

UNIT - IV

Application Layer: DNS, DDNS, TELNET, EMAIL, FTP, WWW, HTTP, SNMP, Bluetooth, Firewalls

Network Security: Electronic mail, directory services and network management, Basic concepts of Cryptography

Application layer: overview

1. **WEB & HTTP**
2. FTP
3. E-mail
4. DNS - The Internet's Directory Service
5. DDNS
6. TELNET
7. BLUETOOTH
8. FIREWALLS
9. SNMP

1. The Web and HTTP

- 1.1 Overview of HTTP
- 1.2 Non-Persistent and Persistent Connections
- 1.3 HTTP Message Format
- 1.4 Cookies
- 1.5 Web Caching

What is HTTP? (HyperText Transfer Protocol)

- **HTTP (HyperText Transfer Protocol)** is the foundation of communication on the World Wide Web.
- It is a protocol used for transmitting hypertext (HTML documents, images, videos, etc.) between a client (browser) and a server.

Key Features of HTTP

◆ **Stateless**

- Each HTTP request is independent; the server does not retain any information from previous requests.
- To maintain session data, cookies, sessions, or tokens are used.

◆ **Connectionless**

- The client sends a request, the server processes it and responds, and then the connection is closed.
- Persistent connections (HTTP Keep-Alive) can be used to improve efficiency.

◆ **Text-Based & Human-Readable**

- Requests and responses are sent in plain text, making debugging easy.

◆ **Supports Various Data Formats**

- Can handle HTML, JSON, XML, images, videos, etc.

HTTP vs. HTTPS

Feature	HTTP	HTTPS
Security	Not secure (data is sent in plaintext)	Secure (uses SSL/TLS encryption)
Port	80	443
Data Encryption	✗ No encryption	✓ Encrypted communication
SEO & Trust	Less trusted	Preferred by search engines & users

Web and HTTP

First, a quick review...

- web page consists of *objects*, each of which can be stored on different Web servers
- object can be HTML file, JPEG image, Java applet, audio file,...
- web page consists of *base HTML-file* which includes *several referenced objects, each* addressable by a *URL*, e.g.,

`www.someschool.edu/someDept/pic.gif`

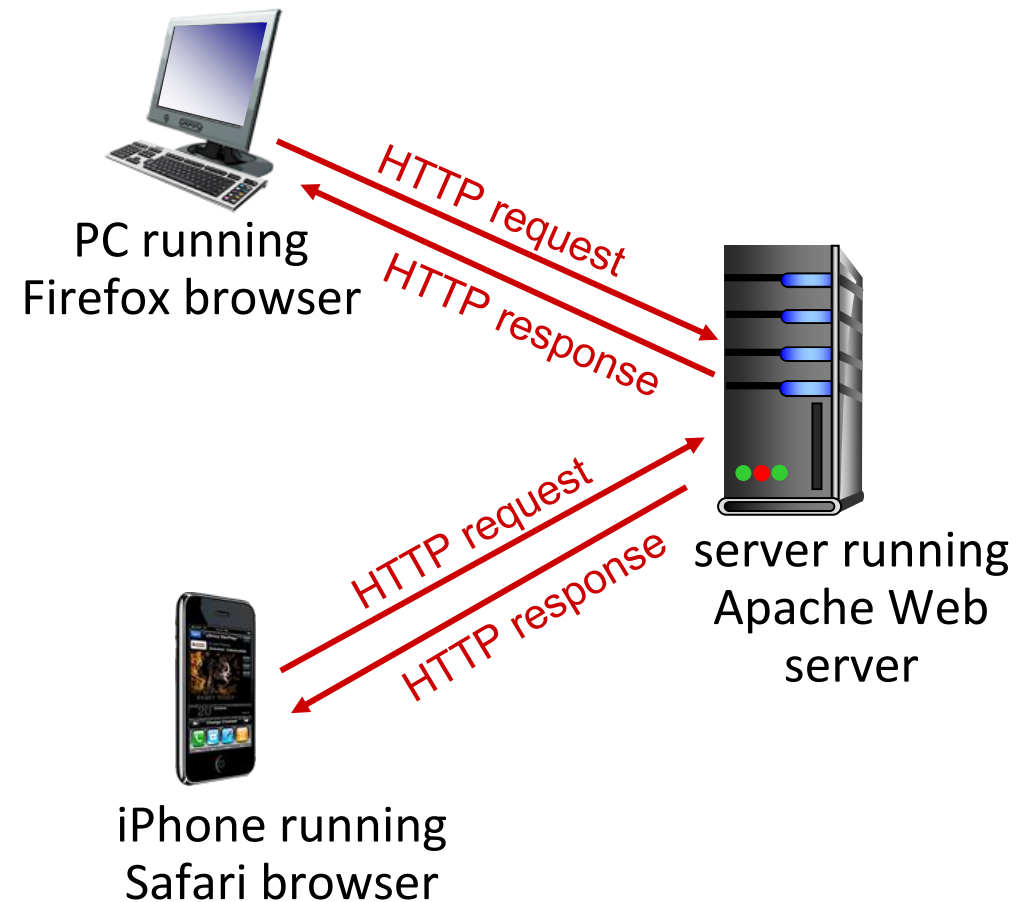
host name

path name

HTTP overview

HTTP: hypertext transfer protocol

- Web's application-layer protocol
- client/server model:
 - *client*: browser that requests, receives, (using HTTP protocol) and “displays” Web objects
 - *server*: Web server sends (using HTTP protocol) objects in response to requests



HTTP overview (continued)

HTTP 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: two types

Non-persistent HTTP

1. TCP connection opened
2. at most one object sent over TCP connection
3. TCP connection closed

downloading multiple objects required multiple connections

Persistent HTTP

- TCP connection opened to a server
- multiple objects can be sent over *single* TCP connection between client, and that server
- TCP connection closed

Non-persistent HTTP: example

User enters URL: `www.someSchool.edu/someDepartment/home.index`
(containing text, references to 10 jpeg images)



1a. HTTP client initiates TCP connection to HTTP server (process) at `www.someSchool.edu` on port 80



1b. HTTP server at host `www.someSchool.edu` waiting for TCP connection at port 80 “accepts” connection, notifying client

2. HTTP client sends 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

time



Non-persistent HTTP: example (cont.)

User enters URL: `www.someSchool.edu/someDepartment/home.index`
(containing text, references to 10 jpeg images)



5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

4. HTTP server closes TCP connection.

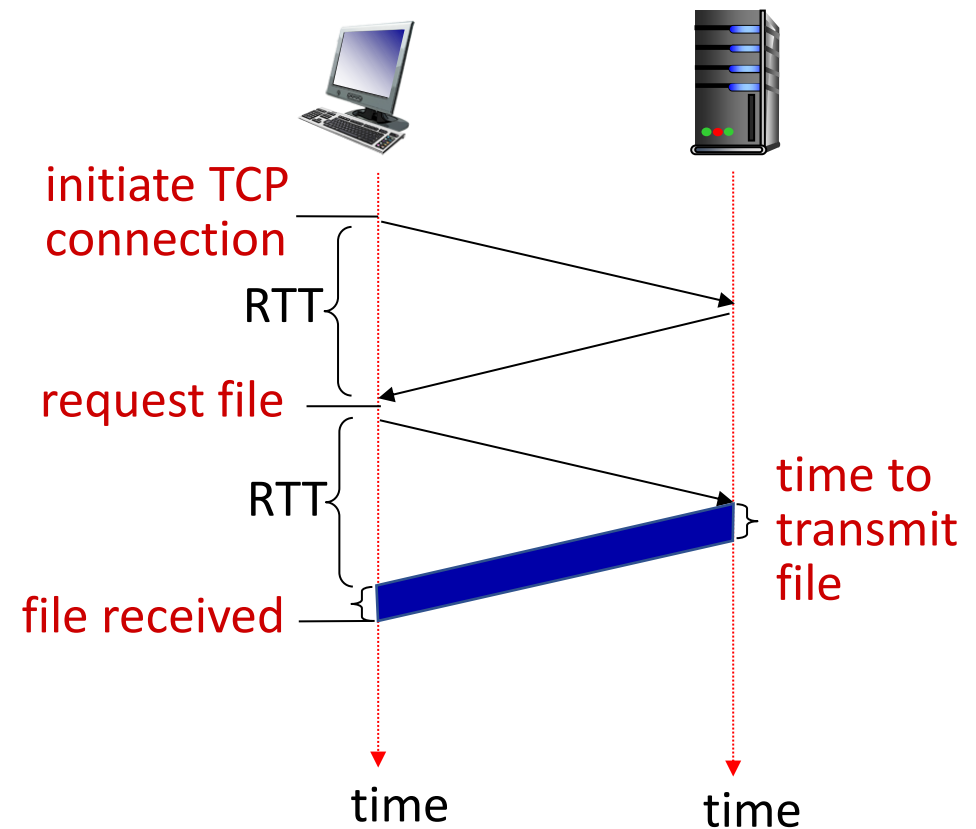


Non-persistent HTTP: response time

RTT (definition): time for a small packet to travel from client to server and back

HTTP response time (per object):

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- object/file transmission time



Non-persistent HTTP response time = $2RTT + \text{file transmission time}$

Persistent HTTP (HTTP 1.1)

Non-persistent HTTP issues:

- requires 2 RTTs per object
- OS overhead for *each* TCP connection
- browsers often open multiple parallel TCP connections to fetch referenced objects in parallel

Persistent HTTP (HTTP1.1):

- 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 the referenced objects (cutting response time in half)

HTTP Request Structure

- An HTTP request consists of:
 - i. **Request Line** – Specifies method, resource, and HTTP version.
 - ii. **Headers** – Provide additional information (e.g., content type, user-agent).
 - iii. **Body (Optional)** – Contains data for POST, PUT, or PATCH requests.

HTTP request message

- two types of HTTP messages: *request, response*
- **HTTP request message:**
 - ASCII (human-readable format)

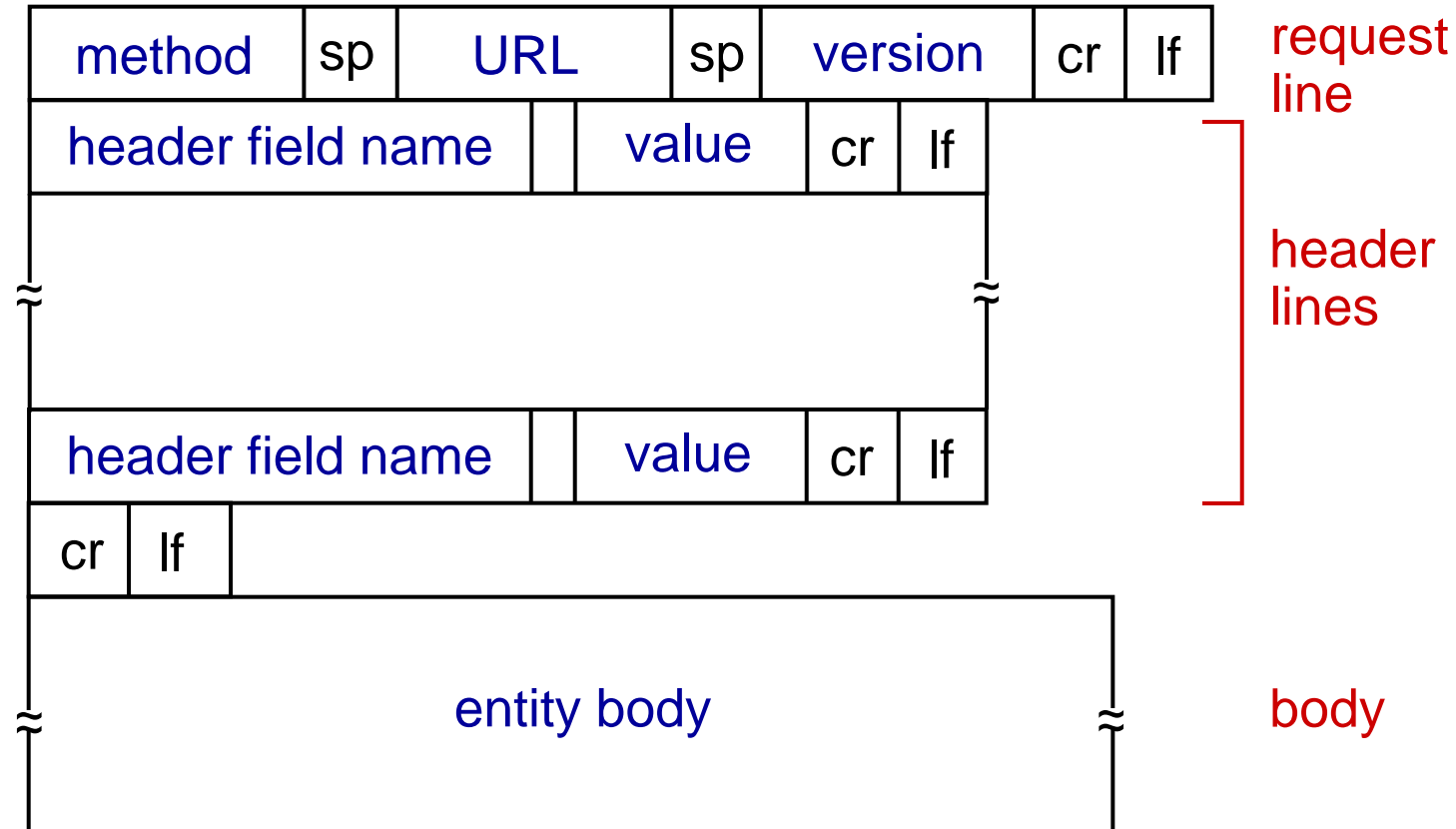
request line (GET, POST,
HEAD commands) →

/ carriage return character
/ line-feed character

carriage return, line feed →
at start of line indicates
end of header lines

* Check out the online interactive exercises for more
examples: http://gaia.cs.umass.edu/kurose_ross/interactive/

HTTP Request message: general format



HTTP Request Methods

1. GET
2. POST
3. PUT
4. HEAD
5. DELETE
6. PATCH
7. OPTIONS
8. CONNECT
9. TRACE

The GET Method

- GET is used to request data from a specified resource.
- Note that the query string (name/value pairs) is sent in the URL of a GET request:

`/test/demo_form.php?name1=value1&name2=value2`

- GET requests can be cached
- GET requests remain in the browser history
- GET requests can be bookmarked
- GET requests should never be used when dealing with sensitive data
- GET requests have length restrictions
- GET requests are only used to request data (not modify)

Path to the source
on Web Server

Parameters to the server

Protocol Version
Browser supports

The HTTP
Method

GET /RegisterStudent.asp?user=jhon&pass=java HTTP/1.1

The Request
Headers



Host: guru99.com

User-Agent: Mozilla/5.0

Accept-text/xml,text/html,text/plain, image/jpeg

Accept-Language: en-us,en

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-1, utf-8

Keep-Alive: 300

Connection: keep-alive

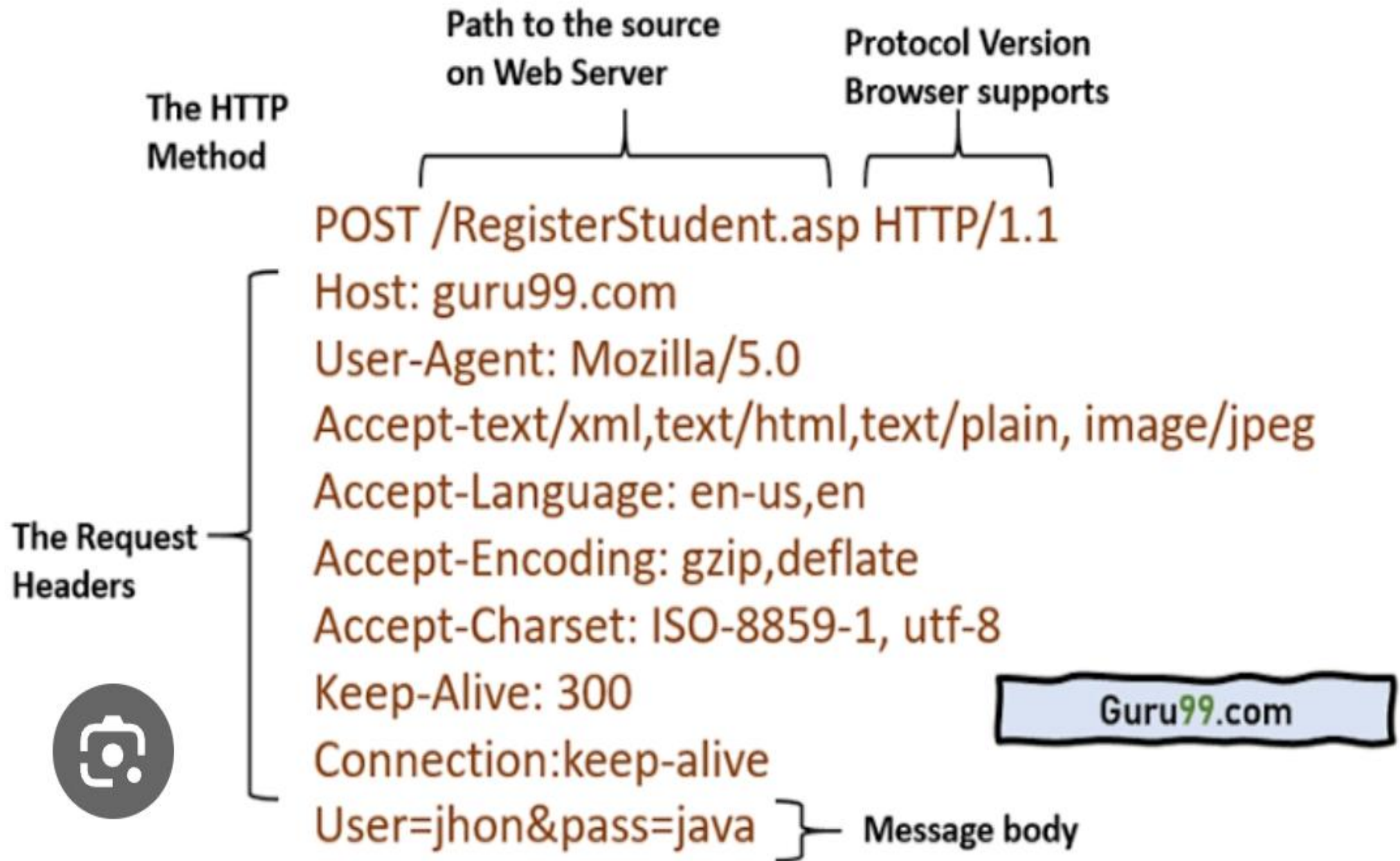
Guru99.com

The POST Method

- POST is used to send data to a server to create/update a resource.
- The data sent to the server with POST is stored in the request body of the HTTP request:
- POST /test/demo_form.php HTTP/1.1
Host: www.sastra.edu

name1=value1&name2=value2

- POST requests are never cached
- POST requests do not remain in the browser history
- POST requests cannot be bookmarked
- POST requests have no restrictions on data length



The PUT Method

- PUT is used to send data to a server to create/update a resource.
- The difference between POST and PUT is that PUT requests are **idempotent**.
- Calling the same PUT request multiple times will always produce the same result.
- Calling a POST request repeatedly have side effects of creating the same resource multiple times.
- Used to update or replace an existing resource.
- If the resource does not exist, it may create a new one.

The HEAD Method

- The HEAD method is identical to GET except that the server MUST NOT return a message-body in the response.
- The metainformation contained in the HTTP headers in response to a HEAD request SHOULD be identical to the information sent in response to a GET request.
- The HEAD method is used to ask only for information about a document, not for the document itself.
- **HEAD is much faster than GET**, as a much smaller amount of data is transferred.
- It's often used by clients who use caching, to see if the document has changed since it was last accessed.
- If it was not, then the local copy can be reused, otherwise the updated version must be retrieved with a GET.
- This method is often used for **testing hypertext links for validity, accessibility, and recent modification.**

- The DELETE Method - Deletes the specified resource.
- The PATCH Method - Partial modifications to a resource.
- The OPTIONS Method - Describes the communication options for the target resource.
- The CONNECT Method - Used to establish a tunnel (e.g., for HTTPS via a proxy).
- The TRACE Method - Used for debugging by returning the request as received by the server.

Method	Purpose
GET	Retrieve data
POST	Create new resource
PUT	Replace a resource
PATCH	Partially update a resource
DELETE	Remove a resource
HEAD	Retrieve headers
OPTIONS	List allowed methods
CONNECT	Establish a tunnel
TRACE	Debugging request

- HTML Country Codes

https://www.w3schools.com/tags/ref_country_codes.asp

- HTML Language Codes

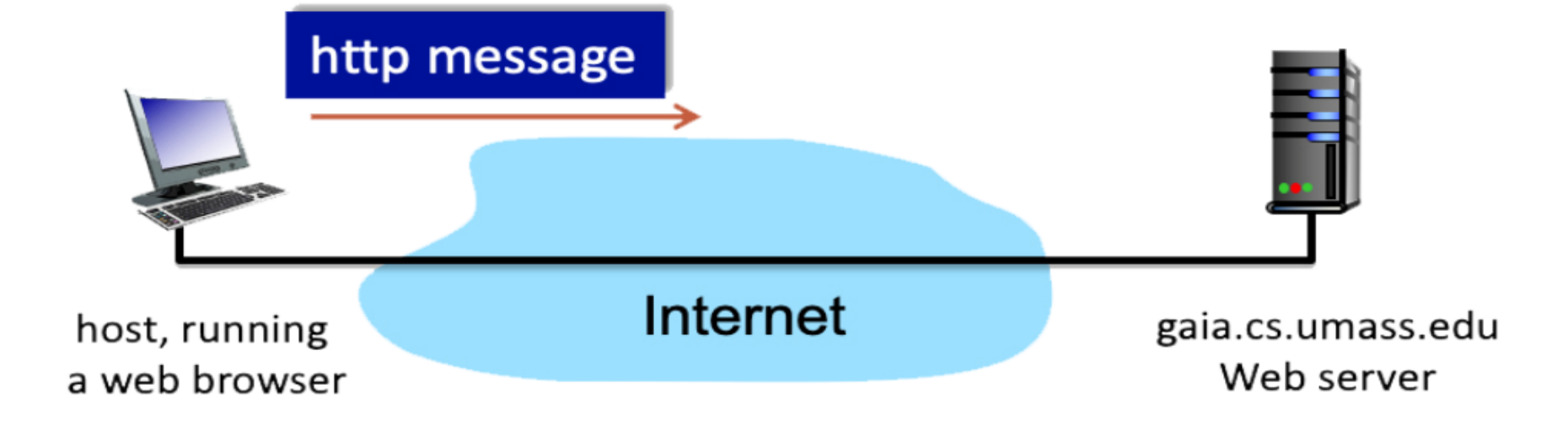
https://www.w3schools.com/tags/ref_language_codes.asp

- HTTP Error Messages

https://www.w3schools.com/tags/ref_httpmessages.asp

THE HTTP GET MESSAGE

Consider the figure below, where a client is sending an HTTP GET message to a web server, `gaia.cs.umass.edu`



Suppose the client-to-server HTTP GET message is the following:

```
GET /kurose_ross_sandbox/interactive/quotation8.htm HTTP/1.1
Host: gaia.cs.umass.edu
Accept: text/plain, text/html, image/gif, image/jpeg, audio/mp4, audio/basic, video/wmv, video/mpeg,
Accept-Language: en-us, en-gb;q=0.1, en;q=0.3, fr, fr-ch, zh, da, fi, ar
If-Modified-Since: Tue, 15 Aug 2023 23:39:39 -0700
User Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0)
```

QUESTION LIST

1. What is the name of the file that is being retrieved in this GET message?
2. What version of HTTP is the client running?
3. True or False: The client will accept html files
4. True or False: The client will accept jpeg images
5. What is the client's preferred version of English?
6. What is the client's least preferred version of English?
7. True or False: The client will accept the German language
8. True or False: The client already has a cached copy of the file


SOLUTION

1. The name of the file is quotation8.htm.
2. The client is running on HTTP/1.1
3. True. In the 'Accept' field the client includes 'text/html' files.
4. True. The client does include 'image/jpeg' in its 'Accept' field.
5. The client's preferred version of English is American English. Any language without a defined q value has a default value of 1
6. The client's least preferred version of English is British English because it has the lowest q value.
7. False. The client does NOT include German in its 'Accepted-Language' field.
8. True. The client has a cached copy of the file that was updated on: Tue, 15 Aug 2023 23:39:39 -0700

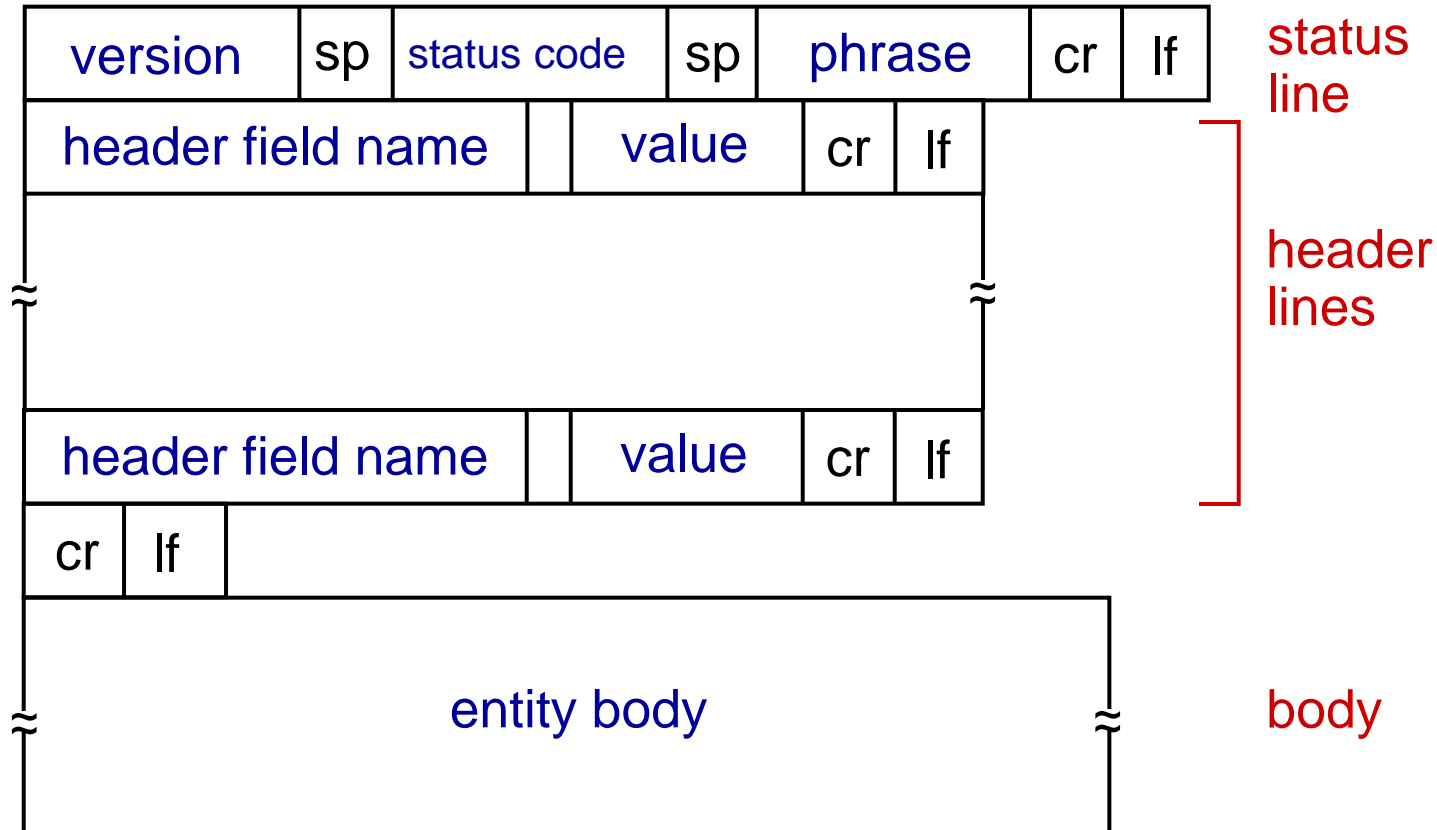
HTTP Response Structure

- An HTTP response consists of:
 - i. **Status Line** – Contains HTTP version, status code, and status message.
 - ii. **Headers** – Provide metadata about the response.
 - iii. **Body** – Contains the actual content (HTML, JSON, etc.).

HTTP response message

status line (protocol  HTTP/1.1 200 OK
status code status phrase)

HTTP response message: general format



Common HTTP Status Codes

■ Success Codes

Code	Meaning
200 OK	Request was successful
201 Created	Resource was successfully created
204 No Content	Request was successful, but no content returned

■ Client Error Codes

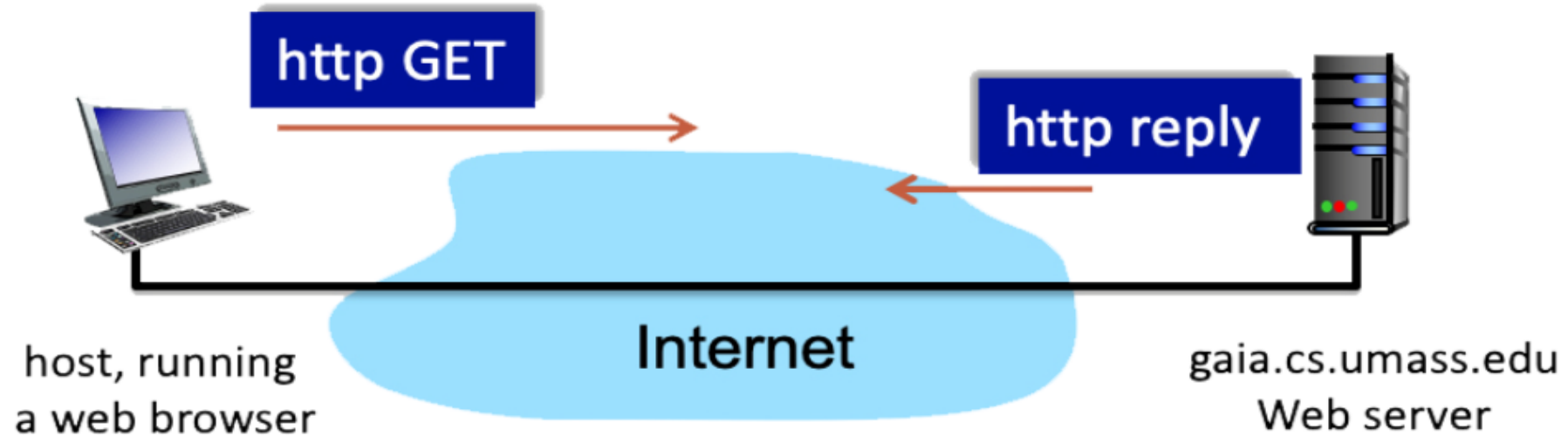
Code	Meaning
400 Bad Request	Invalid request syntax
401 Unauthorized	Authentication required
403 Forbidden	Access denied
404 Not Found	Resource not found

■ Server Error Codes

Code	Meaning
500 Internal Server Error	Generic server error
502 Bad Gateway	Invalid response from upstream server
503 Service Unavailable	Server is overloaded or down

THE HTTP RESPONSE MESSAGE

Consider the figure below, where the server is sending a HTTP RESPONSE message back the client.



Suppose the server-to-client HTTP RESPONSE message is the following:

```
HTTP/1.1 404 Not Found
Date: Wed, 16 Aug 2023 06:20:18 +0000
Server: Apache/2.2.3 (CentOS)
Content-Length: 2877
Keep-Alive: timeout=25, max=75
Connection: Keep-alive
Content-type: image/html
```

QUESTION LIST

1. Is the response message using HTTP 1.0 or HTTP 1.1?
2. Was the server able to send the document successfully? Yes or No
3. How big is the document in bytes?
4. Is the connection persistent or nonpersistent?
5. What is the type of file being sent by the server in response?
6. What is the name of the server and its version? Write your answer as server/x.y.z

SOLUTION

1. The response is using HTTP/1.1
2. Since the response code is 404 Not Found, the document was NOT received successfully.
3. The document is 2877 bytes.
4. The connection is persistent.
5. The file type the server is sending is image/html.
6. The name and version of the server is Apache/2.2.3

Maintaining user/server state: cookies

Web sites and client browser use *cookies* to maintain some state between transactions

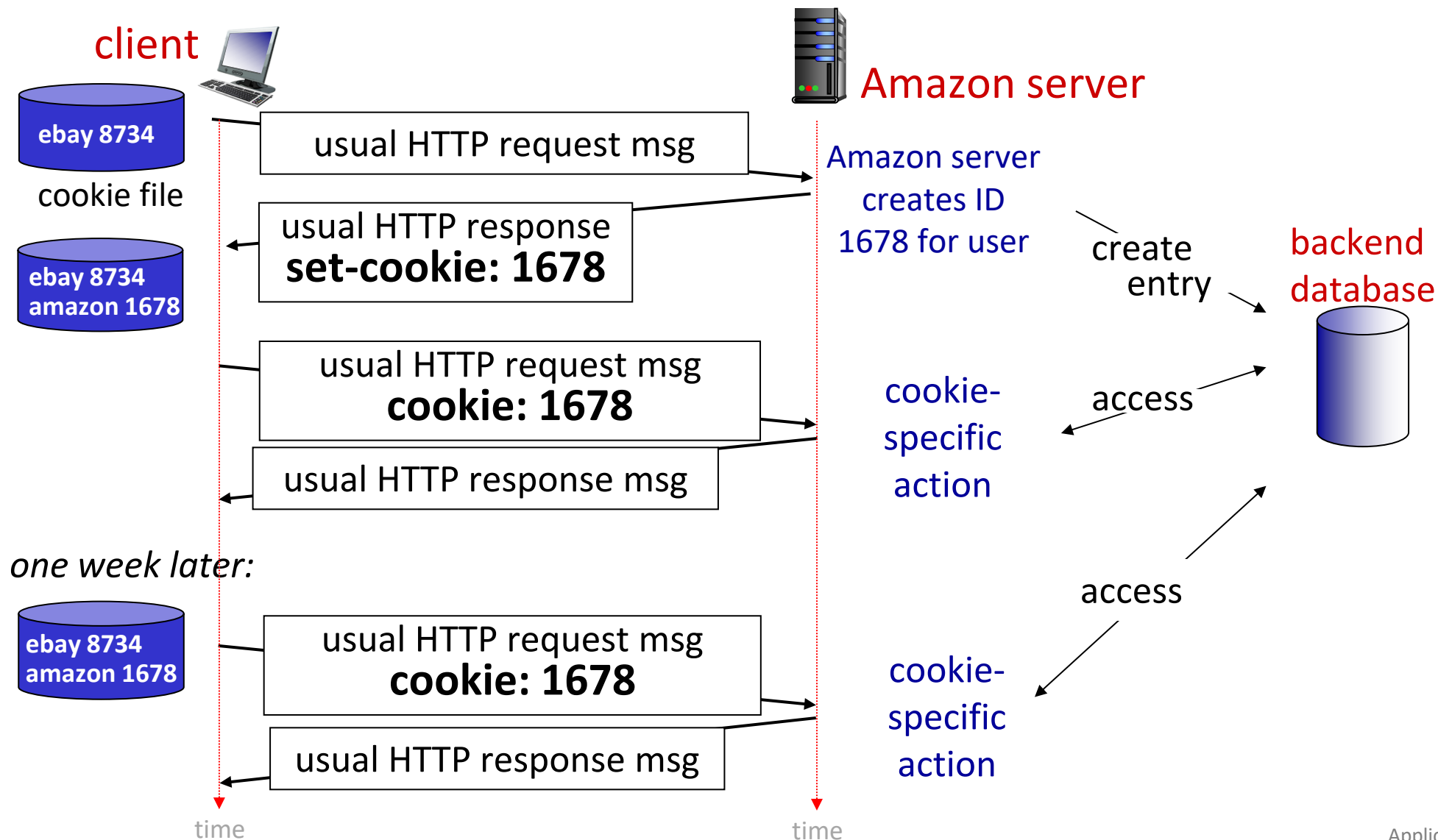
four components:

- 1) cookie header line of HTTP *response* message
- 2) cookie header line in next HTTP *request* message
- 3) cookie file kept on user's host, managed by user's browser
- 4) back-end database at Web site

Example:

- Susan uses browser on laptop, visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
 - unique ID (aka “cookie”)
 - entry in backend database for ID
- subsequent HTTP requests from Susan to this site will contain cookie ID value, allowing site to “identify” Susan

Maintaining user/server state: cookies



HTTP cookies: comments

What cookies can be used for:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

Challenge: How to keep state?

- *at protocol endpoints:* maintain state at sender/receiver over multiple transactions
- *in messages:* cookies in HTTP messages carry state

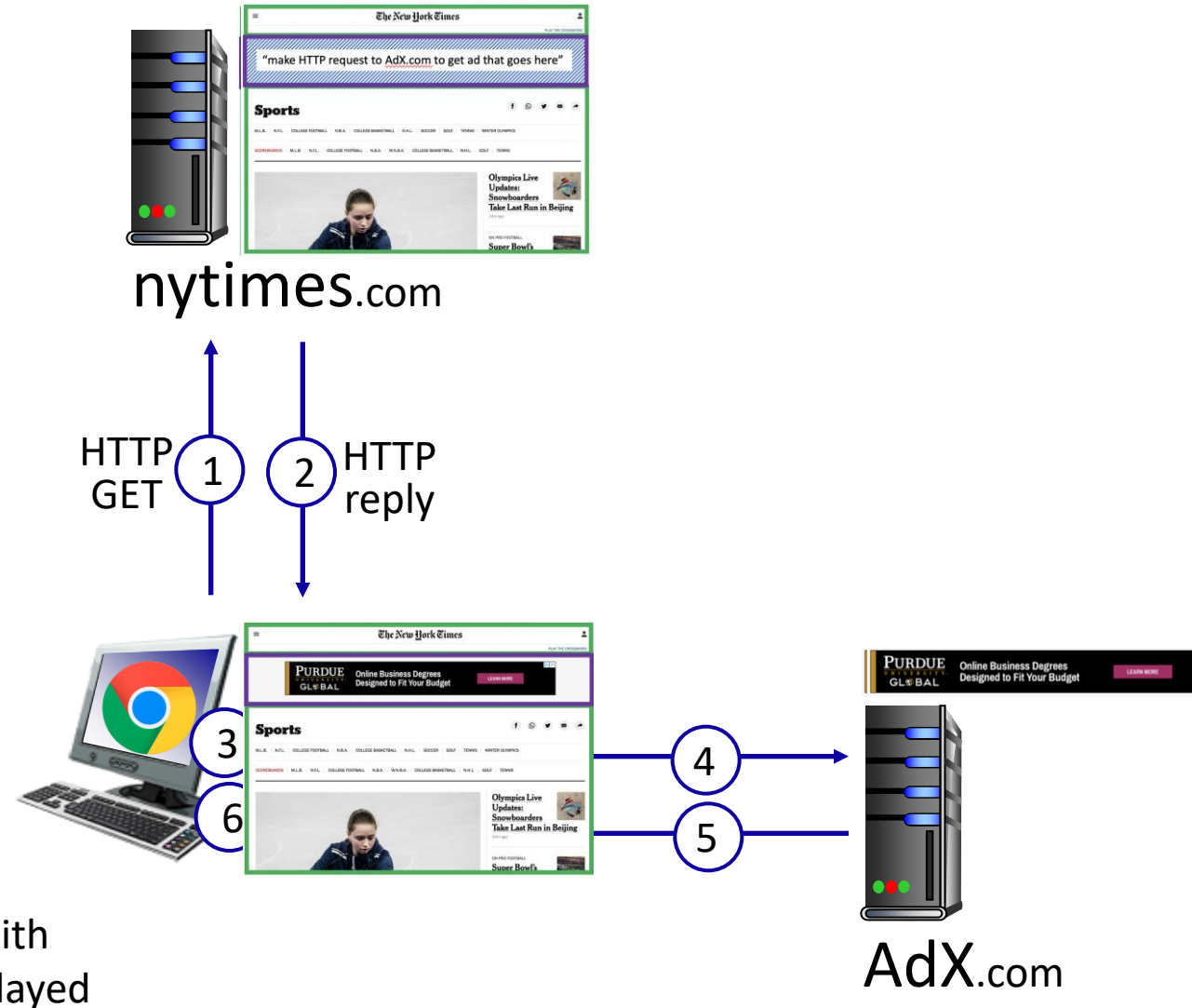
aside

cookies and privacy:

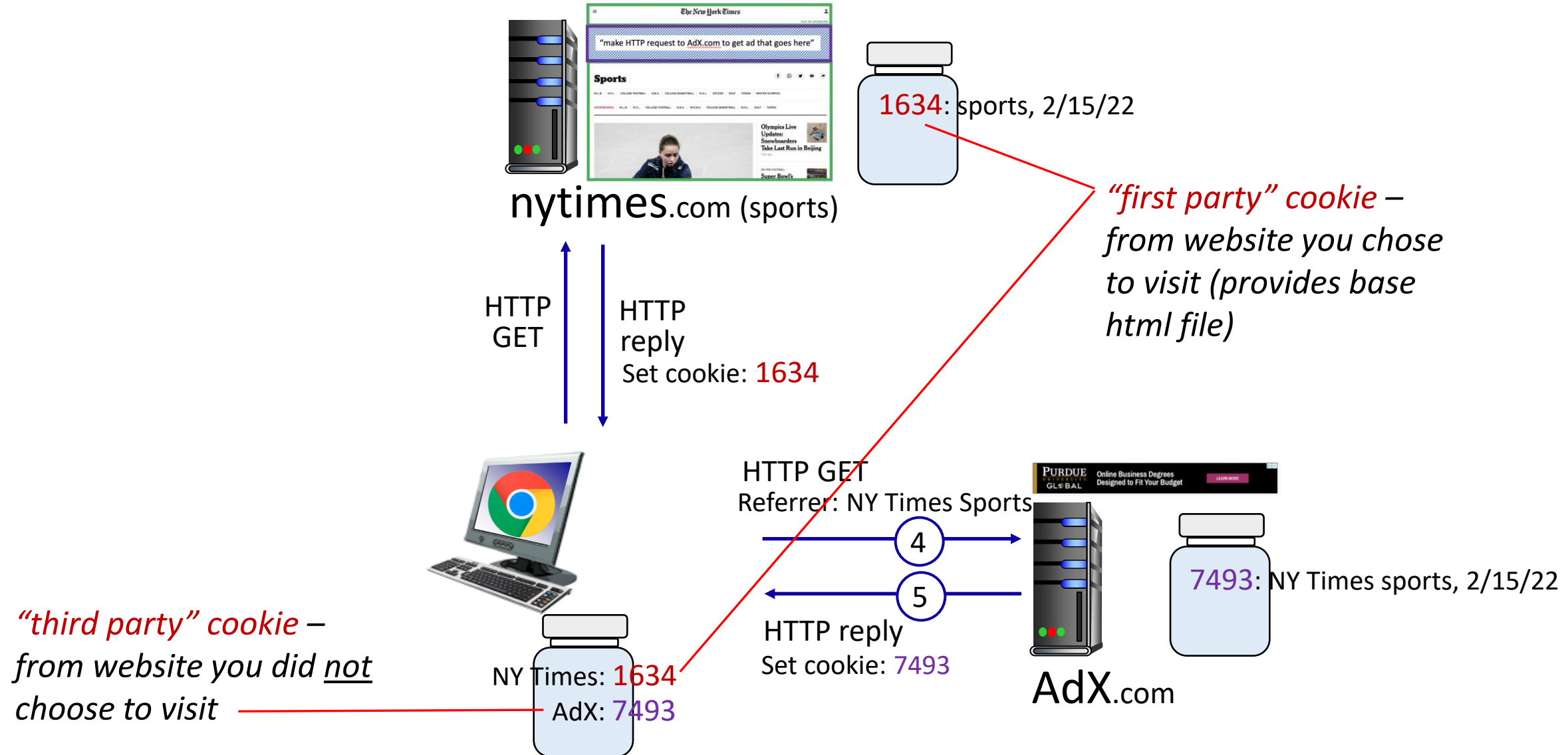
- cookies permit sites to *learn* a lot about you on their site.
- third party persistent cookies (tracking cookies) allow common identity (cookie value) to be tracked across multiple web sites

Example: displaying a NY Times web page

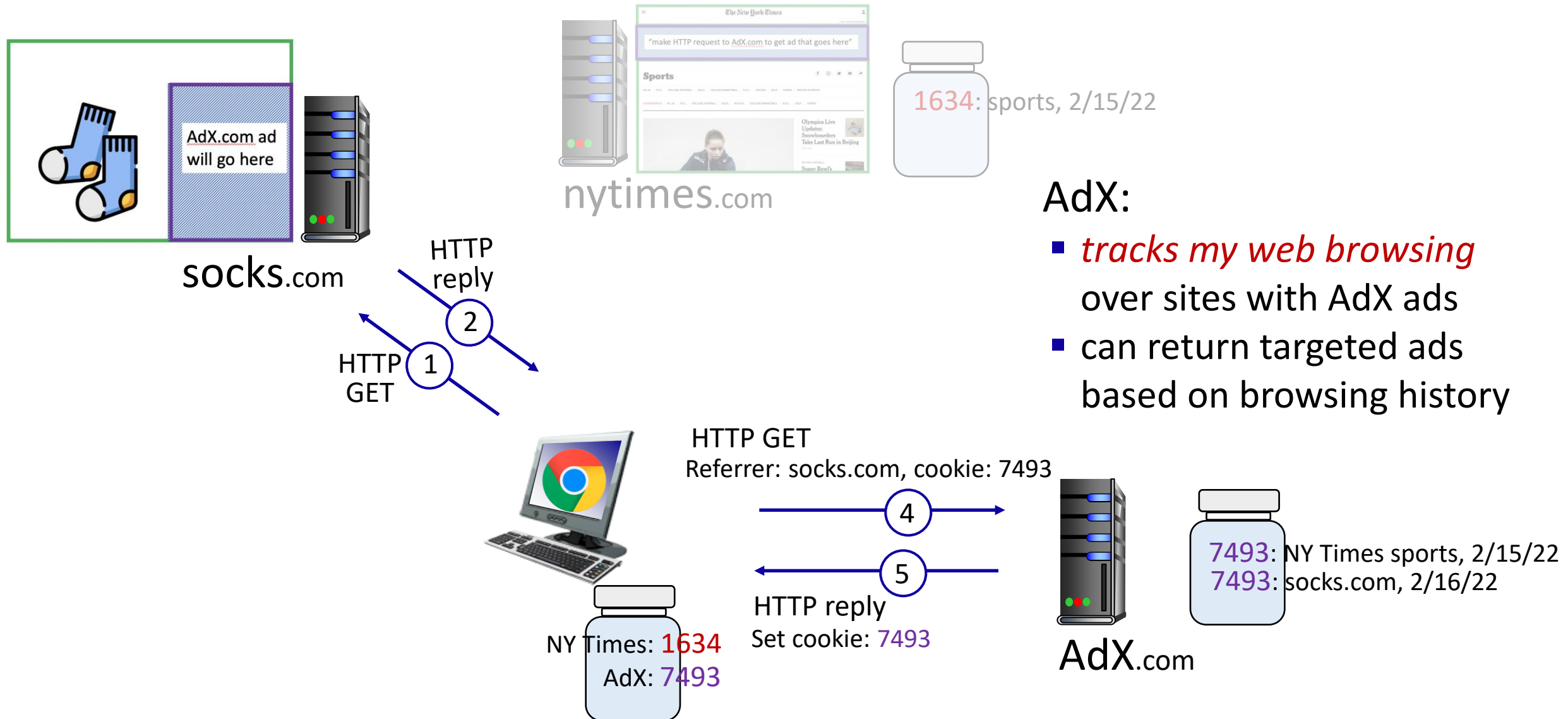
- 1 GET base html file
- 2 from nytimes.com
- 4 fetch ad from
- 5 AdX.com
- 7 display composed page



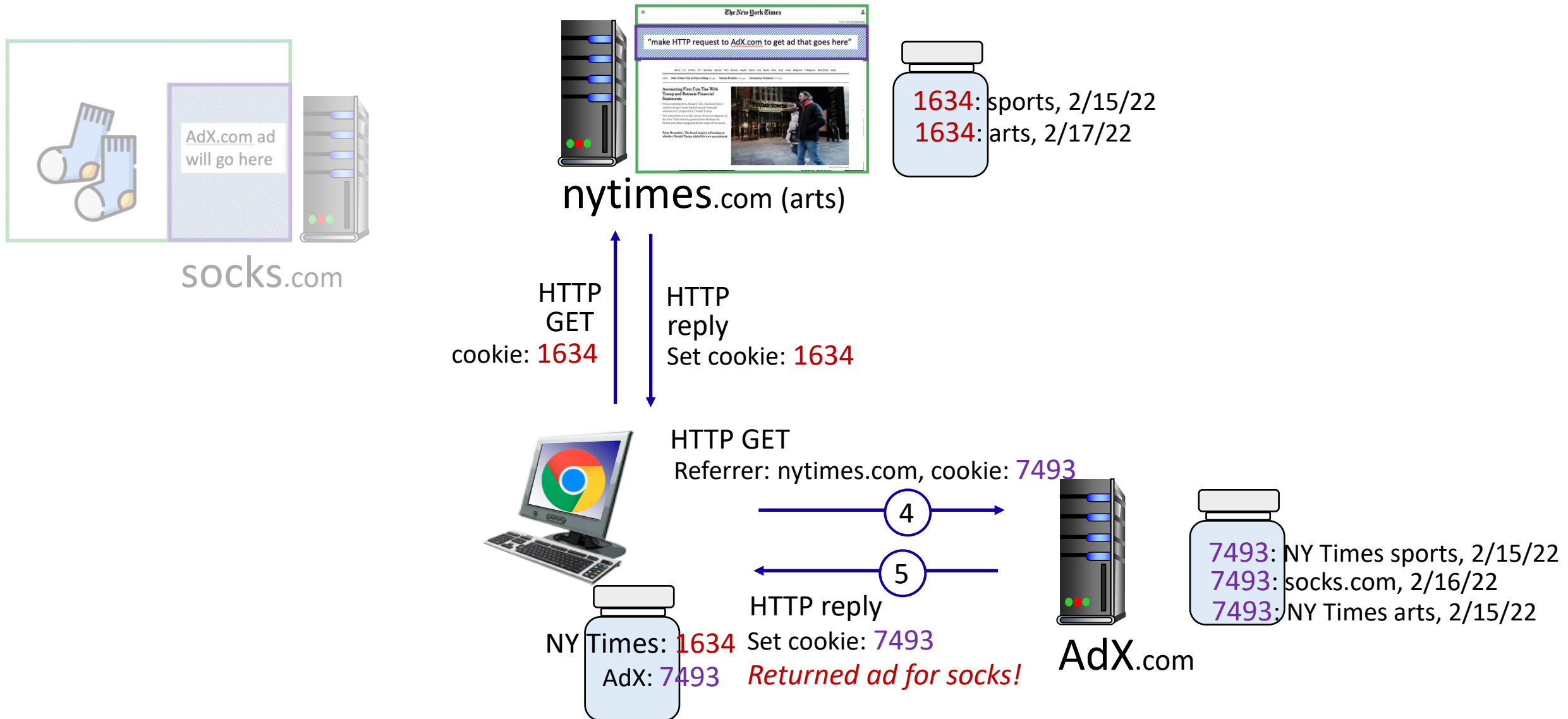
Cookies: tracking a user's browsing behavior



Cookies: tracking a user's browsing behavior



Cookies: tracking a user's browsing behavior (one day later)



Cookies: tracking a user's browsing behavior

Cookies can be used to:

- track user behavior on a given website (**first party cookies**)
- track user behavior across multiple websites (**third party cookies**) without user ever choosing to visit tracker site (!)
- tracking may be *invisible* to user:
 - rather than displayed ad triggering HTTP GET to tracker, could be an invisible link

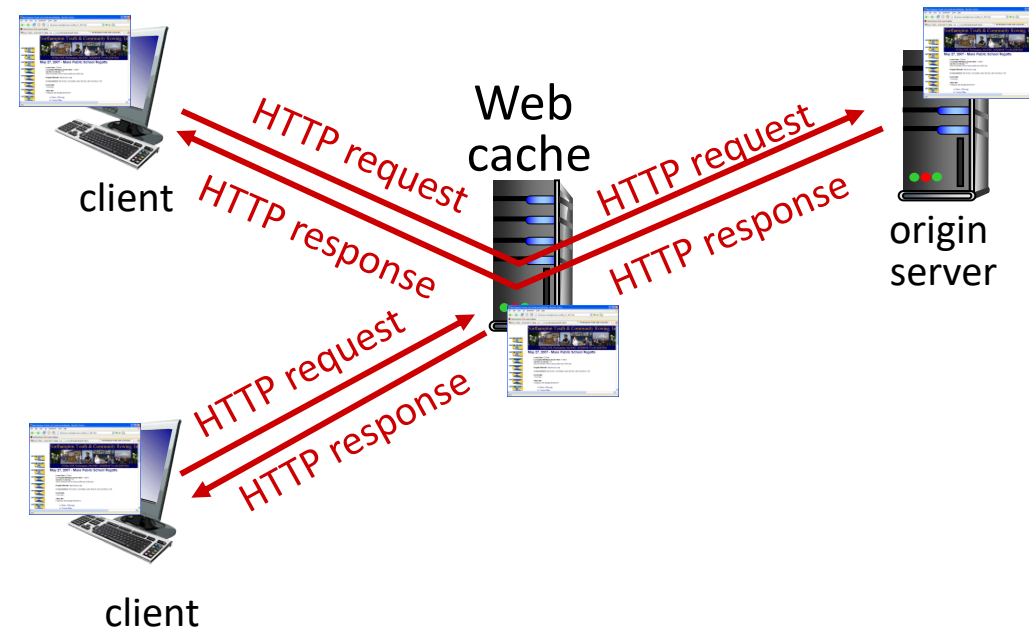
third party tracking via cookies:

- disabled by default in Firefox, Safari browsers
- to be disabled in Chrome browser in 2023

Web caches

Goal: satisfy client requests without involving origin server

- user configures browser to point to a (local) *Web cache*
- browser sends all HTTP requests to cache
 - *if* object in cache: cache returns object to client
 - *else* cache requests object from origin server, caches received object, then returns object to client



Web caches (aka proxy servers)

- Web cache acts as both client and server
 - server for original requesting client
 - client to origin server
- server tells cache about object's allowable caching in response header:

```
Cache-Control: max-age=<seconds>
```

```
Cache-Control: no-cache
```

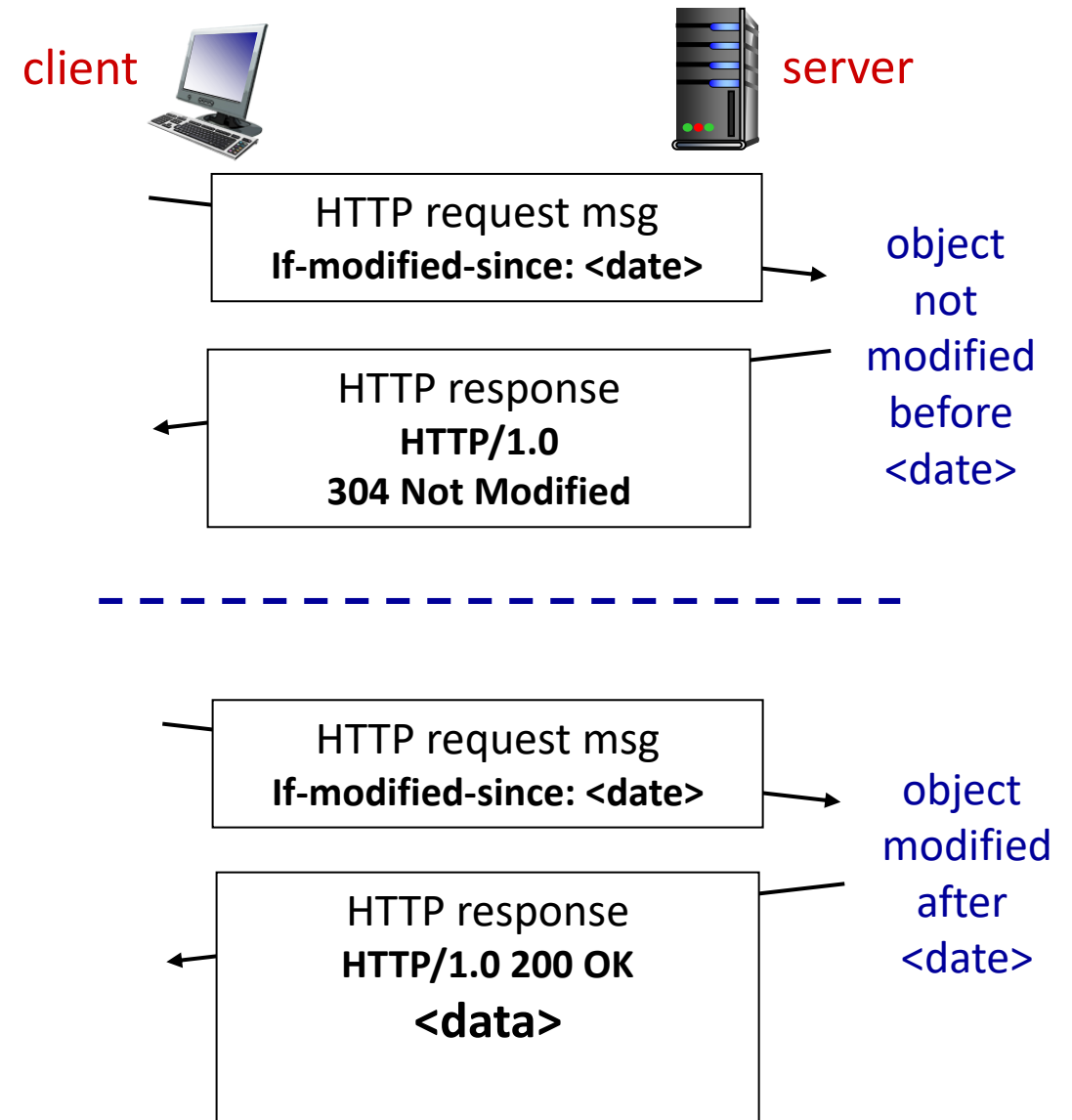
Why Web caching?

- reduce response time for client request
 - cache is closer to client
- reduce traffic on an institution's access link
- Internet is dense with caches
 - enables “poor” content providers to more effectively deliver content

Browser caching: Conditional GET

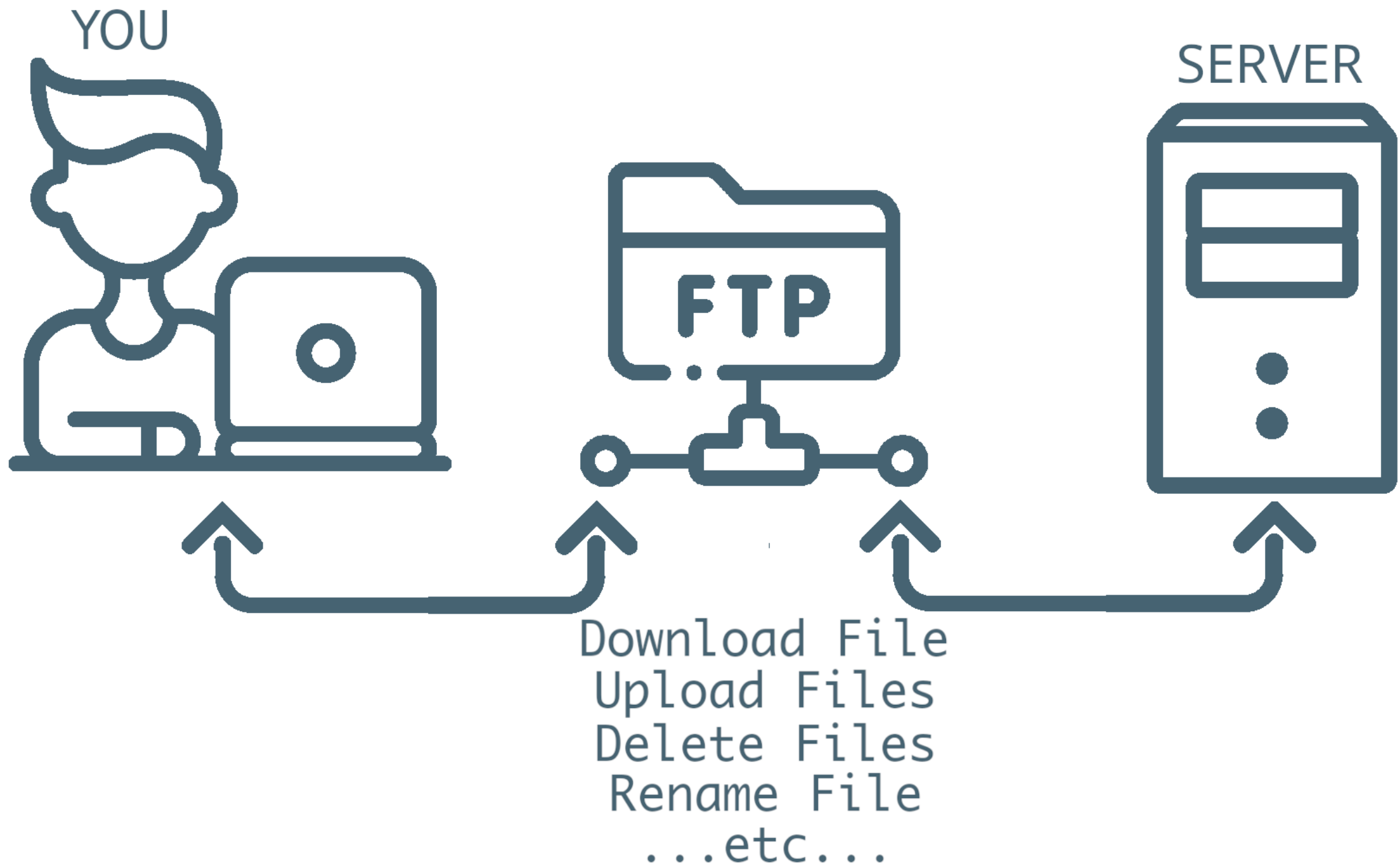
Goal: don't send object if browser has up-to-date cached version

- no object transmission delay (or use of network resources)
- **client:** specify date of browser-cached copy in HTTP request
If-modified-since: <date>
- **server:** response contains no object if browser-cached copy is up-to-date:
HTTP/1.0 304 Not Modified



Application layer: overview

1. WEB & HTTP
2. **FTP**
3. E-mail
4. DNS - The Internet's Directory Service
5. DDNS
6. TELNET
7. BLUETOOTH
8. FIREWALLS
9. SNMP



FTP

FTP (File Transfer Protocol) is a standard network protocol used for transferring files between a client and a server over a TCP-based network, such as the Internet. It operates on a client-server model and uses two separate channels:

- 1. Control Connection (Port 21)** – Used for sending commands and receiving responses.
- 2. Data Connection (Port 20 or Dynamic Ports)** – Used for transferring files.

Modes of FTP

- **Active Mode:** The client opens a port and waits for the server to initiate the data connection.
- **Passive Mode:** The client requests the server to open a port, and the client initiates the data connection (used when firewalls block incoming connections).

Types of FTP Transfers

- **ASCII Mode:** For transferring text files.
- **Binary Mode:** For transferring non-text files (e.g., images, videos, and executables).

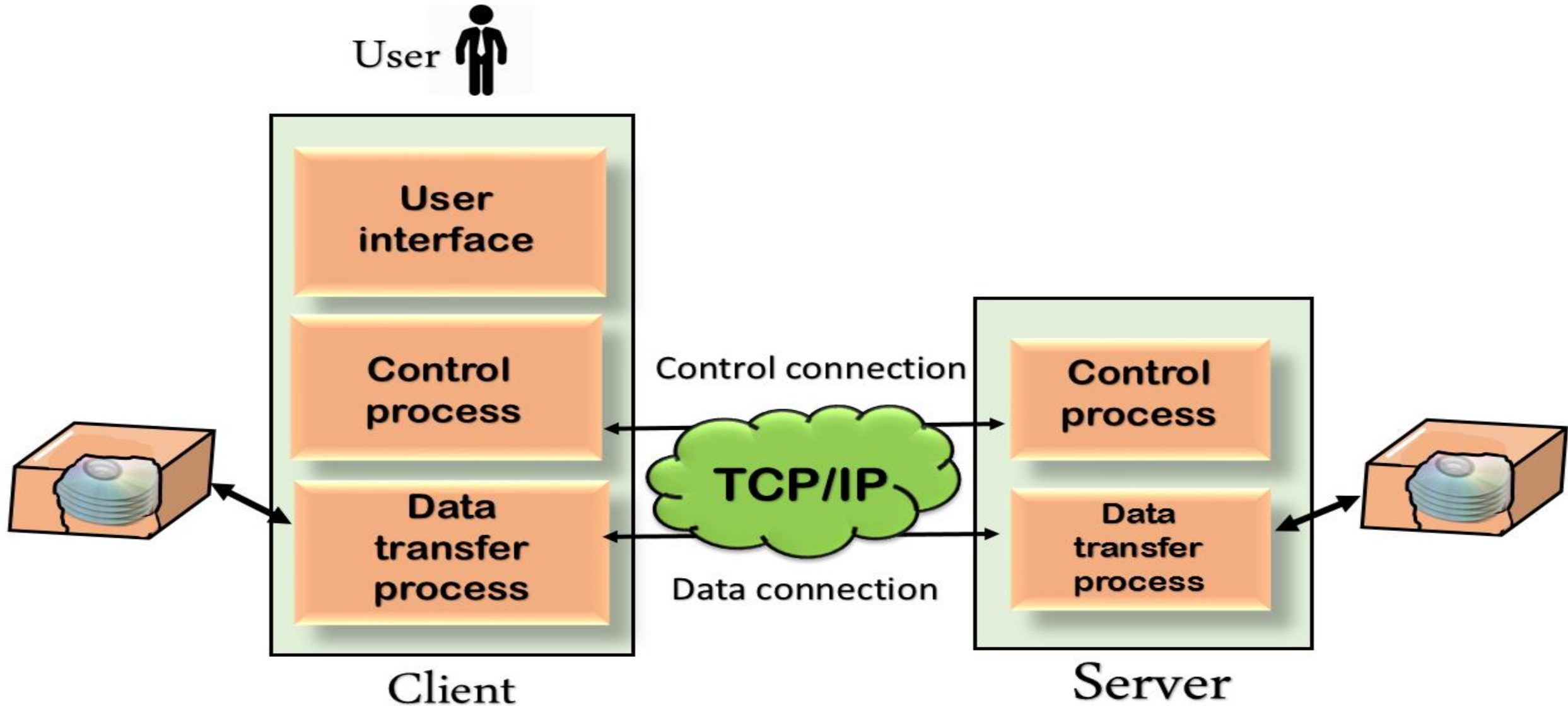
Security Concerns

- **FTP is not secure** since data, including passwords, is sent in plaintext.
- **Secure alternatives:**
 - FTPS (FTP Secure) – Uses SSL/TLS encryption.
 - SFTP (SSH File Transfer Protocol) – Uses SSH for secure file transfer.

FTP Fundamentals

- FTP is built on the client-server model architecture.
- To use it you need 2 main components:
 1. **FTP server**: software program that runs on the server from which you will be manipulating files
 2. **FTP client**: software you run to access the files on that server.

Mechanism of FTP



Different type of FTP clients

- Assuming you have an FTP server somewhere you know either its IP address or have a domain name, you can access such server through a tool called an **FTP client**.
- Different types of clients:
 1. command-line based client: ftp or lftp
 2. desktop client: Filezilla, Cyberduck
 3. web client: Filestash, MonstaFTP

The role of these clients is to talk to your server using the FTP protocol and perform the action you want to do on that server. For example, if you are hosting WordPress and want to change your theme, you will be able to use that client to update some data on your server.

Application layer: overview

1. WEB & HTTP
2. FTP
- 3. E-mail**
4. DNS - The Internet's Directory Service
5. DDNS
6. TELNET
7. BLUETOOTH
8. FIREWALLS
9. SNMP

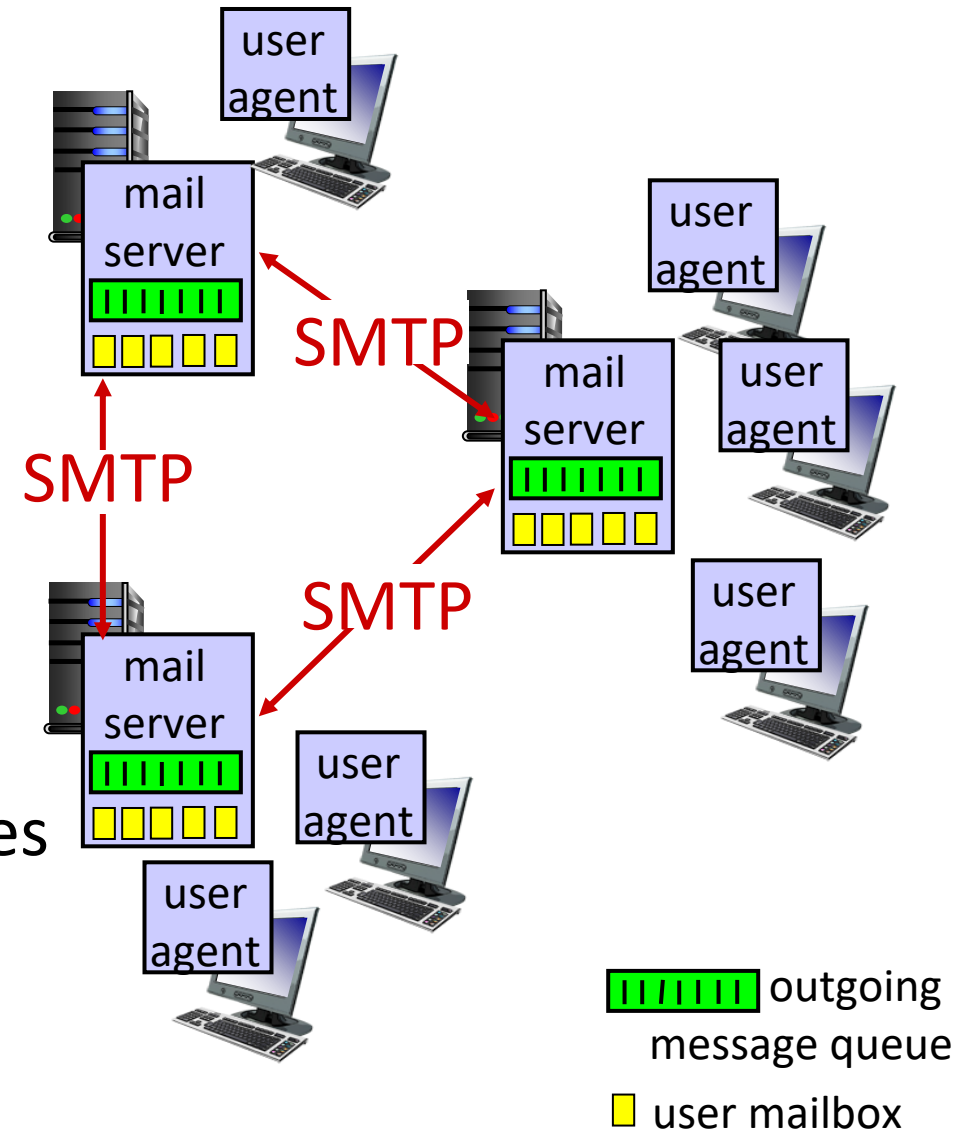
E-mail

Three major components:

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

User Agent

- a.k.a. “mail reader”
- composing, editing, reading mail messages
- e.g., Outlook, iPhone mail client
- outgoing, incoming messages stored on server



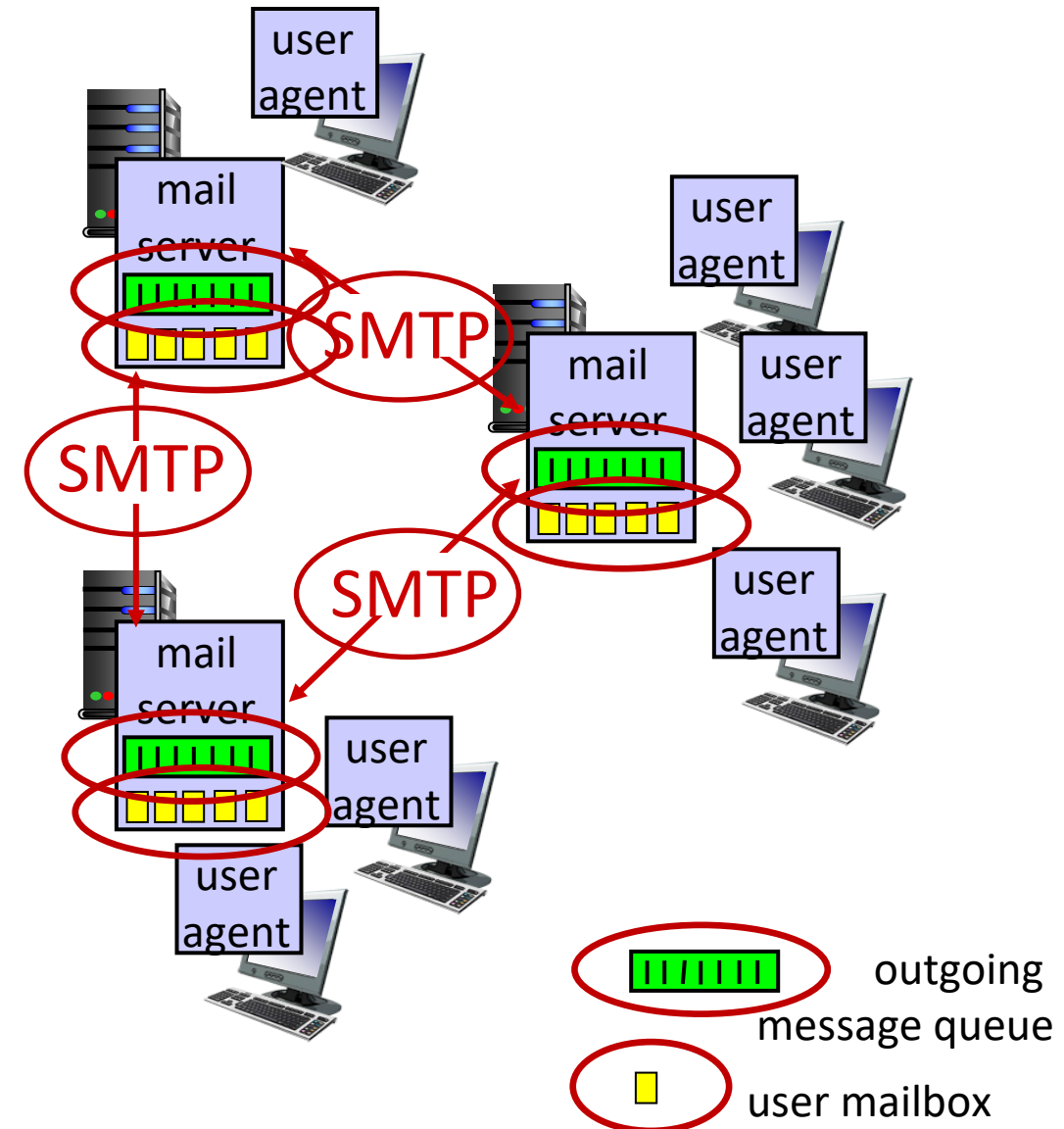
E-mail: mail servers

mail servers:

- *mailbox* contains incoming messages for user
- *message queue* of outgoing (to be sent) mail messages

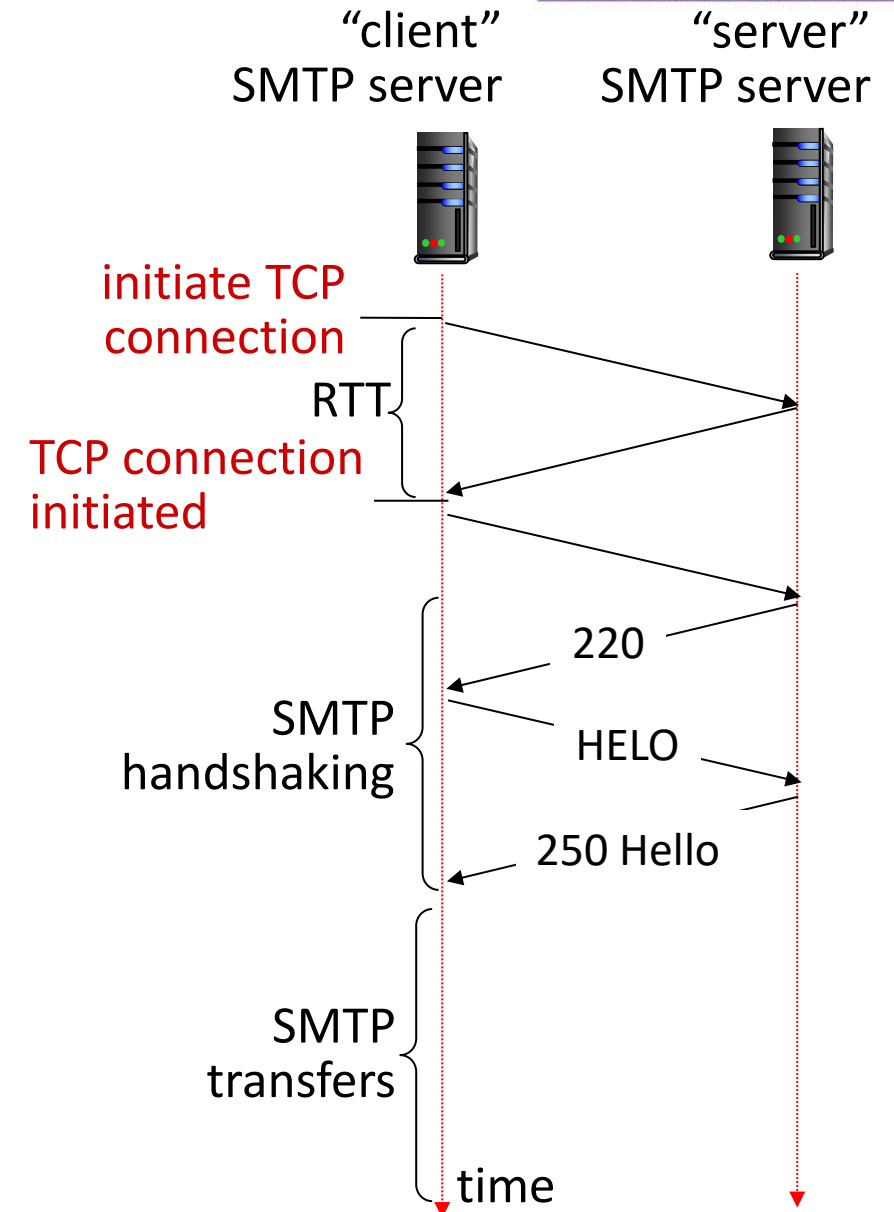
SMTP protocol between mail servers to send email messages

- *client*: sending mail server
- “*server*”: receiving mail server



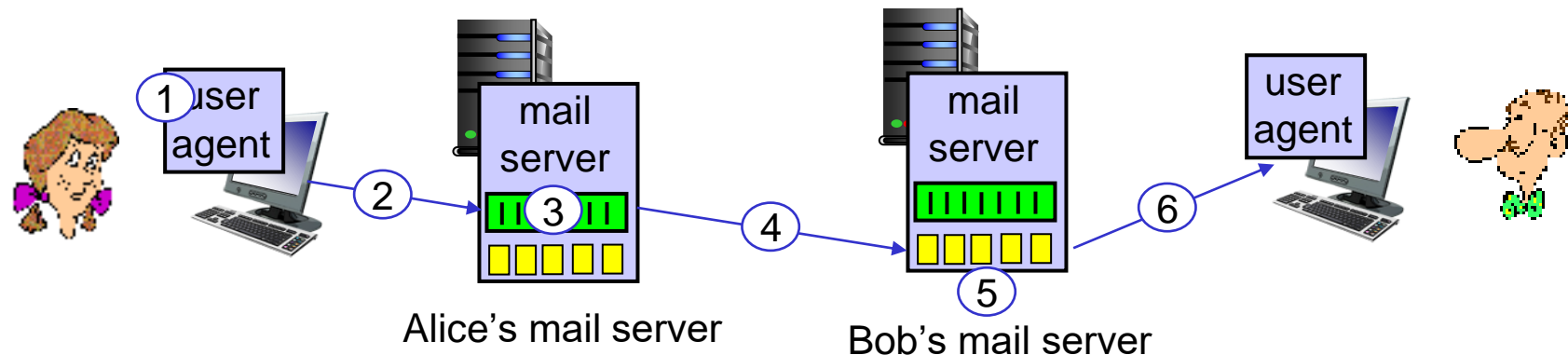
SMTP RFC (5321)

- uses TCP to reliably transfer email message from client (mail server initiating connection) to server, port 25
 - direct transfer: sending server (acting like client) to receiving server
- three phases of transfer
 - SMTP handshaking (greeting)
 - SMTP transfer of messages
 - SMTP closure
- command/response interaction (like HTTP)
 - **commands:** ASCII text
 - **response:** status code and phrase



Scenario: Alice sends e-mail to Bob

- 1) Alice uses UA to compose e-mail message “to” **bob@sastra.edu**
- 2) Alice’s UA sends message to her mail server using SMTP; message placed in message queue
- 3) client side of SMTP at mail server 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



Sample SMTP interaction

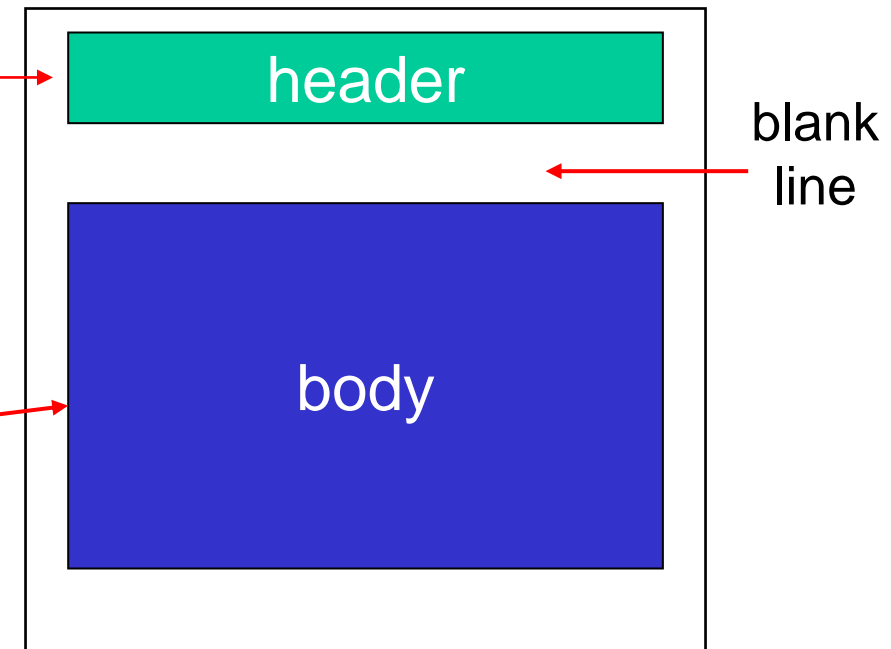
S: 220 hamburger.edu

Mail message format

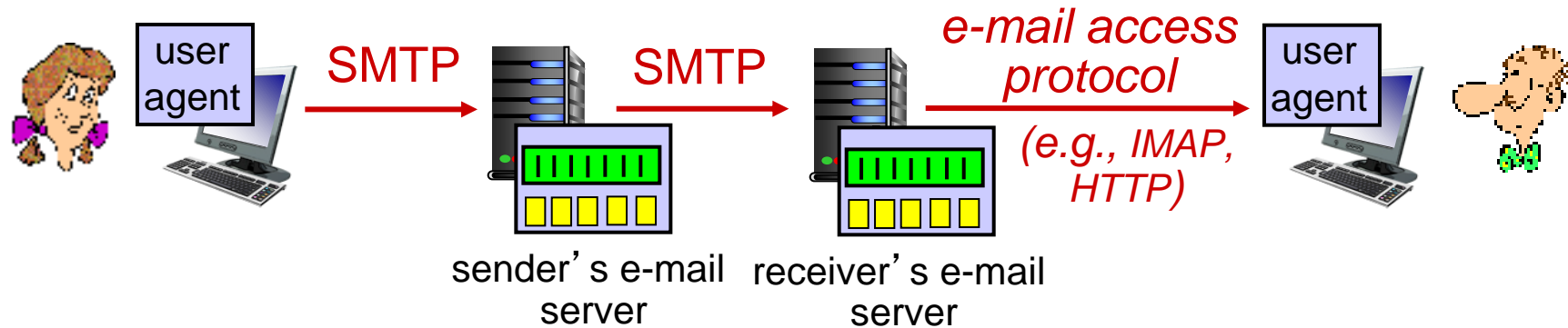
SMTP: protocol for exchanging e-mail messages, defined in RFC 5321 (like RFC 7231 defines HTTP)

RFC 2822 defines *syntax* for e-mail message itself (like HTML defines syntax for web documents)

- header lines, e.g.,
 - To:
 - From:
 - Subject:these lines, within the body of the email message area different from SMTP MAIL FROM:, RCPT TO: commands!
- Body: the “message” , ASCII characters only



Retrieving email: mail access protocols



- **SMTP:** delivery/storage of e-mail messages to receiver's server
- mail access protocol: retrieval from server
 - **IMAP:** Internet Mail Access Protocol [RFC 3501]: messages stored on server, IMAP provides retrieval, deletion, folders of stored messages on server
- **HTTP:** gmail, Hotmail, Yahoo!Mail, etc. provides web-based interface on top of SMTP (to send), IMAP (or POP) to retrieve e-mail messages



Internet

Sender's mail
Server



Receiver's mail
Server



IMAP (Internet Messaging Access Protocol)

- With IMAP accounts, messages are stored in a remote server. Users can log in via multiple email clients on computers or mobile device and read the same messages.
 - All changes made in the mailbox will be synced across multiple devices and messages will only be removed from the server if the user deletes the email.
-
- a) You can be logged in with multiple computers and devices simultaneously.
 - b) Your mail archive is synced and stored on the server for all connected devices to access.
 - c) Sent and received mail is stored on the server until the user permanently deletes it.

POP3 (Post Office Protocol)

- POP3 is an older protocol that was originally designed to be used on only one computer.
- Unlike modern protocols that use two-way synchronization, POP3 only supports one-way email synchronization, only allowing users to download emails from a server to a client.

IMAP

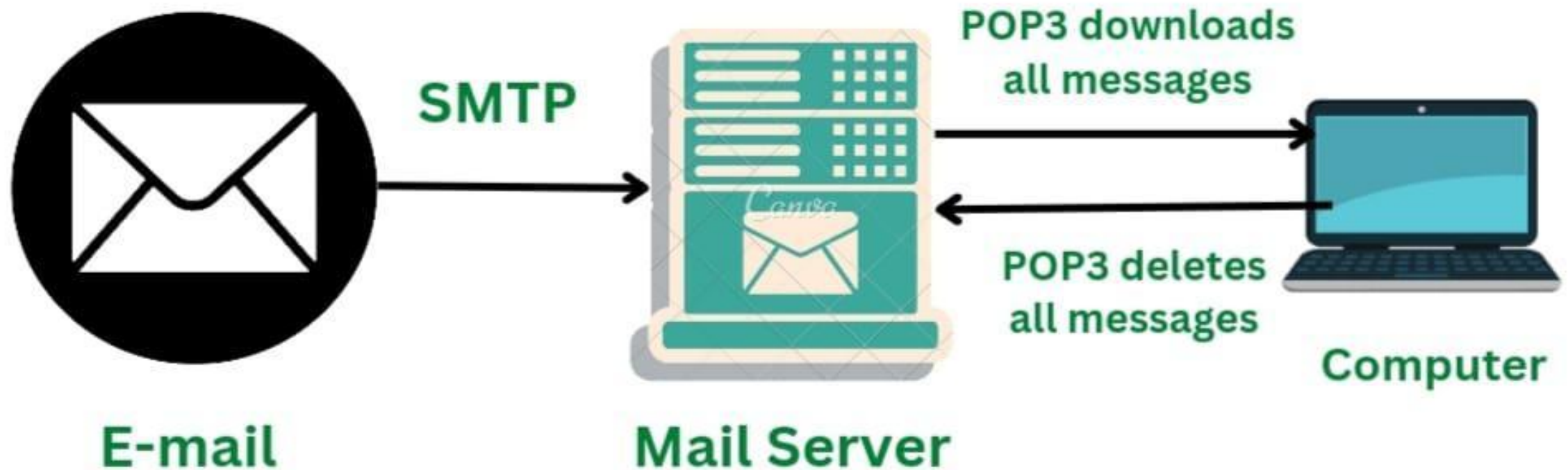
- Check your email from different devices, anywhere in the world:. Eg: your phone, a computer, a friend's computer.
- When you read an email message using IMAP, you aren't actually downloading or storing it on your computer; instead, you're reading it from the email service.
- IMAP only downloads a message when you click on it, and attachments aren't automatically downloaded.
- This way you're able to check your messages a lot more **quickly than POP**.

POP

- POP works by contacting your email service and downloading all of your new messages from it.
- Once they are downloaded onto your PC or Mac, they are deleted from the email service. **This means that after the email is downloaded, it can only be accessed using the same computer.**
- If you try to access your email from a different device, the messages that have been previously downloaded won't be available to you.
- ***Sent mail is stored locally on your PC or Mac, not on the email server.***
- A lot of ISPs give you email accounts that use POP.

The user can access the messages locally even if the user is offline

Various email applications such as Microsoft Outlook, Apple Mail, Gmail supports POP3 protocol.



- Say, throughout the night while you're sleeping, a stack of emails are sent to your email address and are starting to pile up on your mail server.
- **When you wake up and access your mail from your phone,**
- POP3 will download all the emails to your phone for you to view, and by doing so, all emails are removed from the mail server
- IMAP will send a copy of the emails to your phone, but leaving the originals on your mail server
- **Then you return to your office and check your emails on your computer,**
- POP3 will connect to your mail server and download all new emails – emails that were received since you last checked your email account (via your phone). But because all previous emails you have checked in the morning were already downloaded to your phone and removed from the mail server, **those emails will not show up on your computer.**
- IMAP will connect to your mail server and look for emails that are available and not yet on your computer. This includes all new emails received since the last time you checked your account, as well as all emails you have already accessed but from a different device (via your phone earlier the day).

POP3 or IMAP, which is better?

- If you use multiple devices to check, respond to and send emails, you would benefit from IMAP because of its cross-device access. All changes you make to the email as well as your email account (ie. setting up folders) are synced with the mail server and all devices you are using to access that email account. Also, if anything were to happen to your computer or phone, you don't have to worry about losing your emails as the **originals are still on your mail server**.
- On the other hand, if you have a designated device for emails and prefer to have all emails (including all attachments) accessible even offline, then POP will guarantee that you always have them, even if you don't have access to the Internet. Emails are stored locally on your device. But, unless you have POP configured to store your emails on the server instead of deleting them, if anything should happen to your device, all emails that you have previously downloaded or accessed will be gone.

Use IMAP if:

you check your emails from multiple devices (such as phone, computers, tablets, etc.)

Use POP3 if:

you are using one email client on one dedicated device (ie, your work computer at your office) and if you have a huge history of emails and/or you have a limited email storage space

*Port **110** is the default POP3 port and it is not encrypted.*

*The encrypted port for POP3 is **995** and works over TLS/SSL.*

*IMAP over SSL/TLS (IMAPS) is assigned the port number **993**.*

*SMTP used port **25** but today, SMTP should instead use port **587** for encrypted email transmissions using SMTP Secure (SMTPS).*

Application layer: overview

1. WEB & HTTP
2. FTP
3. E-mail
- 4. DNS - The Internet's Directory Service**
5. DDNS
6. TELNET
7. BLUETOOTH
8. FIREWALLS
9. SNMP

DNS (Domain Name System)

- DNS is a hierarchical and decentralized system that translates domain names (e.g., `www.google.com`) into IP addresses (e.g., `142.250.182.4`).
- This allows users to access websites using human-readable names instead of numerical IP addresses.

DNS: services, structure

DNS services:

- hostname-to-IP-address translation
- host aliasing
 - canonical, alias names
- mail server aliasing-(web server & Mail server)
- load distribution
 - replicated Web servers: many IP addresses correspond to one name
 - DNS Servers are often UNIX machines running BIND Software.
 - **DNS runs over UDP and uses port 53**

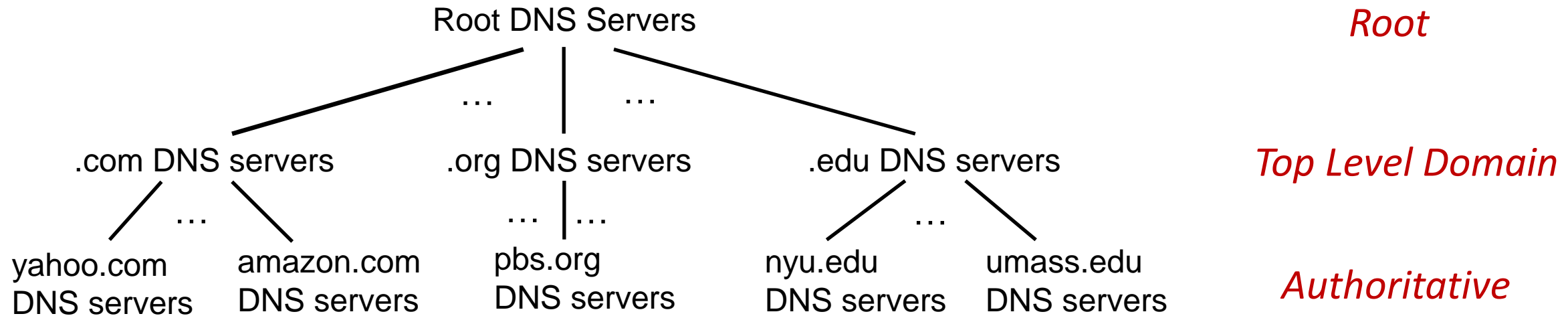
Q: Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

A: doesn't scale!

- Comcast DNS servers alone: 600B DNS queries/day
- Akamai DNS servers alone: 2.2T DNS queries/day

DNS: a distributed, hierarchical database



Client wants IP address for www.amazon.com; 1st approximation:

- client queries root server 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 www.amazon.com

Types of DNS Servers

- **Recursive Resolver** – Resolves domain names by querying multiple DNS servers.
- **Root DNS Server** – First step in translating domain names; contains info about TLD servers.
- **TLD (Top-Level Domain) Server** – Manages domains under specific extensions (e.g., .com, .org).
- **Authoritative DNS Server** – Stores actual domain-to-IP mappings for websites.

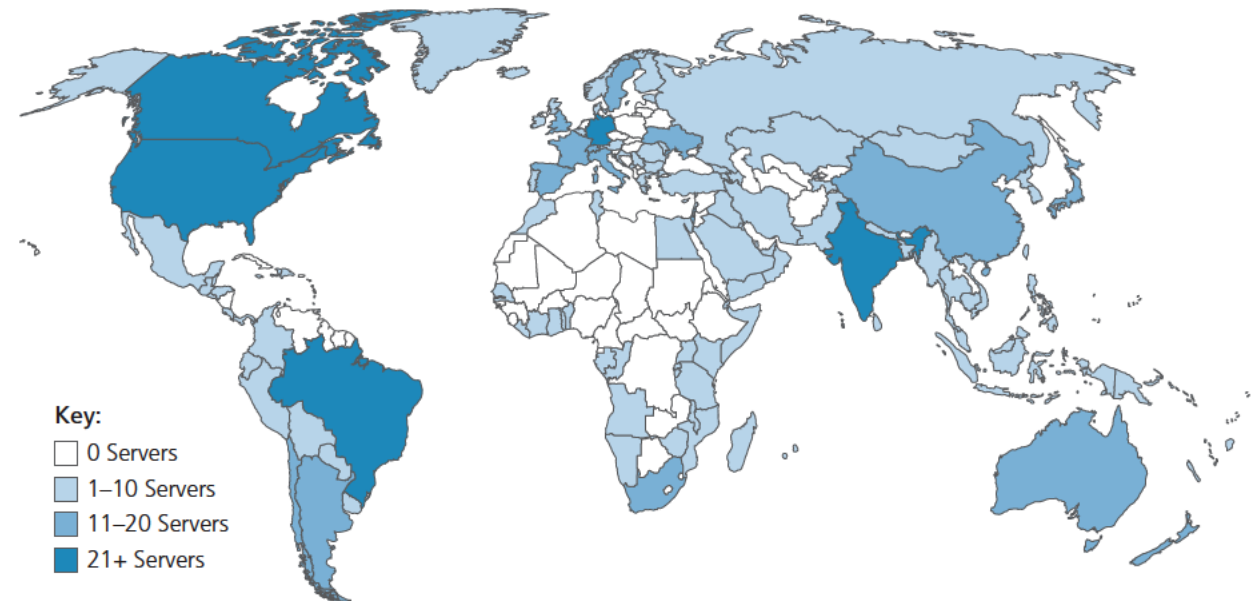
How DNS Works

1. User enters a domain name in a browser.
2. The request is sent to a recursive DNS resolver (provided by the ISP).
3. The resolver queries DNS root servers, which point to the appropriate Top-Level Domain (TLD) server (e.g., .com, .org).
4. The TLD server directs the query to the authoritative DNS server for the domain.
5. The authoritative server returns the IP address to the resolver.
6. The resolver sends the IP address to the browser, which connects to the website.

DNS: root name servers

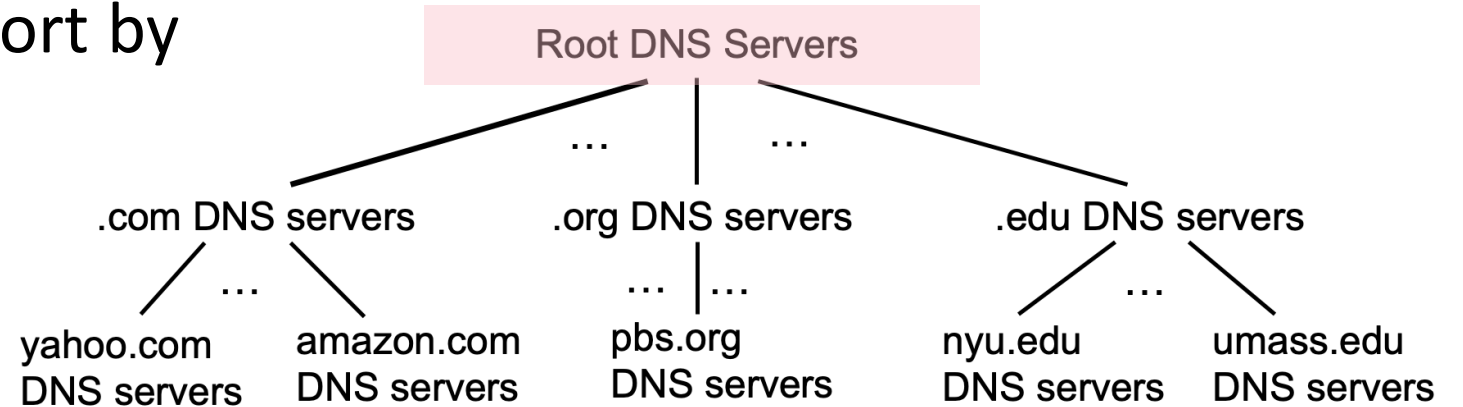
- official, contact-of-last-resort by name servers that can not resolve name
- *incredibly important* Internet function
 - Internet couldn't function without it!
 - DNSSEC – provides security (authentication, message integrity)
- ICANN (Internet Corporation for Assigned Names and Numbers) manages root DNS domain

13 logical root name “servers”
worldwide each “server” replicated
many times (~200 servers in US)



DNS: root name servers

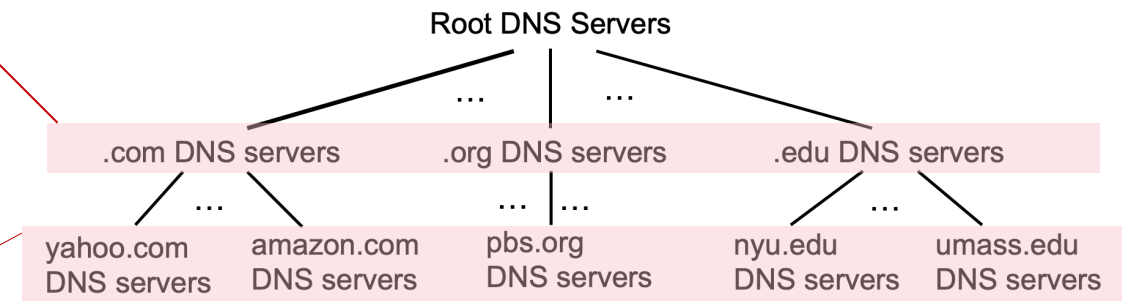
- official, contact-of-last-resort by name servers that can not resolve name



Top-Level Domain, and authoritative servers

Top-Level Domain (TLD) servers:

- responsible for .com, .org, .net, .edu, .aero, .jobs, .museums, and all top-level country domains, e.g.: .cn, .uk, .fr, .ca, .jp
- Verisign Global Registry Services: authoritative registry for .com TLD
- Educause: .edu TLD



authoritative DNS servers:

- organization's own DNS server(s), providing authoritative hostname to IP mappings for organization's named hosts
- can be maintained by organization or service provider

Local DNS name servers

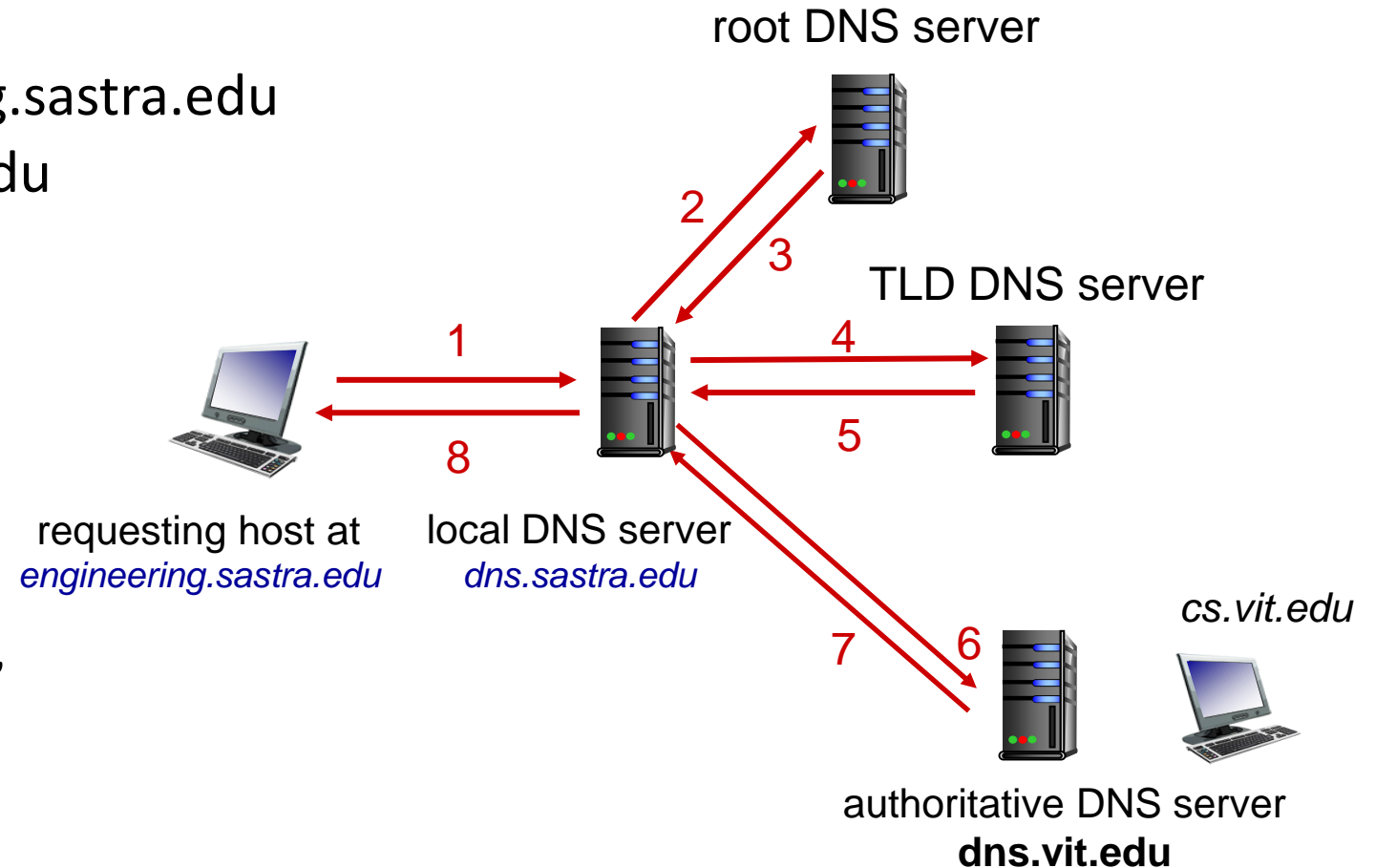
- when host makes DNS query, it is sent to its *local* DNS server
 - Local DNS server returns reply, answering:
 - from its local cache of recent name-to-address translation pairs (possibly out of date!)
 - forwarding request into DNS hierarchy for resolution
 - each ISP has local DNS name server; to find yours:
 - MacOS: `% scutil --dns`
 - Windows: `>ipconfig /all`
- local DNS server doesn't strictly belong to hierarchy

DNS name resolution: iterated query

Example: host at `engineering.sastra.edu` wants IP address for `cs.vit.edu`

Iterated query:

- contacted server replies with name of server to contact
- “I don’t know this name, but ask this server”

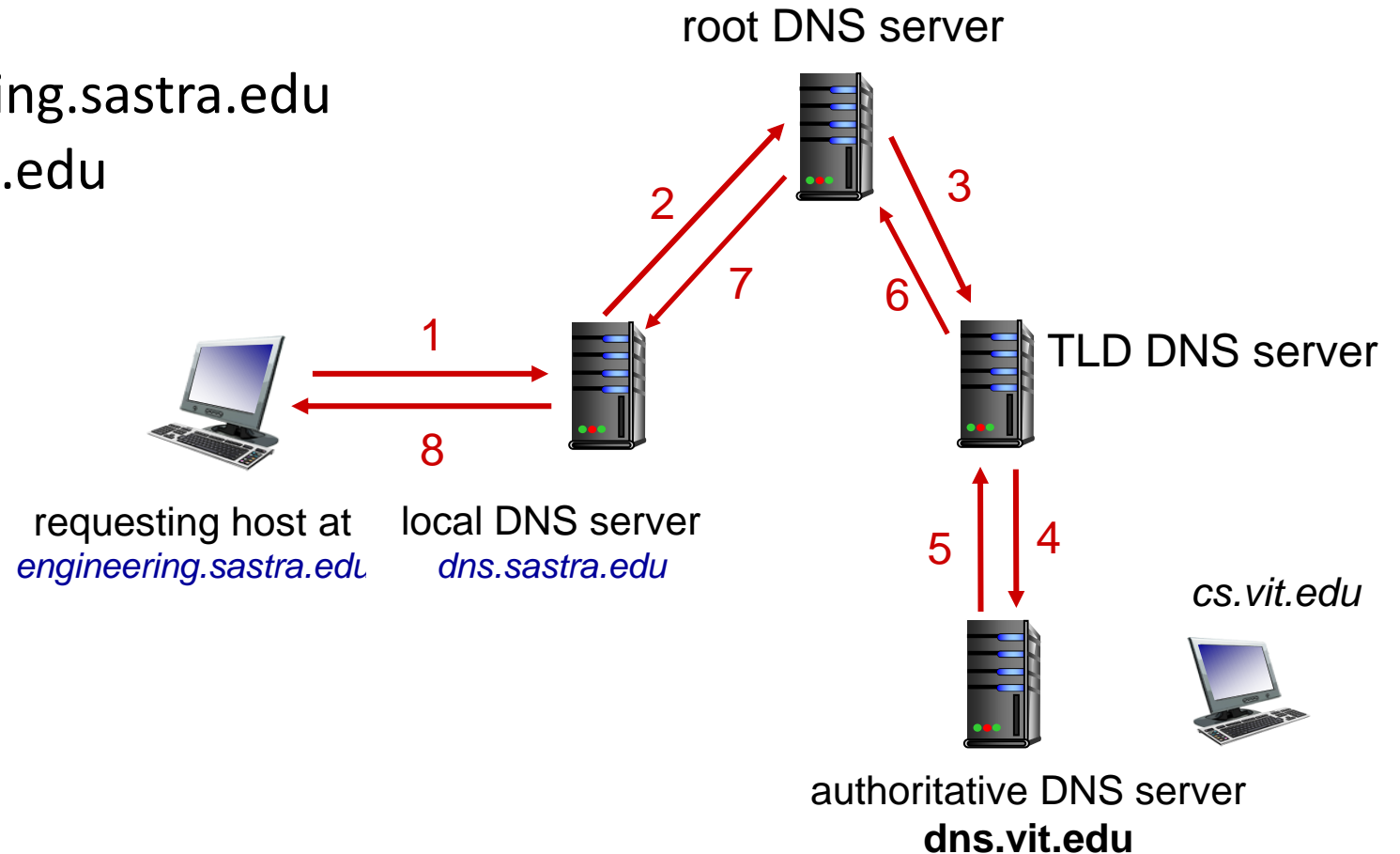


DNS name resolution: recursive query

Example: host at `engineering.sastra.edu` wants IP address for `cs.vit.edu`

Recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?



DNS Caching

- once (any) name server learns mapping, it *caches* mapping, and *immediately* returns a cached mapping in response to a query
 - caching improves response time
 - cache entries timeout (disappear) after some time (TTL)
 - TLD servers typically cached in local name servers
- cached entries may be *out-of-date*
 - if named host changes IP address, may not be known Internet-wide until all TTLs expire!
 - *best-effort name-to-address translation!*

Types of DNS Records

- ✓ **A Record** – Maps a domain to an IPv4 address.
- ✓ **AAAA Record** – Maps a domain to an IPv6 address.
- ✓ **CNAME Record** – Creates an alias for another domain name.
- ✓ **MX Record** – Specifies mail servers for email delivery.
- ✓ **TXT Record** – Stores text information, often used for security (e.g., SPF, DKIM).
- ✓ **NS Record** – Defines the authoritative name servers for a domain.

DNS records

DNS: distributed database storing resource records (RR)

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

type=A

- name is hostname
- value is IP address
- **(soc.sastra.edu, 145.67.88.126, A)**

type=CNAME FQDN (Fully Qualified Domain Name)

- name is alias name for some “canonical” (the real) name
- value is canonical name
- **(www.ibm.com, servereast.backup2.ibm.com, CNAME)**

type=NS

- name is domain (e.g., foo.com)
- value is hostname of authoritative name server for this domain
- **(sastra.edu, dns.sastra.edu, NS)**

type=MX

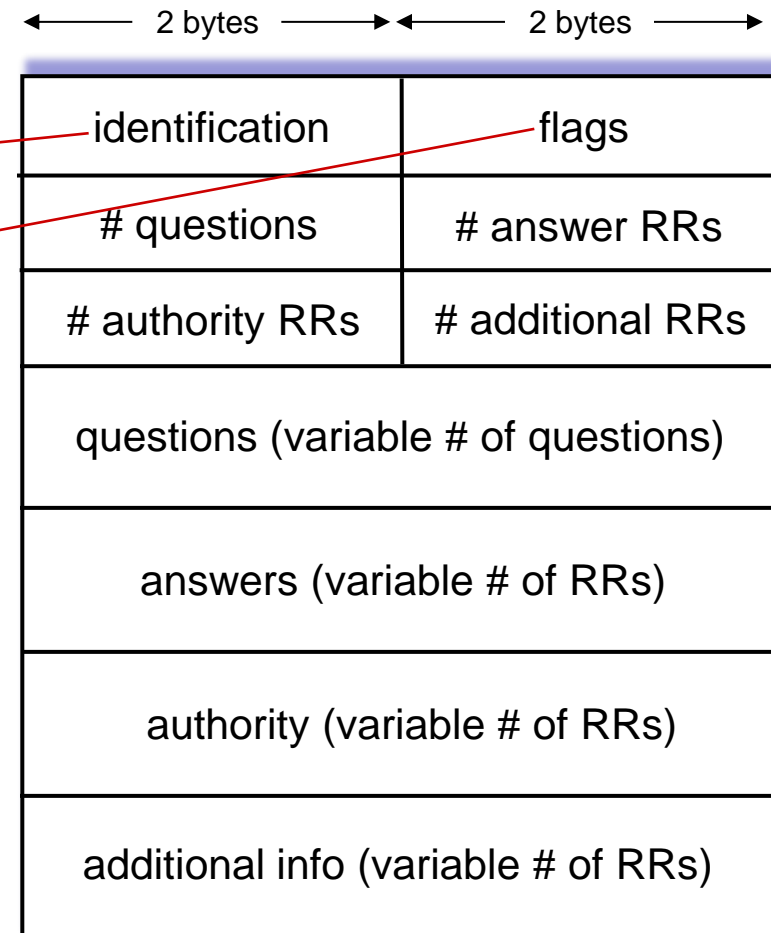
- value is canonical name of SMTP mail server associated with name
- **(Sastra.edu, mail.univ.sastra.edu, MX)**

DNS protocol messages

DNS *query* and *reply* messages, both have same *format*:

message header:

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



DNS protocol messages

DNS *query* and *reply* messages, both have same *format*:

← 2 bytes → ← 2 bytes →

identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable # of questions)	
answers (variable # of RRs)	
authority (variable # of RRs)	
additional info (variable # of RRs)	

name, type fields for a query

RRs in response to query

records for authoritative servers

additional “helpful” info that may be used

Getting your info into the DNS

example: new startup “sastra”

- register name sastra.edu at *DNS registrar* (e.g., Network Solutions)
 - provide names, IP addresses of authoritative name server (primary and secondary)
 - registrar inserts NS, A RRs into .edu TLD server:
(sastra.edu, dns1.sastra.edu, NS)
(dns1.sastra.edu, 212.212.212.1, A)
- create authoritative server locally with IP address 212.212.212.1
 - type A record for www.sastra.edu
 - type MX record for sastra.edu

DNS Security & Challenges

- **DNS Spoofing/Poisoning** – Attackers manipulate DNS responses to redirect users to malicious sites.
- **DDoS Attacks** – Attackers flood DNS servers to disrupt internet services.
- **Privacy Issues** – Traditional DNS queries are not encrypted, allowing ISPs to monitor browsing activity.



Security Solutions:

- ✓ **DNSSEC (DNS Security Extensions)** – Protects against DNS spoofing.
- ✓ **DNS over HTTPS (DoH) & DNS over TLS (DoT)** – Encrypts DNS queries for privacy.

Top Public DNS Servers

Provider	Primary DNS	Secondary DNS	Features
Google DNS	8.8.8.8	8.8.4.4	Fast, reliable, globally distributed
Cloudflare	1.1.1.1	1.0.0.1	Privacy-focused, no logging
OpenDNS	208.67.222.222	208.67.220.220	Security filtering, parental controls
Quad9	9.9.9.9	149.112.112.112	Blocks malicious domains, privacy-first
Comodo	8.26.56.26	8.20.247.20	Malware protection
CleanBrowsing (Family Filter)	185.228.168.168	185.228.169.168	Blocks adult content & malicious sites

free, fast, and often more secure or private than your ISP's default DNS.

<https://www.nslookup.io/>

Application layer: overview

1. WEB & HTTP
2. FTP
3. E-mail
4. DNS - The Internet's Directory Service
5. DDNS
6. TELNET
7. BLUETOOTH
8. FIREWALLS
9. SNMP

DDNS (Dynamic Domain Name System)

DDNS (Dynamic Domain Name System) is a service that automatically updates a domain name's DNS records when the IP address of a device or network changes.

It is commonly used for:

- **Home Networks:** Assigning a stable domain name to a dynamically changing IP address from an ISP.
- **Remote Access:** Allowing users to access home or office networks remotely using a consistent domain name.
- **Hosting Services:** Running web servers, game servers, or CCTV systems on a dynamic IP.

How DDNS Works:

1. A client device (router, computer, or software) detects an IP address change.
2. It sends the updated IP to the DDNS provider.
3. The DDNS provider updates the DNS record, ensuring the domain always points to the latest IP.

DNS vs. DDNS: Key Differences

Feature	DNS (Domain Name System)	DDNS (Dynamic DNS)
Definition	Resolves domain names to IP addresses for websites and online services.	Updates domain name mappings dynamically when IP addresses change.
IP Address Type	Works with static IP addresses .	Designed for dynamic IP addresses (changing IPs from ISPs).
Use Case	Used for websites, emails, and online services with permanent IPs.	Used for remote access to devices, home servers, security cameras, etc.
Update Frequency	IP-to-domain mapping is manually updated .	Updates automatically when the IP changes.
Who Uses It?	Web hosting, enterprise networks, cloud services.	Home users, small businesses, remote workers.
Configuration	Managed via domain registrars and DNS providers.	Requires DDNS client software or router support.
Security	Uses DNSSEC for security but can be targeted by DNS spoofing attacks.	Can be vulnerable to unauthorized updates if not secured properly.

When to Use Each?

- ✓ **Use DNS** if you have a **static IP** (e.g., hosting a website on a fixed server).
- ✓ **Use DDNS** if your ISP assigns a **dynamic IP** and you need remote access (e.g., home security cameras, personal web servers).

Popular DDNS Providers:

- No-IP
- DynDNS
- DuckDNS
- Cloudflare (with API updates)

TELNET (TELecommunication NETwork)

- **TELNET** is a network protocol used for remote communication with another device over a TCP/IP network.
- It allows users to establish a command-line interface (CLI) session with a remote computer, usually for administrative tasks.

Key Features of TELNET:

- Provides a **text-based** interface for remote access.
- Operates on **port 23** by default.
- Supports bidirectional communication.
- Lacks encryption, making it **insecure** for modern use.

Telnet is not secure—it transmits data in plaintext. For secure communication, use SSH instead.

How TELNET Works:

1. A **TELNET client** connects to a **TELNET server** using an IP address or hostname.
2. The server prompts for a **username and password**.
3. Once authenticated, users can execute commands on the remote system as if they were physically present.

TELNET Commands

- telnet <IP_Address> → Connect to a remote system.
- quit → Exit TELNET session.
- open <IP_Address> → Open a new connection.
- close → Close the current connection.

Security Concerns & Alternatives:

- **TELNET transmits data in plaintext**, making it vulnerable to **eavesdropping and attacks**.
- **SSH (Secure Shell)** is the preferred alternative as it encrypts communication.

Bluetooth

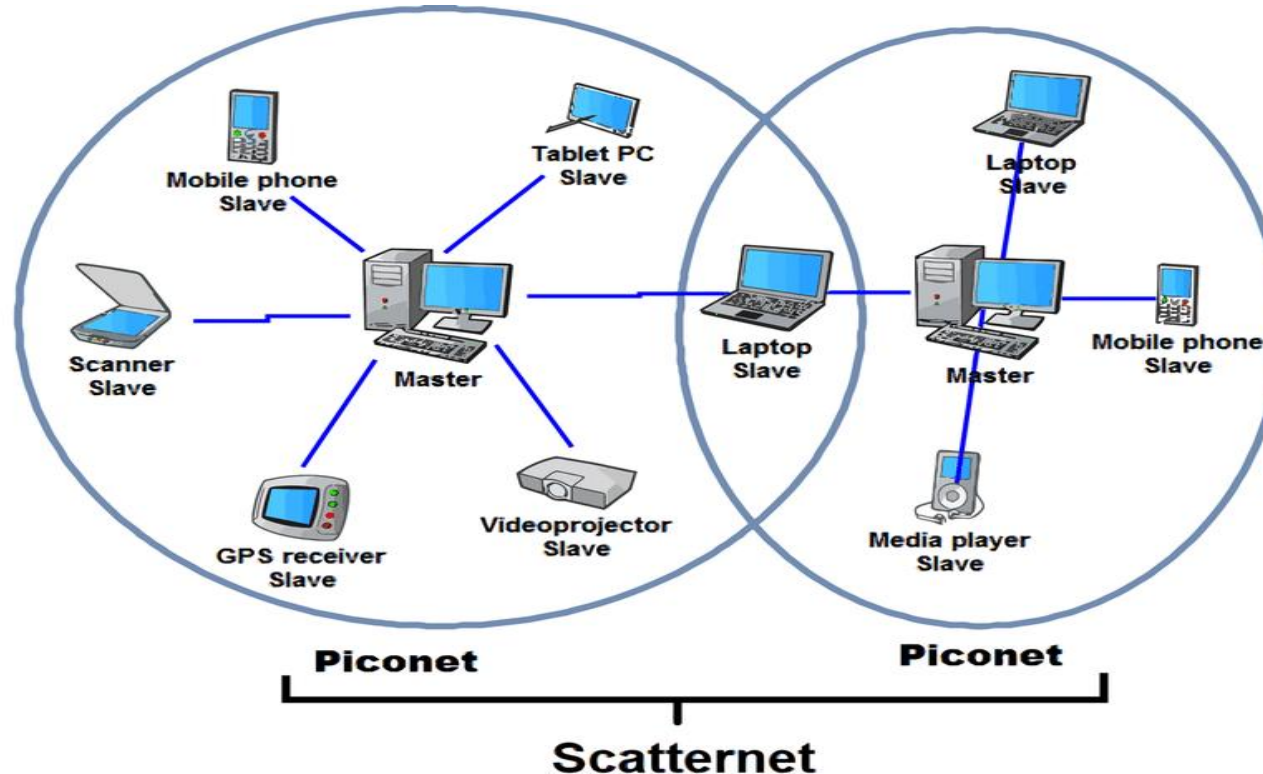
- Bluetooth is a short-range wireless communication technology that enables devices to exchange data over short distances using radio waves. (usually up to 10 meters or ~30 feet).
- It operates in the **2.4 GHz ISM** (Industrial, Scientific, and Medical) band and is widely used for wireless connectivity in consumer electronics, healthcare, automotive, and industrial applications.
- It's commonly used to connect:
 - Wireless headphones, keyboards, and mice
 - Smartphones and smartwatches
 - File transfers between phones or computers
 - Car audio systems

Key Features of Bluetooth:

- ✓ **Short-range communication** (typically up to **10 meters**, but can extend up to **100 meters** with high-power devices).
- ✓ **Low power consumption**, making it ideal for battery-operated devices.
- ✓ **Supports voice, data, and multimedia transmission.**
- ✓ **Uses frequency hopping** (1,600 hops/sec) to reduce interference.
- ✓ **Enables peer-to-peer and networked communication** (via piconets and scatternets).

Bluetooth Architecture:

- 1. Piconet** – A small network of Bluetooth devices with **one master** and up to **seven active slaves**.
- 2. Scatternet** – Multiple interconnected piconets where devices can communicate across different networks.



All communication is controlled by the master. Slaves cannot talk to each other directly — only through the master.

Piconet

Imagine your phone is connected via Bluetooth to:

- Wireless headphones
- A smartwatch
- A fitness tracker
- Your **phone acts as the master**, and the others are **slaves**, forming a **piconet**.

Scatternet

- A Scatternet is when multiple piconets are connected.
- A device can act as a slave in one piconet and a master in another, linking them together.
- It allows Bluetooth devices to **participate in more than one piconet at the same time**, enabling more complex and larger Bluetooth networks.

A device can act as a: Master in one piconet, Slave in another piconet.
This shared device links the two piconets, forming a Scatternet.

Scatternet - Example

Imagine:

- Your **laptop** is connected to a **Bluetooth keyboard and mouse** (Piconet 1).
- The same **laptop** is also acting as a **master to your phone** for file transfer (Piconet 2).
- Your laptop becomes a **bridge device**, forming a **Scatternet**.

Bluetooth Versions & Improvements:

- **Bluetooth 1.0 & 1.1** – Basic functionality with slow speeds (~1 Mbps).
- **Bluetooth 2.0 + EDR** – Enhanced Data Rate (EDR) up to **3 Mbps**.
- **Bluetooth 3.0 + HS** – High-Speed (HS) mode using Wi-Fi (~24 Mbps).
- **Bluetooth 4.0 (BLE)** – Introduced **Bluetooth Low Energy (BLE)** for IoT and wearable devices.
- **Bluetooth 5.0** – Increased range (up to **240 meters**) and speed (up to **2 Mbps**).
- **Bluetooth 5.1 & 5.2** – Improved location tracking and audio capabilities.
- **Bluetooth 5.3 & 5.4** – Enhanced energy efficiency, security, and connectivity.

Common Bluetooth Applications:

- **Wireless Audio** – Headphones, earbuds, speakers.
- **File Transfer** – Mobile devices, computers.
- **Wearable Devices** – Smartwatches, fitness trackers.
- **Automotive Connectivity** – Hands-free calling, infotainment.
- **IoT & Smart Home** – Smart locks, lights, and appliances.
- **Medical Devices** – Heart rate monitors, glucose meters.

Firewalls

- A **firewall** is a network security system that monitors and controls incoming and outgoing network traffic based on predefined security rules.
- It acts as a barrier between trusted internal networks and untrusted external networks (such as the internet) to **prevent unauthorized access and cyber threats**.

Functions of Firewalls

- ✓ Block unauthorized access to internal networks.
- ✓ Prevent malware and cyberattacks (e.g., DoS, ransomware).
- ✓ Filter network traffic based on rules.
- ✓ Monitor logs and generate security reports.
- ✓ Control user access to specific websites or applications.

Types of Firewalls

◆ Packet-Filtering Firewalls

- Works at the **Network Layer (Layer 3)** of the OSI model.
- **Filters packets based on IP addresses, ports, and protocols.**
- Example: **Access Control Lists (ACLs)** in routers.

◆ Stateful Inspection Firewalls

- Works at the **Transport Layer (Layer 4)**.
- **Tracks the state of active connections and allows only legitimate traffic.**
- More secure than packet-filtering firewalls.

◆ Proxy Firewalls

- Works at the **Application Layer (Layer 7)**.
- **Intercepts and inspects traffic before forwarding it to the destination.**
- Example: **Squid Proxy, Apache Proxy.**

Contd..

- ◆ **Next-Generation Firewalls (NGFWs)**
 - Combines **Deep Packet Inspection (DPI)**, **Intrusion Prevention Systems (IPS)**, and **AI-based threat detection**.
 - Example: **Palo Alto Networks, Fortinet, Cisco Firepower**.
- ◆ **Cloud Firewalls**
 - Hosted in the cloud to protect cloud-based applications and services.
 - Example: **AWS WAF, Azure Firewall**.

Firewall Deployment Methods

- **Hardware Firewalls** – Standalone devices placed between a network and the internet.
- **Software Firewalls** – Installed on individual devices (Windows Defender Firewall, Linux iptables).
- **Cloud-Based Firewalls** – Protects cloud applications and services.

Simple Network Management Protocol (SNMP)

- It is a widely used **protocol for network management** that allows devices like routers, switches, servers, workstations, printers, and more to be monitored and managed from a central location.
- **SNMP is a UDP-based network protocol.**
- SNMP was introduced in 1988

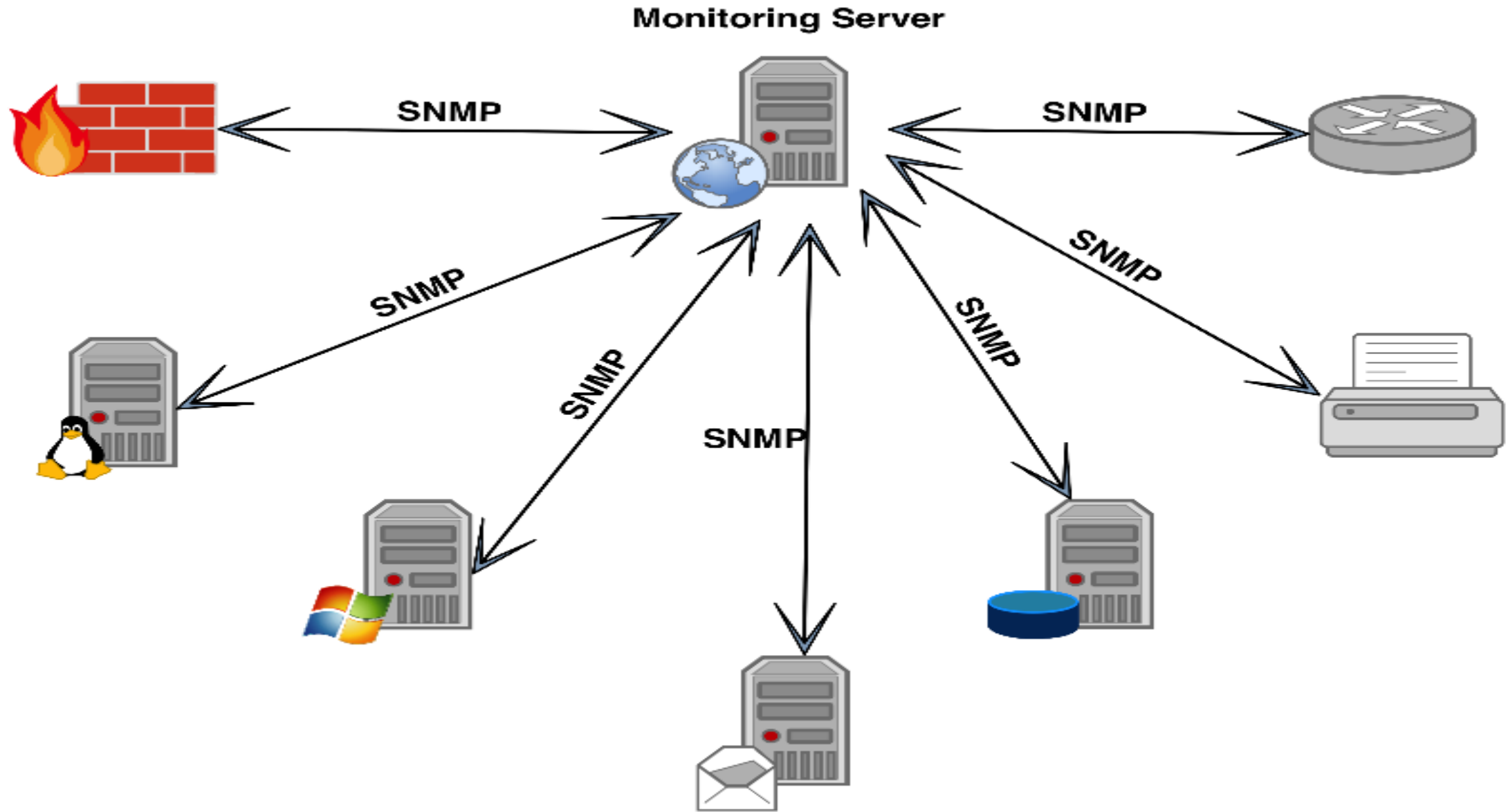
Key Features of SNMP:

- 1. Protocol Type:** Application-layer protocol (uses UDP typically on ports 161 for queries and 162 for traps).
- 2. Management Architecture:** Based on a **manager-agent** model.
- 3. Lightweight:** Designed to be simple and require minimal resources.
- 4. Standardized:** Part of the TCP/IP protocol suite.

Uses of SNMP

- Gives administrators the ability to change the state of some SNMP-based device.
- For example, you can use SNMP to shut down an interface on your router or check the speed at which your Ethernet interface is operating.
- SNMP can even monitor the temperature on your switch and warn you when it is too high.
- SNMP can be used to manage Unix systems, Windows systems, printers, modem racks, power supplies, and more.

How Does SNMP Work?



SNMP Components

SNMP Manager

Centralised Software for Network Management

SNMP Agent

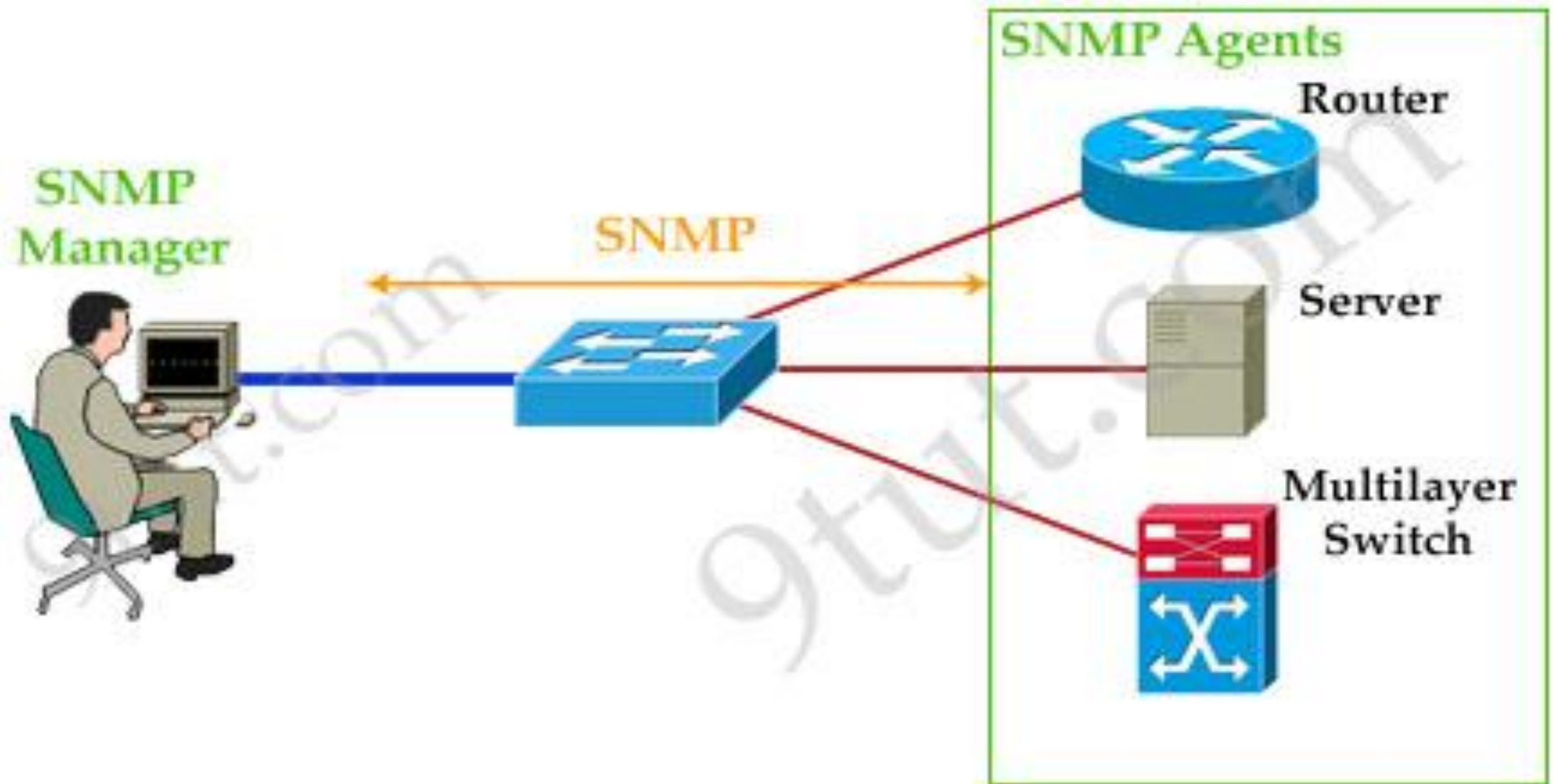
Network Elements (a SW on them) that are managed e.g. routers, switches, servers, hosts, printers, UPSs, etc...

MIB

SNMP Database that makes sure that the data exchange between the Manager and the Agent remains structured

SIMPLE NETWORK MANAGEMENT PROTOCOL





Components of SNMP:

Component

SNMP Manager

SNMP Agent

MIB (Management Information Base)

OID (Object Identifier)

Trap

Description

Central system that queries agents, collects data, and can send configuration commands.

Software running on network devices that collects and stores management data.

A virtual database schema that defines what can be queried or controlled.

Unique identifier for each variable in the MIB.

An alert message sent from agent to manager, indicating an event (like error or failure).

What is a MIB?

- A **Management Information Base (MIB)** is a **collection of hierarchically organized information** used by SNMP to manage devices in a network.
- It defines **OIDs (Object Identifiers)** and the **types of information** that an SNMP agent can provide.

MIB Example: SampleDevice-MIB

SampleDevice-MIB DEFINITIONS ::= BEGIN

IMPORTS

OBJECT-TYPE, MODULE-IDENTITY, enterprises FROM SNMPv2-SMI

DisplayString FROM SNMPv2-TC;

sampleDevice MODULE-IDENTITY

LAST-UPDATED "202504110000Z"

ORGANIZATION "Sample Corp"

CONTACT-INFO

"Email: support@samplecorp.com"

DESCRIPTION

"MIB for SampleDevice SNMP Agent"

::= { enterprises 99999 }

deviceInfo OBJECT IDENTIFIER ::= { sampleDevice 1 }

deviceName OBJECT-TYPE

SYNTAX DisplayString

MAX-ACCESS read-only

STATUS current

DESCRIPTION "The name of the device"

::= { deviceInfo 1 }

deviceUptime OBJECT-TYPE

SYNTAX INTEGER

MAX-ACCESS read-only

STATUS current

DESCRIPTION "Device uptime in minutes"

::= { deviceInfo 2 }

deviceStatus OBJECT-TYPE

SYNTAX INTEGER { ok(1), warning(2), critical(3) }

MAX-ACCESS read-only

STATUS current

DESCRIPTION "Current status of the device"

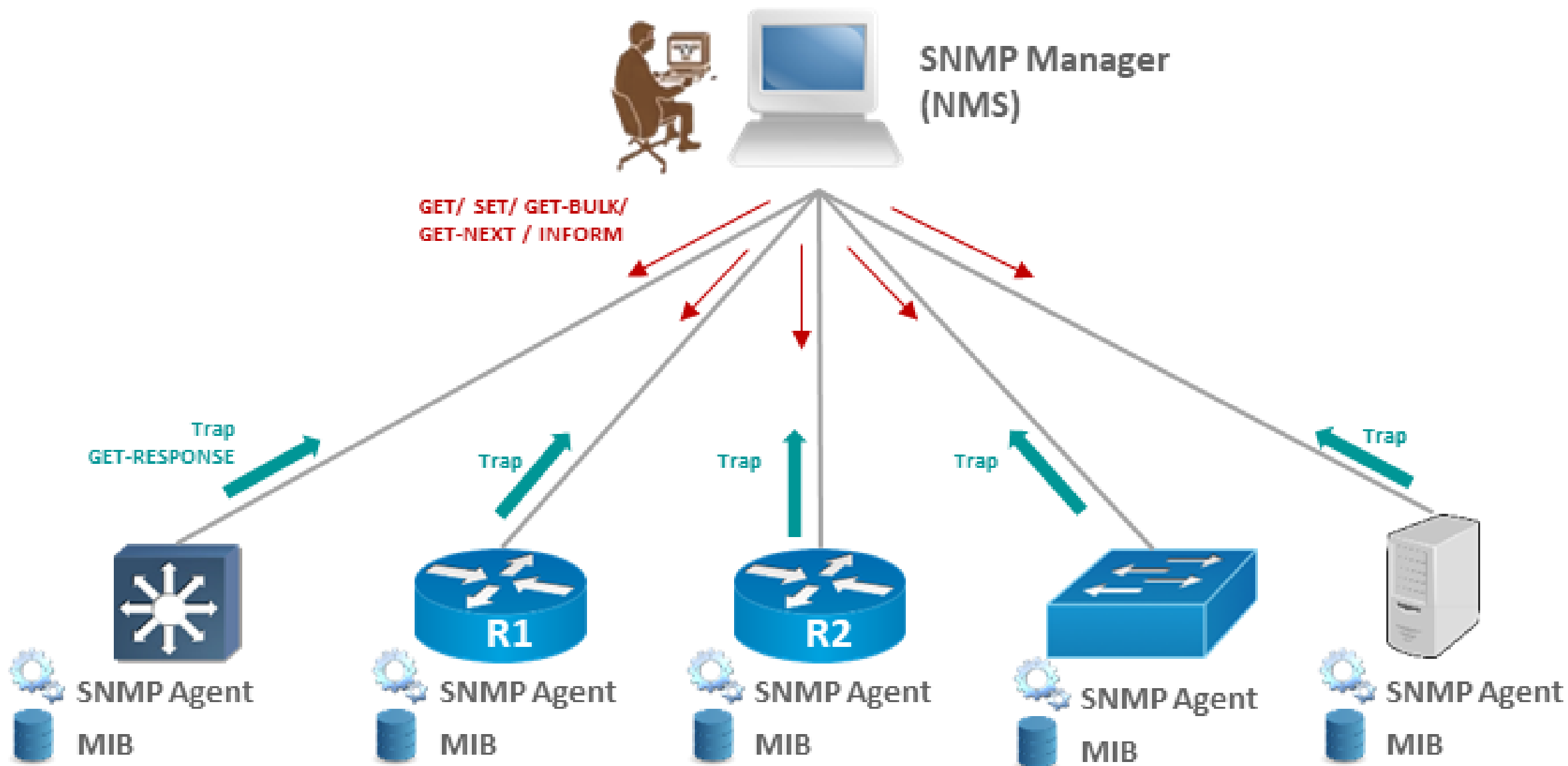
::= { deviceInfo 3 }

END

OID Mapping: An Example

Object Name	OID Path	Description
deviceName	.1.3.6.1.4.1.99999.1.1	Name of the device
deviceUptime	.1.3.6.1.4.1.99999.1.2	Uptime in minutes
deviceStatus	.1.3.6.1.4.1.99999.1.3	Status: ok(1), warning(2), critical(3)

- .1 – ISO
- .3 – org
- .6 – dod
- .1 – internet
- .4 – private
- .1 – enterprises
- .99999 – Sample Corp (fictional enterprise ID)
- .1 – deviceInfo branch
- .1 – deviceName



Operations in SNMP

- There are three basic operations that are used in SNMP:
 1. Managed devices can alert the NMS
 2. NMS can ask the managed devices for info about their present state
 3. NMS can inform the managed devices to modify aspects of their configuration

3 Versions of SNMP

- **SNMP Version 1 (SNMPv1)** - First implementation of SNMP, and it supports 32 bits at a time. Less secure than what the newer versions use.
- **SNMP Version 2 (SNMPv2)** - replaces the 32-bit counters with 64-bit ones. Has the same issues that come with community strings.
- **SNMP Version 3 (SNMPv3)** - Version 3 comes with a combination of authentication and encryption options, which allows it to prevent unauthorized access, as well as attempts by hackers to spy on communications.
- As a result, SNMPv3 is more secure than the previous two versions.

NOTE

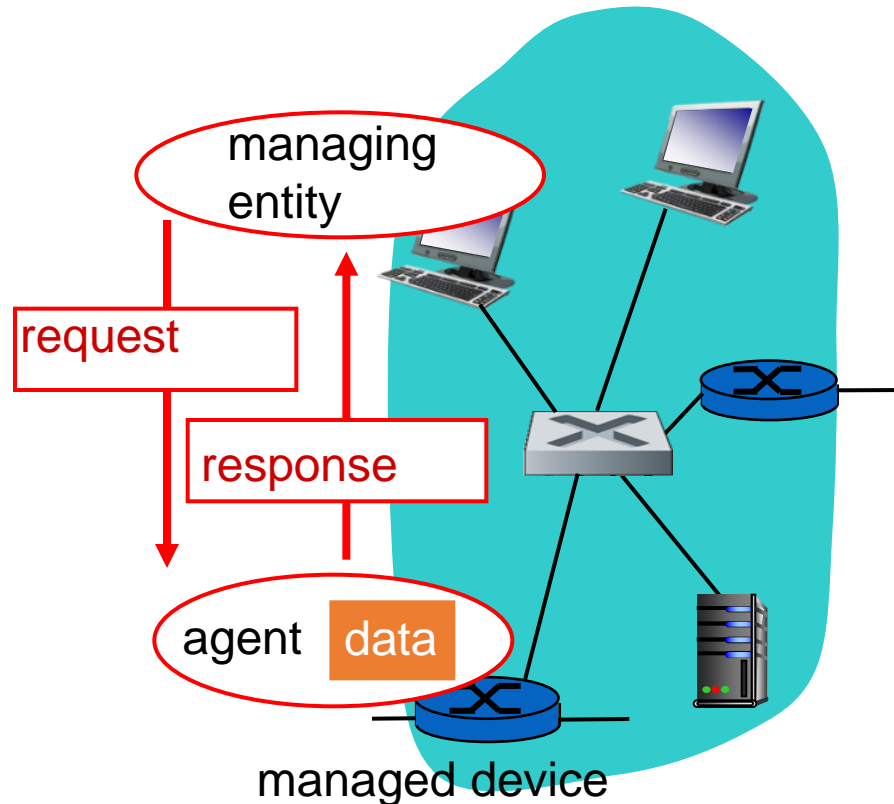
- SNMP agents use a UDP port 161, while the manager uses a UDP port 162.
- The current SNMP version is SNMPv3.
- The prior versions, SNMPv1 and SNMPv2 are considered obsolete and should not be used.

SNMP protocol

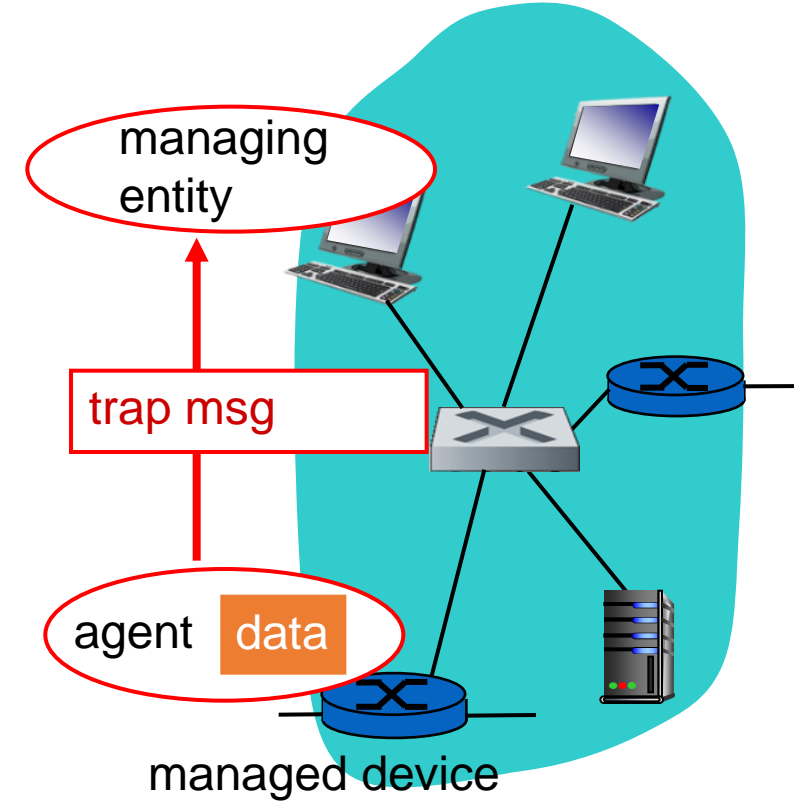
Two ways to convey MIB info, commands:

Agent can send an unrequested message to the manager to notify about an important event.

For Eg: link going down, an authentication or a power failure



request/response mode



trap mode

SNMP protocol: message types

<u>Message type</u>	<u>Function</u>
GetRequest GetNextRequest GetBulkRequest	Mgr-to-agent: “get me data” (instance,next in list, block)
InformRequest	Mgr-to-Mgr: here’ s MIB value
SetRequest	Mgr-to-agent: set MIB value
Response	Agent-to-mgr: value, response to Request
Trap	Agent-to-mgr: inform manager of exceptional event

SNMP Operations:

- GET: Retrieve a value from the agent.
- GET-NEXT: Retrieve the next value in the MIB hierarchy.
- SET: Modify a value on the agent.
- TRAP: Notify the manager of an event (asynchronous).
- INFORM: Like a trap, but with acknowledgment.
- GET-BULK: Retrieve large blocks of data (SNMPv2 and later).

Use Cases:

- Monitoring bandwidth and traffic.
- Detecting network faults.
- Configuring network devices.
- Collecting performance metrics.
- Logging and auditing events.

SNMP Manager Software

Tool Name	Description	Platform	Key Features
PRTG Network Monitor	Commercial, all-in-one monitoring suite	Windows	Auto-discovery, SNMP monitoring, real-time graphs
Nagios XI / Core	Popular open-source monitoring system with SNMP plugins	Cross-platform	Custom alerts, SNMP traps, dashboard support
Zabbix	Enterprise-class open-source monitoring tool	Linux / Windows	SNMPv1/v2/v3 support, templates, rich visualization
SolarWinds Network Performance Monitor	Comprehensive commercial monitoring software	Windows	SNMP-based discovery, performance analysis
ManageEngine OpManager	Network performance monitoring and diagnostics	Windows / Linux	SNMP, NetFlow, traps, and config management
OpenNMS	Open-source network monitoring platform	Cross-platform	Distributed polling, trap processing, SNMP graphs
The Dude (by MikroTik)	Free network management tool	Windows / Linux (Wine)	Network mapping, SNMP polling, alerts
WhatsUp Gold	Commercial network monitoring software	Windows	SNMP monitoring, real-time maps, reports

SNMP Agent Software

Tool Name	Description	Platform	Key Features
Net-SNMP Agent	Popular open-source SNMP agent suite	Linux, UNIX, Windows	Supports SNMPv1, v2c, v3; extensible with scripts
SNMP Agent Builder (iReasoning)	SNMP agent creation toolkit	Windows	Custom MIB support, GUI-based development
SNMP Research AgentX / EMANATE	Commercial SNMP agent software	Cross-platform	Supports AgentX protocol, secure SNMPv3
Windows SNMP Service	Built-in SNMP agent for Windows	Windows	Enables SNMP on Windows servers/workstations
OpenNMS Minion	Lightweight SNMP agent for edge locations	Linux / containers	Remote collection of SNMP, syslog, and flows
ManageEngine SNMP Agent Simulator	Simulates SNMP agents for testing	Windows / Linux	Useful for training and development
SimpleAgentPro	SNMP agent simulator	Cross-platform	Used for protocol testing and training environments

Basic concepts of Cryptography

- Cryptography is the science of securing information by transforming it into a secure format so that only authorized parties can access it.
- It ensures confidentiality, integrity, authentication, and non-repudiation.

Goals of Cryptography (CIA Triad)

1. **C**onfidentiality – Only authorized users can access the information.
2. **I**ntegrity – Data has not been altered or tampered with.
3. **A**vailability – Information is accessible when needed.

Additional Goals:

- **Authentication** – Verifying the identity of the person or system.
- **Non-repudiation** – Ensures a sender cannot deny sending the message.

Core Concepts of Cryptography

1. **Plaintext** : The original readable message or data that needs to be protected.
2. **Ciphertext** : The scrambled and unreadable version of the plaintext, produced after encryption.
3. **Encryption** : The process of converting plaintext into ciphertext using an algorithm and a key.
4. **Decryption** : The reverse process of converting ciphertext back into plaintext using a key.
5. **Key** : A piece of information used in the encryption and decryption process.

There are two main types:

- **Symmetric Key**: Same key for encryption and decryption.
- **Asymmetric Key**: Uses a **public key** for encryption and a **private key** for decryption.

Types of Cryptography

1. Symmetric Cryptography (Secret Key)

- Same key is used for both encryption and decryption.
- Fast but less secure if the key is intercepted.
- Example: **AES, DES, RC4**

2. Asymmetric Cryptography (Public Key)

- Uses a **public key** for encryption and a **private key** for decryption.
- More secure but slower.
- Example: **RSA, ECC**

3. Hash Functions

- One-way functions that convert data into a fixed-size string (hash).
- Used for data integrity.
- Example: **SHA-256, MD5**

Popular Cryptographic Algorithms

Type

Symmetric

Asymmetric

Hash

Algorithms

AES, DES, 3DES, Blowfish

RSA, ECC, Diffie-Hellman

SHA-1, SHA-2, SHA-3, MD5

Applications of Cryptography

- Secure communication (e.g., HTTPS, VPN)
- Digital signatures
- Blockchain and cryptocurrency
- Secure file storage
- Email security (e.g., PGP)

#1

1. Which of the following ensures that information is not altered during transmission?

- A. Confidentiality
- B. Authentication
- C. Integrity
- D. Non-repudiation

#2

2. What is the goal of cryptography that ensures only the intended recipient can read the message?

- A. Authentication
- B. Integrity
- C. Confidentiality
- D. Availability

#3

3. Which cryptographic goal is achieved by using digital signatures?

- A. Availability
- B. Non-repudiation
- C. Confidentiality
- D. Integrity

#4

4. A fingerprint scanner on a phone is an example of which cryptographic goal?

- A. Confidentiality
- B. Authentication
- C. Availability
- D. Integrity

THE END