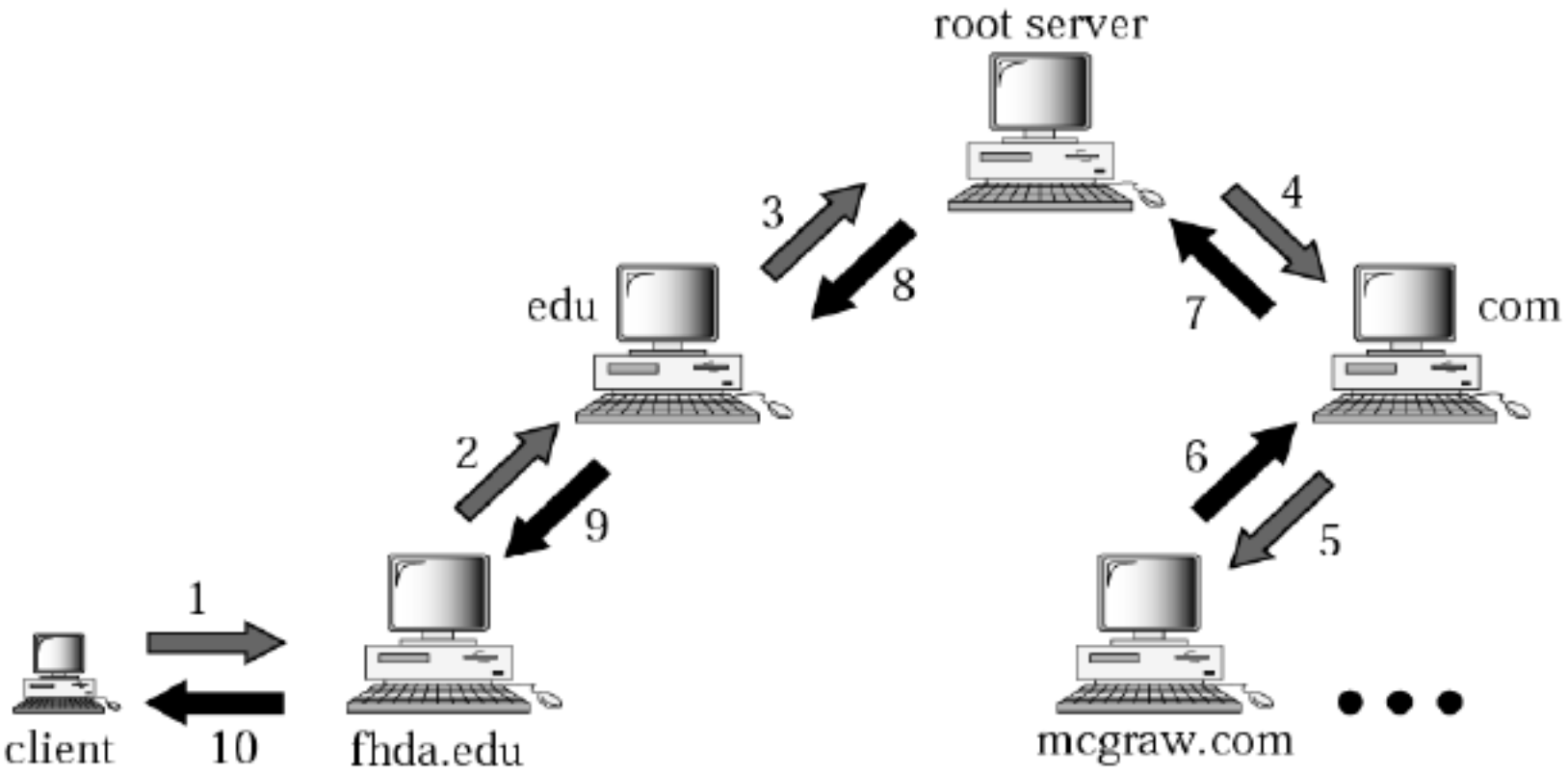
Address Resolution

- A host that needs to map a name to an address or vice versa, calls a DNS client called a resolver.
  - The resolver accesses the closest DNS server with a mapping request.
  - If the server has the information it replies with the address.
  - Otherwise, it either refers the resolver to other servers or asks other servers to provide the information.

Recursive Resolution

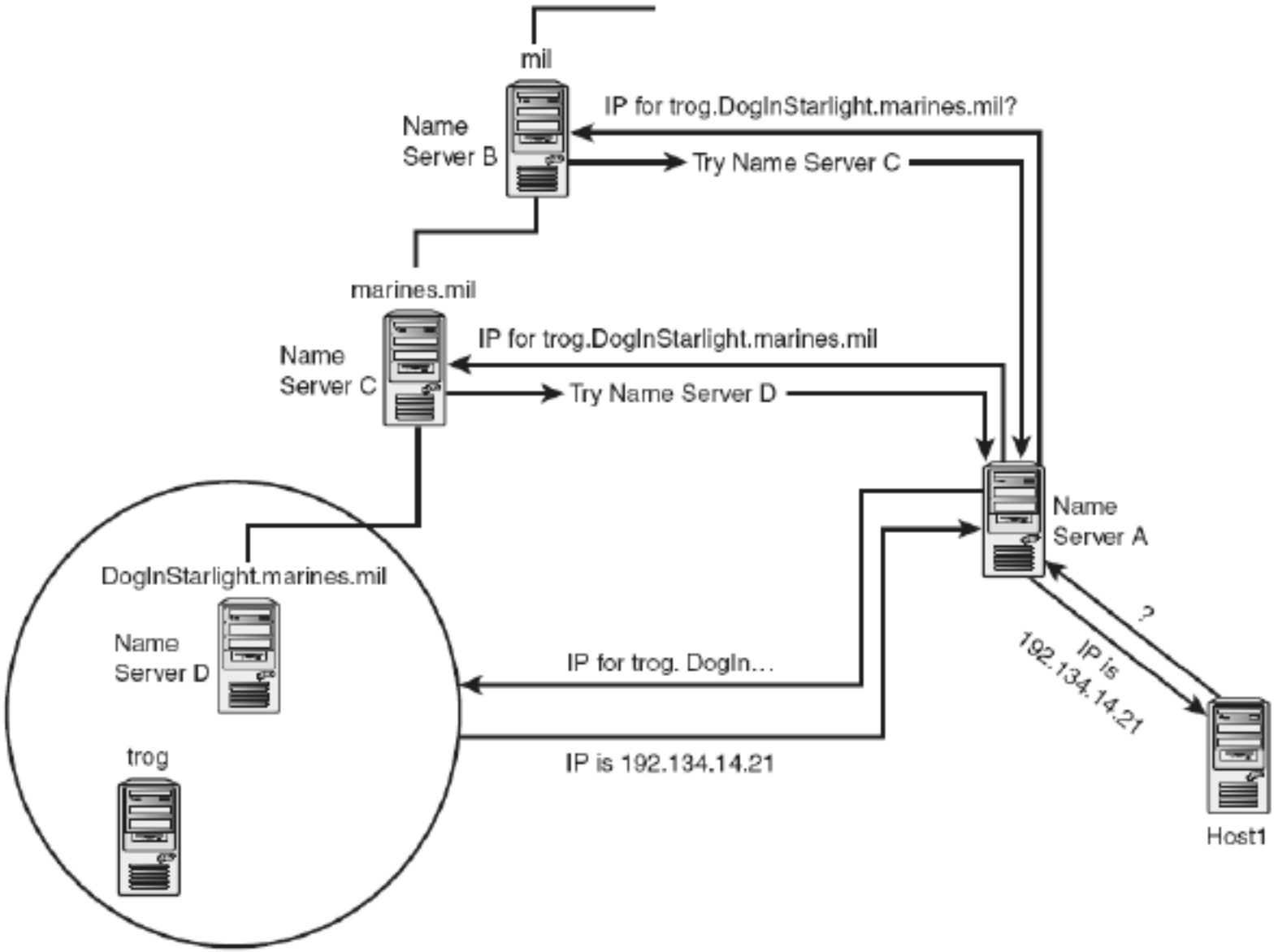
- The client can ask for a recursive answer from a name server.
- This means that the resolver expects the server to supply the final answer.
  - If the server is the authority for the domain name, it checks its database and responds.
  - If the server is not the authority, it sends the request to another server, and waits for the response.
  - This goes until the query is resolved.
  - Then, the response travels back to the requesting client.

Recursive Resolution



Iterative Resolution

- Host1 sends a query to name server A asking for the IP address associated with the domain name trog.DogInStarlight.marines.mil.
- Name server A checks its own records to see if it has the requested address.
  If server A has the address, it returns the address to Host1.
- If name server A does not have the address, it initiates the process of finding the address.
  Name server A sends an iterative request for the address to name server B, a top-level name server for the .mil domain, asking for the address associated with the name trog.DogInStarlight.marines.mil.
- Name server B is not able to supply the address, but it is able to send name server A the address of name server C, the name server for marines.mil.


mil
Name
Server B
IP for trog.DogInStarlight.marines.mil?
Try Name Server C
marines.mil
Name
Server C
IP for trog.DogInStarlight.marines.mil
Try Name Server D
Name
Server A
DogInStarlight.marines.mil
Name
Server D
IP for trog. DogIn...
?
IP is
192.134.14.21
trog
IP is 192.134.14.21
Host1

Iterative Resolution

- Name server A sends a request for the address to name server C.
  Name server C is not able to supply the address, but it is able to send the address of name server D, the name server for DogInStarlight.marines.com.
- Name server A sends a request for the IP address to name server D.
  Name server D looks up the address for the host trog.DogInStarlight.marines.mil and sends the address to name server A.
  Name server A then sends the address to Host1.
- Host1 initiates a connection to the host trog.DogInStarlight.marines.mil.