World Wide Web and Hypertext Transfer Protocol

World Wide Web – an architectural framework for accessing linked documents spread all over Internet

Developed at CERN (Switzerland) and MIT (USA)

http://www.w3.org consortium's home page

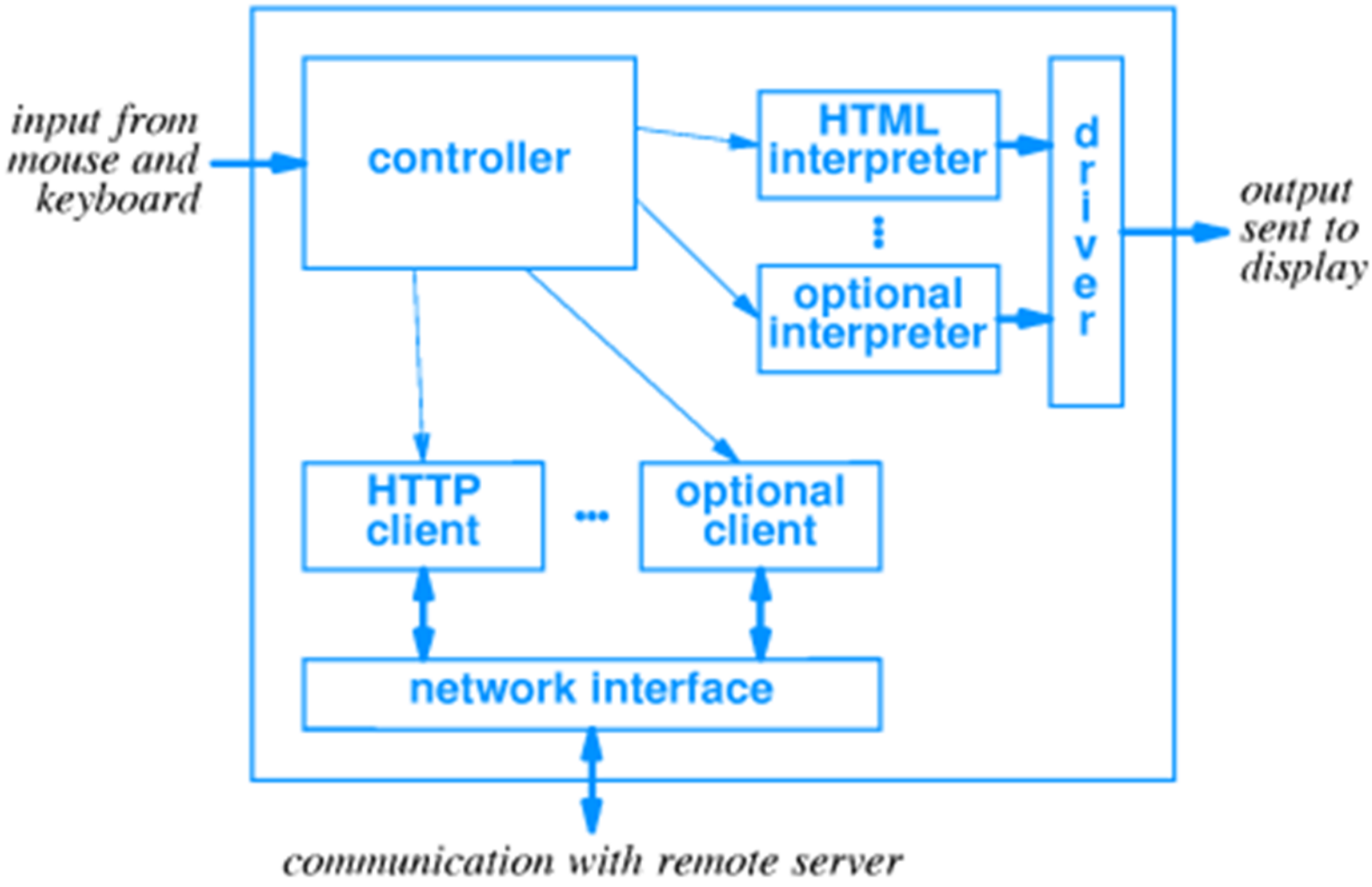
General Notions

- www: basic a client/server system
- *Hypermedia* system: allows interactive access to collections of documents
- Document can hold:
  - Text (*hypertext*, if containing links to other texts)
  - Graphics
  - Sound
  - Animations
  - Video

- Documents linked together
  - Non-distributed - all documents stored locally (like CD-ROM)
  - Distributed - documents stored on remote servers
- Link represented by "active area" on screen
  - Graphic - button
  - Text - highlighted
- Selecting link will fetch referenced document for display
- Links may become invalid
- Link is simply a text name for a remote document
- Remote document may be removed while name in link remains in place
- Use of local disks for caching pages
- Pages are viewed with programs called **browsers** (Netscape, Explorer, Chrome, Mozilla)
  - interactive, "point-and-click" interface to hypermedia documents

- Browser has more components (see next figure):
  - Display driver for painting screen
  - HTML interpreter for HTML-formatted documents
  - Other interpreters (e.g., Shockwave) for other items
  - HTTP client to fetch HTML documents from WWW server
  - Other clients for other protocols (e.g., ftp)
  - Controller to accept input from user
- Must be multi-threaded

Browser Components



- Downloading documents from servers may be **slow**
  - Internet congested
  - Dialup connection
  - Server busy
- Returning to previous (HTML) document requires reload from server
- Local ***cache*** can be used to hold copies of visited pages
- Also can implement organizational *HTTP proxy* that caches documents for multiple users

Document representation

Each www document is called a *page*

•Initial page for individual or organization is called a *home page*

•Page can contain many different types of information; page must specify

- •Content
- •Type of content
- •Location
- •Links

•Rather than fixed WYSIWYG representation (e.g., Word), pages are formatted with a *mark up language* (like TeX)

•Allows browser to reformat, to fit display

•Allows text-only browser to discard graphics

•Standard is **HyperText Markup Language (HTML)**

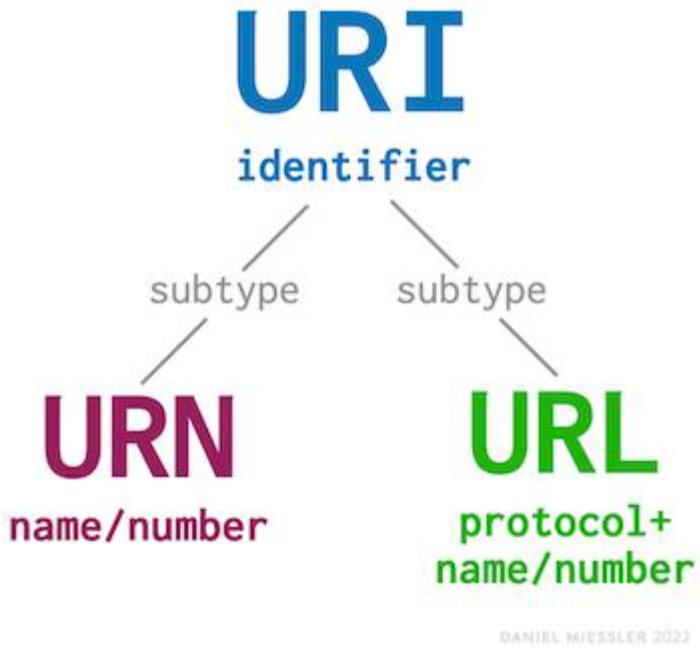
-Markup: everything in a document that is not content

- Page identified by:

  Protocol used to access page, computer on which page is stored, TCP port to access page (optional) and pathname of the file on server

- Each link is specified in HTML
- Item on page is associated with another HTML document
- Need for mechanisms for naming, locating & accessing pages
  - Specific syntax for page's worldwide name: ***Uniform Resource Locator*** (URL): *protocol://computer_name:port/document_name*
    - *protocol* can be http, https, ftp, file, mailto
    - *computer name* is DNS name
    - (Optional) *port* is TCP port
    - *document_name* is path on computer to page

Generalized URL, making possible page replication (don't care where getting page)
Universal Resource Identifier (URI), allowing less resources for common pages

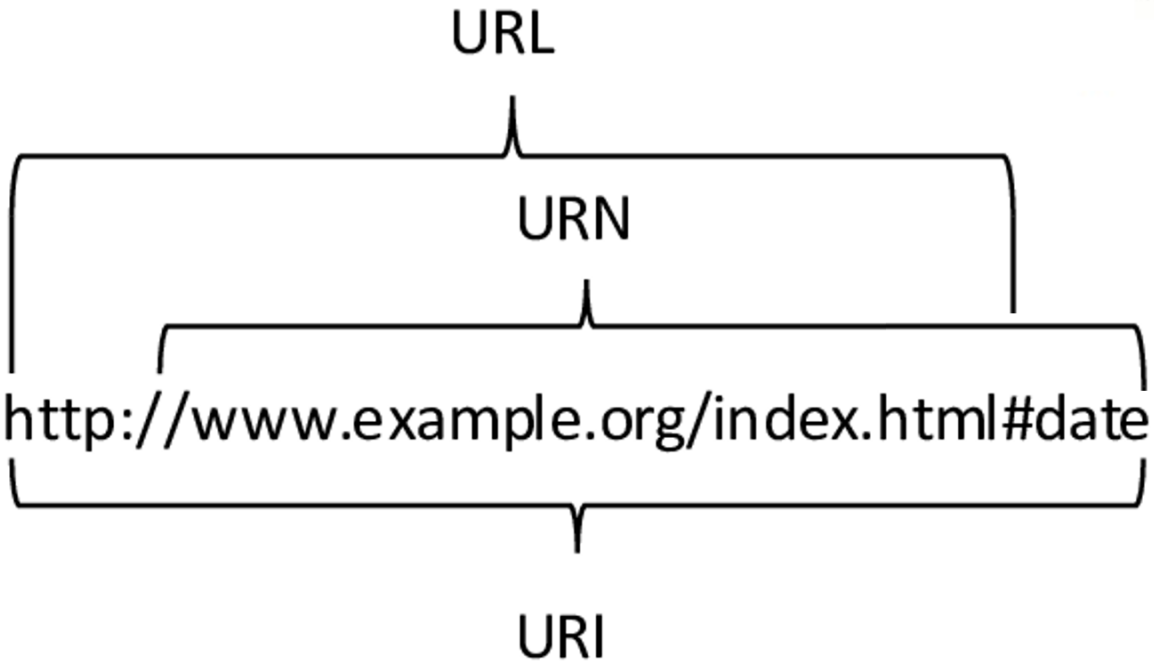


Uniform Resource Locator (URL)
Uniform Resource Name (URN)

URI which specifies the location is URL

URI which specified the name is URN

URI which specifies both name and location is URI



HTML – short survey

Evolved from **Standard Generalized Markup Language (SGML),** specialized for hypertext and adapted to the Web

HTML specifies how documents are to be formatted

- Major structure of document
- Formatting instructions
- Hypermedia links
- Additional information about document contents

- Two parts to document:
  - *Head* contains details about the document
  - *Body* contains information/content
- Page is represented in ASCII text with embedded HTML *tags* formatting instructions
- Tags have format <TAGNAME>
- End of formatted section is </TAGNAME>
- Commands inside the tags: directives

Main HTML Tags

Tag	Description
<html> ... </html>	Declares the Web page to be written in HTML
<head> ... </head>	Delimits the page's head
<title> ... </title>	Defines the title (not displayed on the page)
<body> ... </body>	Delimits the page's body
<h *n*> ... </h*n*>	Delimits a level *n* heading
...	Set ... in boldface
...	Set ... in italics
...	Center ... on the page horizontally
...	Brackets an unordered (bulleted) list
...	Brackets a numbered list
...	Brackets an item in an ordered or numbered list
 	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a horizontal rule
<img src="...">	Displays an image here
 ... 	Defines a hyperlink

HTML Evolution

HTML 1.0 – one way, e.g. users could only call up pages, hard to send back information

⇒Inclusion of **forms**, containing boxes or buttons, allowing users for info filling or make choices and sending info back to the page's owner

Form enclosed between <FORM> and </FORM> tags

One standard to handle forms' data: Common Gateway Interface (CGI)

Example: CGI programs (scripts) allow interface between a data base and the Web

Dynamic Web pages (higher interactivity between user & pages, continuous page updating) are designed using other methodologies, as:

-server side scripting technologies (JSP – Java Server Pages, ASP – Active Server Pages, PHP – Perl Helper Pages, ColdFusion)

-active Web documents (moves computation to the browser), using Java technologies (JavaScripts)

Hypertext Transfer Protocol (HTTP)

Underlying protocol of the World Wide Web

Not a protocol for transferring hypertext

- For transmitting information with efficiency necessary for hypertext jumps
- HTTP specifies commands and client-server interaction

Can transfer plain text, hypertext, audio, images, and Internet accessible information

HTTP Overview

Transaction oriented client/server protocol

Usually between Web browser (client) and Web server

Uses TCP connections

Stateless

- Each transaction treated independently
- Each new TCP connection for each transaction
- Terminate connection when transaction complete

Client/Server model:

- *client:* browser that requests, receives, "displays" WWW objects
- *server:* WWW server daemon, sends objects in response to client requests

Functionality:

- 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 WWW server (HTTP server)
- TCP connection closed

HTTP is "stateless": server maintains no information about past client requests

HTTP1.0: RFC 1945

HTTP1.1: RFC 2068

….

HTTP/2: **RFC 7540** (May 2015) -it allows: interleaving of request and response messages on the same connection and uses an efficient coding for HTTP header fields; prioritization of requests, letting more important requests complete more quickly; longer-lived connections; further improving performance

Key Terms

Cache

Client

Connection

Entity

Gateway

Message

Origin server

Proxy

Resource

Server

Tunnel

User agent

Examples of HTTP operations:

-direct connection with origin server; client opens an end-to-end TCP connection and issues a HTTP request. It consists of a specific command (OO: method), a URL & a MIME-like message with request parameters, info about client, etc. Server receives the request, process it & return HTTP response, containing status & error info, MIME-like message with info about server, about response and the body content. Finally the TCP connection is closed.

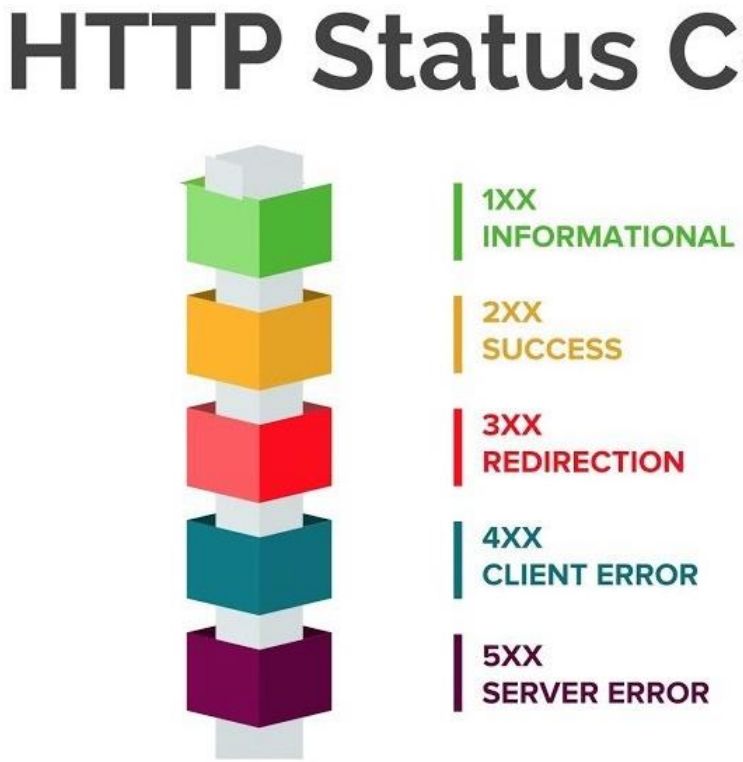
-Intermediate systems with TCP connections (relaying requests & responses)

-Example of a cache: an intermediate system stores previous requests & responses, for handling new requests; no need to access the origin server; caching time limit

HTTP methods and status codes

- GET - request data from a specified resource
- HEAD - almost identical to GET, but without the response body (useful for checking what a GET request will return before actually making a GET request)
- POST - send data to a server to create/update a resource (requests are never cached; do not remain in the browser history/cannot be bookmarked); no restrictions on data type or data length
- PUT - send data to a server to create/update a resource + idempotent characteristic (calling the same PUT request multiple times will always produce the same result)
- DELETE - deletes the specified resource
- CONNECT - requests that the recipient establish a tunnel to the destination origin server identified by the request-target; intended only for use in requests to a proxy
- TRACE - performs a message loop-back test along the path to the target resource, providing a useful debugging mechanism
- OPTIONS - requests information about the communication options available for the target resource, at either the origin server or an intervening intermediary

HTTP methods and status codes



- **1xx (Informational):** The request was received, continuing process
- **2xx (Successful):** The request was successfully received, understood, and accepted
- **3xx (Redirection):** Further action needs to be taken in order to complete the request
- **4xx (Client Error):** The request contains bad syntax or cannot be fulfilled
- **5xx (Server Error):** The server failed to fulfill an apparently valid request

HTTP Messages (Protocol data units)

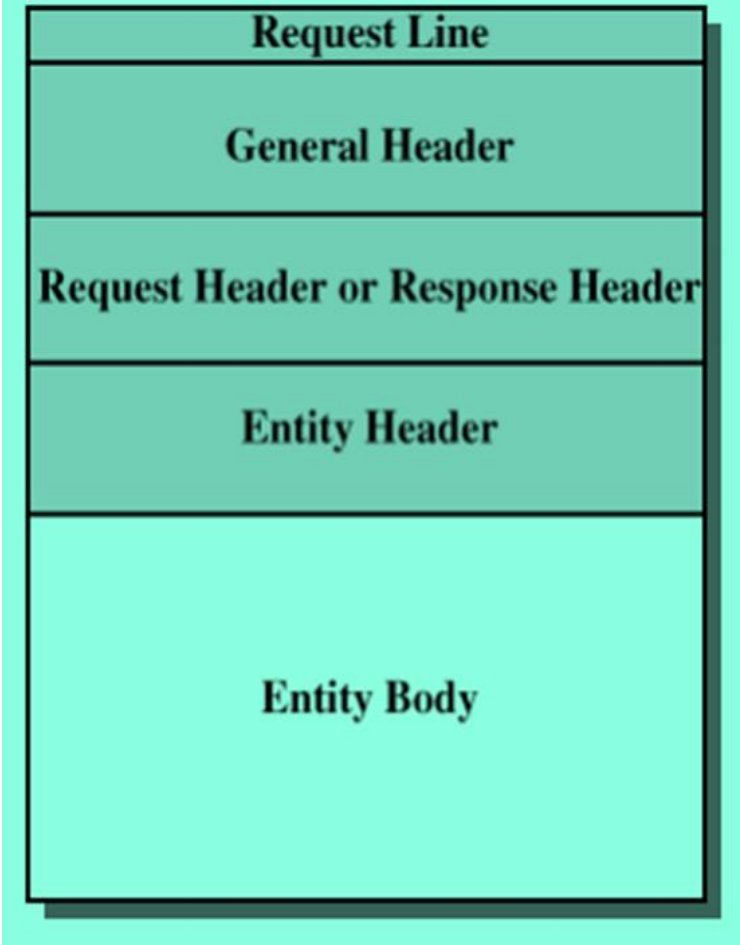
Implementation for:

Requests

Client to server

Responses

Server to client

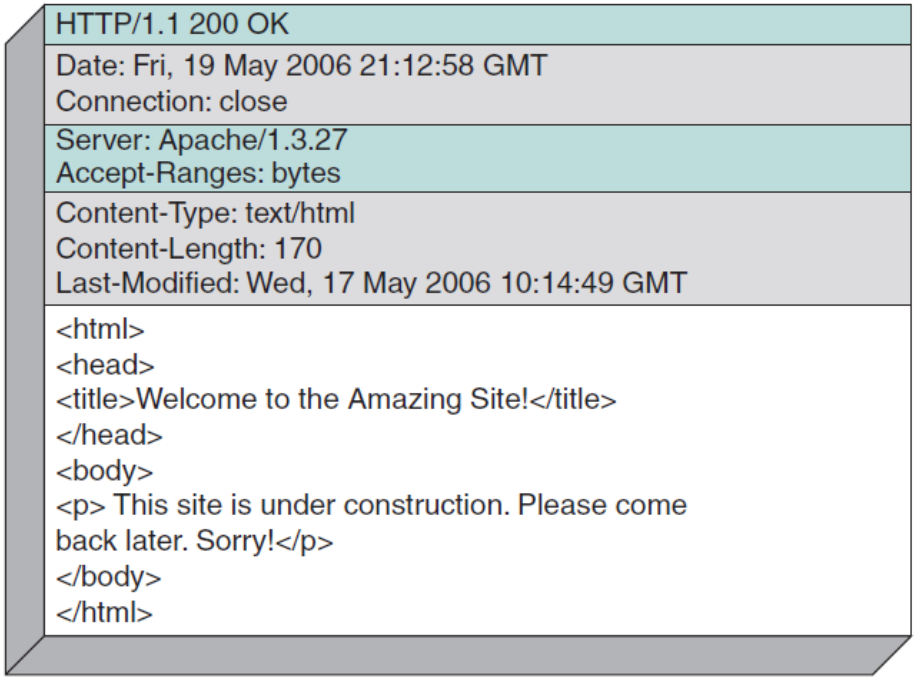


Structure:

- Request line – message type and requested resource
- Response line – status information about message
- General header – applies to both request/response messages
- Request header – info about request and client
- Response header – info about response and server
- Entity header – info about the resource target of the request
- Entity body – body of the message

HTTP Message Format Example

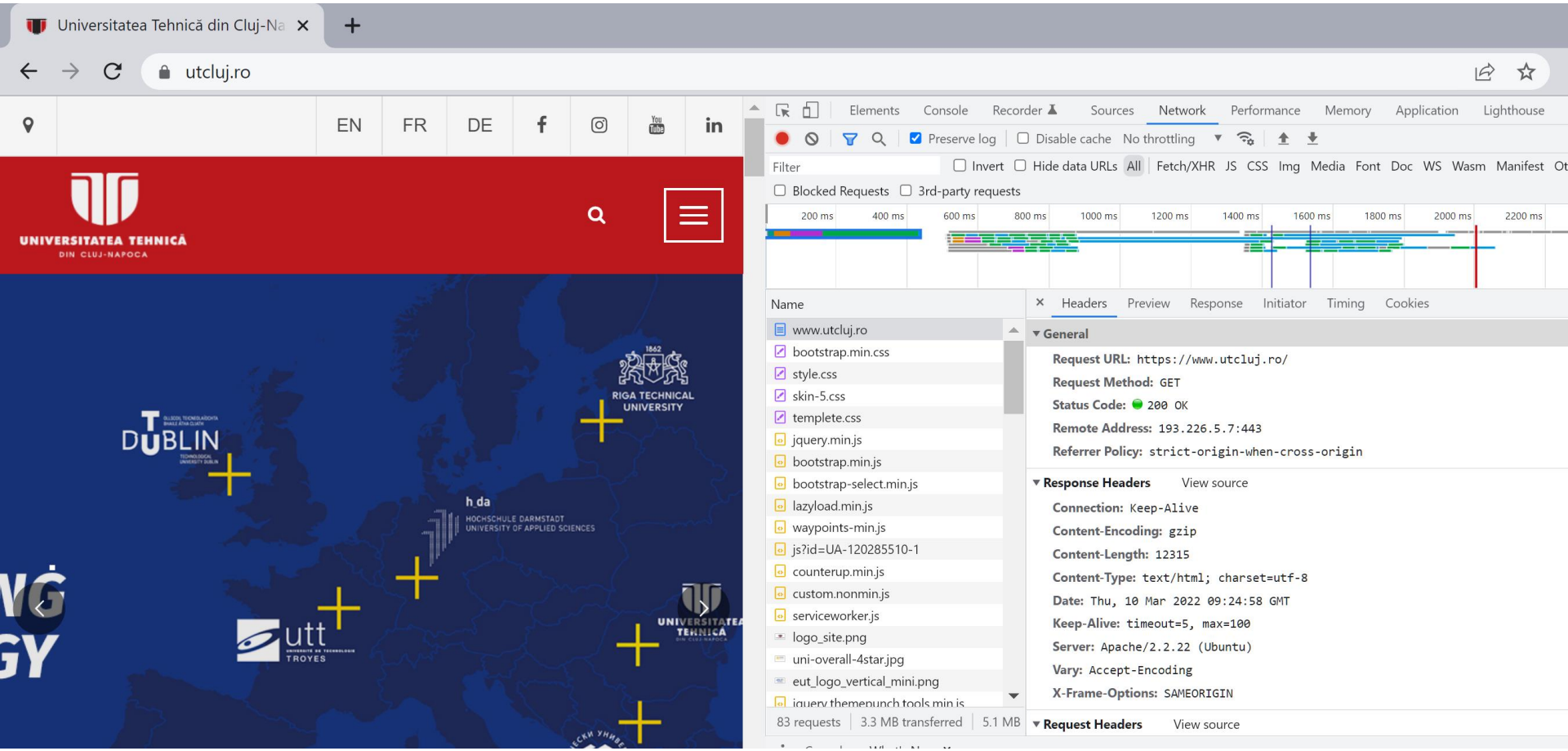
(a) HTTP request



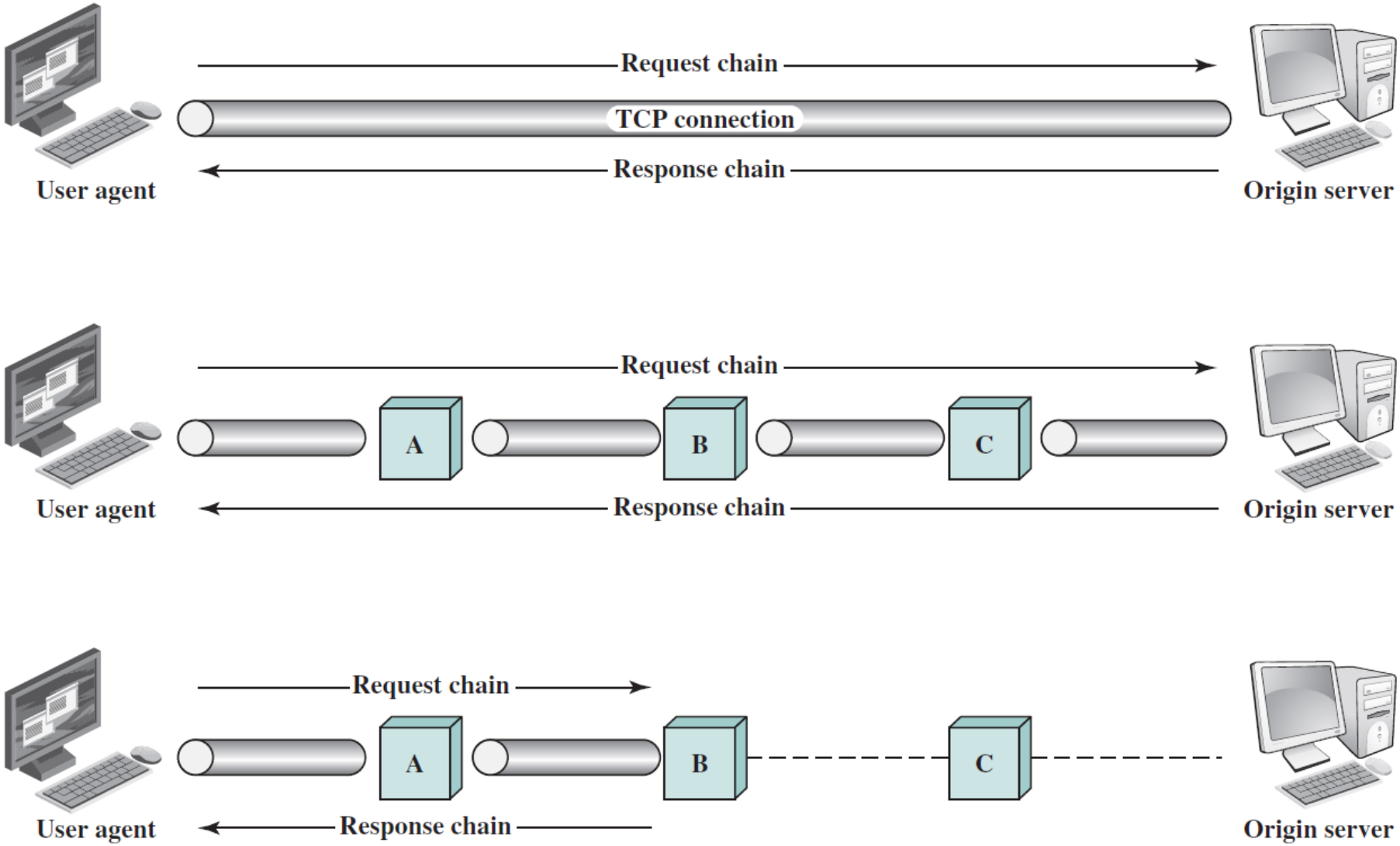
(b) HTTP response

Example of a HTTP Get method on utcluj.ro

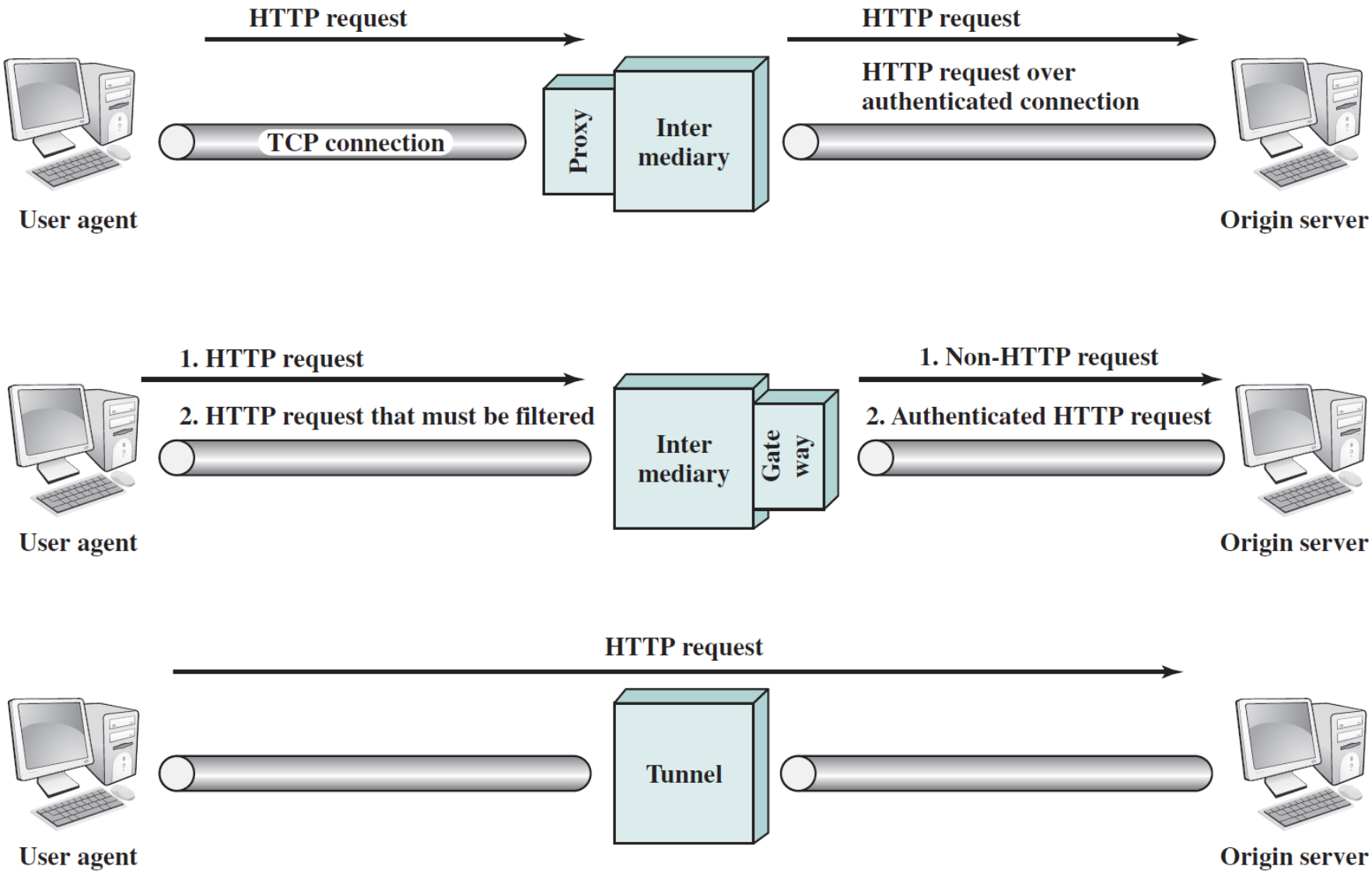
- using Chrome browser and DevTools – F12 on the keyboard



Examples of HTTP Operation



Intermediate HTTP Systems



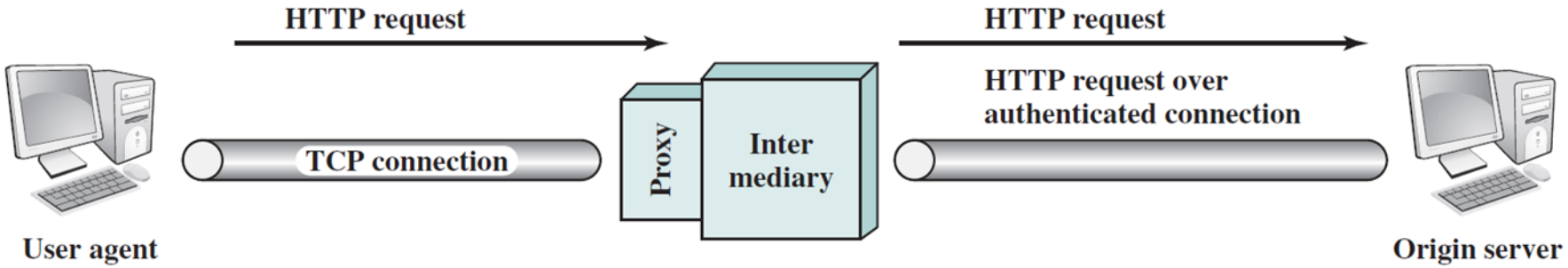
Intermediate Systems

HTTP defines three forms of intermediate systems:

-Proxy, acting on behalf of clients, forwarding their requests;

-Security intermediary: is a server in interacting with client & client vs. server; is on the client side within a firewall; presents clients' requests to server, over an authenticated connection;

-may also implement different versions of HTTP (if the client and server are running different versions of HTTP)



Intermediate Systems

HTTP defines three forms of intermediate systems:

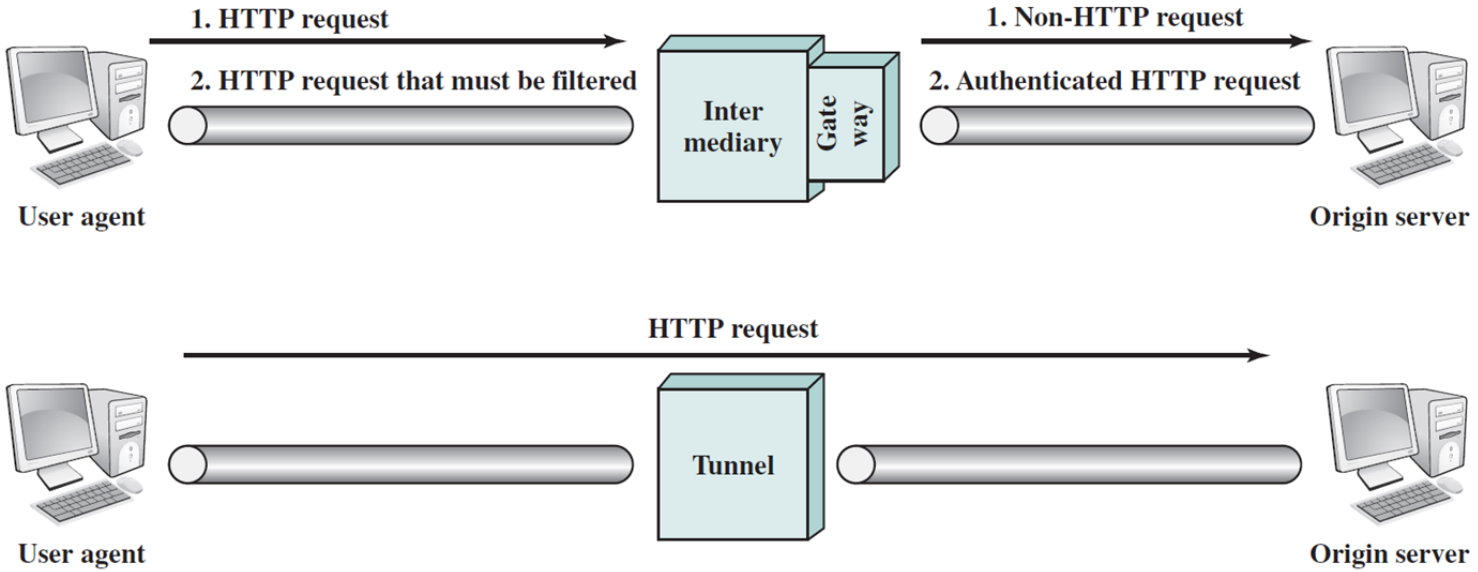
-Gateway, appears as server to client (acts on behalf of servers);

-Security intermediary: is on the server part on a firewall;

-Non-HTTP server: allows for accessing servers running other protocols than HTTP (FTP, Gopher servers)

-Tunnel, relay between TCP connections; no extra info check; example: a firewall in which a client & a server maintains a secure connection for purposes of HTTP transactions

-Cache, is a facility that may store previous requests and responses for handling new requests



Multimedia Networking

Impacts of multimedia networking:

– Users can choose when they want content (such as TV and radio programs) delivered, rather than needing to watch/listen when the content is broadcasted

– Telephone service and data networking will tend to converge

– Two-way video conferencing will often be used instead of telephony

– E-mail will integrate with voice mail (e-mail messages will be spoken rather than Typed)

– Voice/video e-mail: video can be included along with speech

– Multi-way video conferencing already allows virtual meetings and conferences

– As these technologies improve, the need for travel will decrease

Ingredients:

– entertainment video

– IP telephony (or voice-over-IP, VOIP)

– Internet radio

– Multimedia web sites

– teleconferencing, both 2 party and multi-party (e. g. the Access Grid Node).

– interactive games

– virtual worlds (which overlaps with interactive games)

Multimedia Application Specificity

-highly sensitive to end-to-end delay and delay variation (jitter)

-service requirements are quite different from those of traditional data applications

-attempts to extend the Internet architecture to provide explicit support for the service requirements of multimedia applications (reservation techniques, differentiated services)

How should the Internet evolve to support multimedia?

– One view: no fundamental changes are necessary

There should be more use of multicast and caching

More bandwidth should be added to links, and more router capacity should be added

– Another view: applications should be able to reserve end-to-end bandwidth

A protocol to allow applications to reserve bandwidth is needed

Router scheduling policies must take these reservations into account

Some packets will get preferential packets, and probably the senders of those packets will have to pay more.

Applications must be able to describe the traffic that they intend to send

The network must have a means to determine whether it has the bandwidth necessary to support a reservation request

– An intermediate view: a **differentiated services** model

A small number of classes of service

IP Datagrams get different levels of service depending on their service level.

Service characteristics

– Multimedia applications are delay-sensitive and loss-tolerant

– Most data applications are delay-insensitive and loss-intolerant

– TCP/IP was designed for data applications, whereas ATM was designed for both data and multimedia; TCP/IP seems to have mostly prevailed over ATM, so we are faced with adding ATM-like features to TCP/IP (example: reserving network resources between two endpoints requires that something like a virtual circuit be set up and intermediate routers must be aware of the flow between the endpoints)

– Specific Problems

-network-induced jitter. Some solutions: client buffering, packet sequence numbers, packet timestamps.

-lost packets. Some solutions: forward error correction, sending periodic parity packets, sending a redundant lower-quality delayed stream

-interleaving

-media encapsulation: RTP and RTCP

– Real-time audio/video conferencing: H.323, an “umbrella protocol”.

Service models for the Internet

The network-layer protocol of today's Internet provides a "best-effort" service

No promises are made for end-to-end delay nor for variation in end-to-end delay (packet jitter)

Since the Internet service model is not appropriate for multimedia applications, it is challenging to develop such applications, and performance is unsatisfactory when networks are congested

Methods for overcoming these limitations:

– Use UDP rather than TCP, which avoids TCP congestion control such as slow-start.

– Delay playback at the receiver, so that late packets can be used. This is more feasible for non-interactive applications

– Timestamp packets at the sender so that the receiver knows when they should be played

– Prefetch stored data at times when extra bandwidth is available

Examples of Multimedia Applications

Streaming and stored audio and video

Distinguishing features:

– *Stored media:* The multimedia content is stored at the server

-user may pause, rewind, fast-forward, or index through the content.

-acceptable response time is from 1 to 10 seconds

– *Streaming:* A client begins playout of the content a few seconds after it begins receiving the file from the server.

-client will be playing out content from one location while receiving later parts

Products include RealPlayer and Windows Media Player.

– *Continuous playout*

-once playout begins, it should proceed according to the original timing of the recording

-if data is received after it should be played, then the data is useless.

Streaming of Live Audio and Video

-includes radio and television broadcast over the Internet

-fast-forward is not possible

-pausing and rewinding can be implemented with local storage

-currently most often implemented using multiple unicast sessions.

-Access Grid is implemented using multicast.

Real-time Interactive Audio and Video

-includes Internet Phone

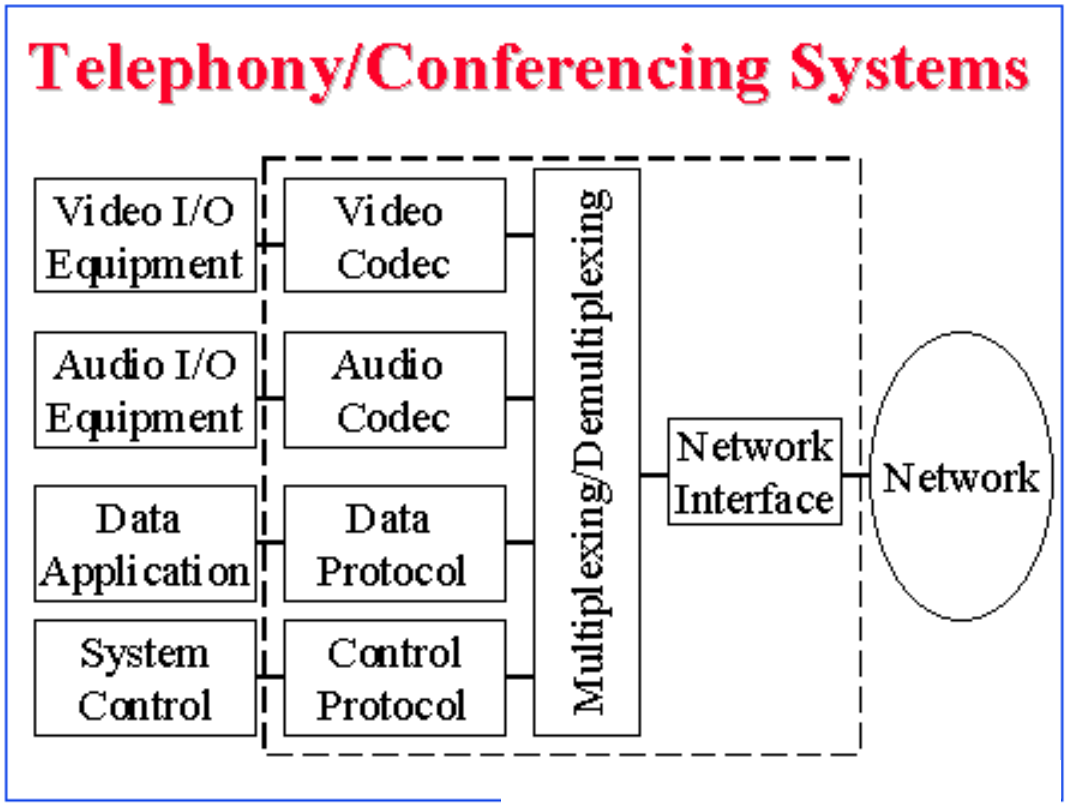
-it sets constraints on delay

-for voice, delays smaller than 150 milliseconds are not perceived by a human listener

-delays of 150 to 400 milliseconds are acceptable

-delays of over 400 milliseconds tend to be unacceptable

General Architecture of Audio/Video Systems



Audio and Video Compression

Before sending audio and video over the network, it must be digitized and compressed

Uncompressed video is extremely large, so compression is almost a necessity

Very high compression ratios are often possible for video

Audio digitization techniques

– Pulse code modulation (PCM): samples the audio signal at some fixed rate, and then represents each sample with some number of bits; standard telephone digitization samples at 8000 times per second, and then encodes each sample with 256 quantization values => an uncompressed signal of 64 Kbps

- not very good fidelity, is not suitable for music

- audio compact disks also use PCM: audio CDs sample at 44,100 samples per second, and use 16 bits per sample; results in a rate of 705.6 Kbps for mono transmissions or 1.411 Mbps for stereo.

Audio compression

– GSM issues (13 Kbps rate)

– G.729 (8 Kbps)

– MP3 (MPEG layer 3 standard): 128 or 112 Kbps - can be used for streaming audio

– Proprietary standards

Video compression

- Use of **MPEG** (Motion Picture Experts Group) standards:

  - MPEG 1 for CD-ROM quality video (1.5 Mbps)

  – MPEG 2 for DVD quality video (3-6 Mbps)

  – MPEG 4 for object-oriented video compression

– H.261, under H.323 umbrella

– Proprietary standards.

Streaming Stored Audio and Video

Clients request compressed audio/video files that are resident on servers

The server sends the file to the client over a socket, either TCP or UDP

A protocol such as **real-time protocol (RTP)** is used to encapsulate segments of the file with special headers

When the file starts to arrive at the client, typically it is played within a few seconds

A protocol such as **real-time streaming protocol (RTSP)** can be used to give interactivity, such as: pause/resume and jumps within the file

A media player application (a web-browser plugin) performs the following functions:

– *Decompression*

– *Jitter removal:* done by buffering the signal

– *Error correction.* Three techniques:

Redundant packets and error-correcting codes

Client requests retransmission of lost packets

Interpolating the lost data from the received data

– provides a *GUI (*Graphic User Interface) with control knobs.

Accessing Audio and Video from a Web Server

Normally, the media will be stored as files (compressed) on the server

In the case of video, there may be separate audio and video files, or they might be interleaved in the same file

It is the responsibility of the media player to manage synchronization of the two streams

For simplicity, we assume that there is only one file

A possible architecture:

– The browser connects to the web server

– The browser requests the audio/video file with an HTTP request message

– The web server sends the file to the browser in an HTTP response message

– The browser looks at the content type, launches the media player if necessary, and forwards the file to the media player

– The media player renders the audio/video file

One drawback: the browser is an intermediary between the server and the media player => the file would need to be completely downloaded before it gets forwarded to the media player

It makes it more difficult for the media player to allow the user to control the stream.

A better method is to make use of a **meta file**, which provides information (e. g., the location and encoding) about the audio/video file => the browser passes the meta file to the media player, and the media player makes its own connection to the server

This server may be a **streaming server** rather than a web server (may be specifically designed to deliver streaming media, and may run a different protocol than HTTP)

Options for delivering the audio/video from the streaming server to the client:

1. The server sends the stream over UDP at the encoded rate of the audio/video (*drain rat*e.) The client decompresses and plays the stream as it receives it.

2. Same, but the media player delays playout for 2-5 seconds in order to eliminate network-induced jitter (the client buffers the incoming stream)

3. The stream is sent over TCP, and the media player buffers the stream for 2-5 sec. It gives TCP the chance to retransmit lost packets, but the flow & congestion control of TCP might interfere with sending the stream at the drain rate.

RTSP (Real-Time Streaming Protocol)

RTSP is an **out-of-band protocol** that allows a media player to control the transmission of a media stream

– Out-of-band protocol means that RTSP messages are sent over a separate channel from the media itself

– RTSP does not specify how the media is compressed, encapsulated in packets, transported, or buffered

RTP (Real Time Protocol)

RTP is a protocol that provides for a standard packet structure that includes fields for sequence numbers, timestamps, etc. RTP typically runs on top of UDP.

Each source (e. g., camera or microphone) can be assigned its own independent stream of RTP packets

RTP can be sent over unicast of multicast.

Multiple RTP streams belonging to a single application can make up an **RTP sessio**n.

The RTP header fields:

– The *payload type* gives the compression method

– The *sequence number field* is 16 bits long, and can be used to detect packet loss

– The *timestamp field* is 32 bits long, and it reflects the sampling instant of the first byte in the RTP packet

– The *synchronization source identifier (SSRC)* is 32 bits long, and identifies the source of the RTP stream

Typically, each stream in the RTP session has a distinct SSRC. The SSRC is chosen randomly.

RTCP (Real Time Control Protocol)

Typically used by participants using RTP in a multicast scenario

RTCP packets are sent periodically and contain sender and/or receiver reports and statistics than can be useful to applications

H.323

H.323 is a standard for real-time audio and video conferencing among end systems on the Internet; also covers how end systems on the Internet communicate with ordinary circuit-switched telephone networks. It means:

The products of Internet telephony and videoconferencing that use H.323 should

be able to interoperate and should be able to communicate with ordinary telephones

H.323 is an umbrella specification that includes:

-A specification for how endpoints negotiate audio/video encodings.

-A specification for how audio and video chunks are encapsulated and sent over the network; this is done by means of RTP

-A specification for how endpoints communicate with their respective gatekeepers

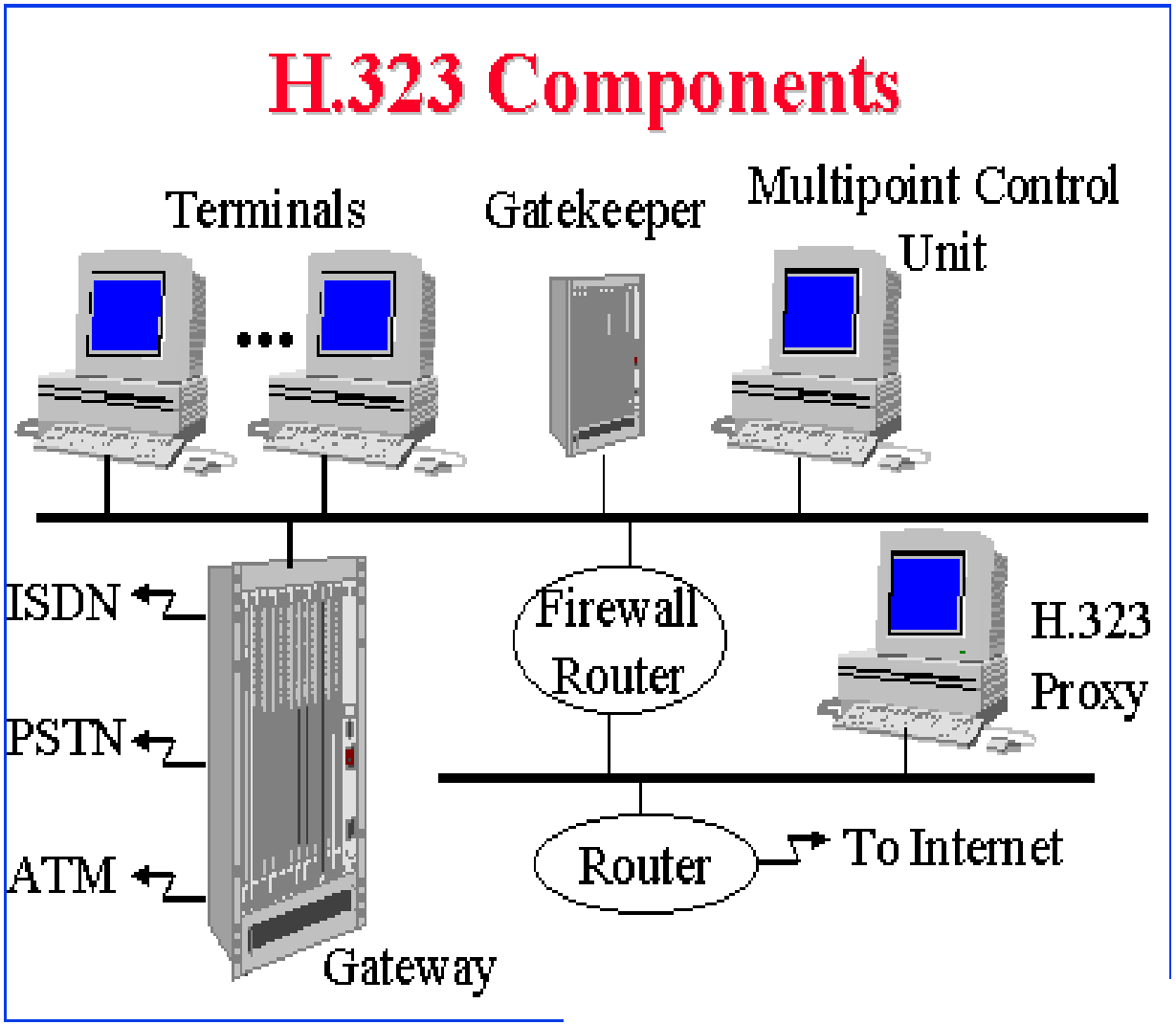
- A specification for how Internet phones communicate with ordinary phones

H.323 components

End points can be standalone devices or computer applications

Gateways permit communication among H.323 endpoints and ordinary telephones

Gatekeepers (optional) provide address translation authorization, bandwidth management, and accounting for LAN terminals



H.323 endpoints must support the following protocols:

G.711 PCM (pulse code modulation) speech encoding

Real-Time Protocol

H.245: an “out-of-band” protocol for controlling media between H.323 endpoints

Q.931: a signaling protocol for establishing and terminating calls. Provides for interoperability with traditional telephones.

RAS: a protocol that allows endpoints to communicate with a gatekeeper

(see next figure)

H.323 channels

An endpoint maintains several channels

The H.245 control channel is a TCP connection that is used for capability exchanged and opening and closing media channels

H.245 is similar in purpose to RTSP (which is used for stored media and streaming sessions)

Q.931 provides traditional telephone functionality.

H.323 Protocols

- Multimedia over LANs
- Provides component descriptions, signaling procedures, call control, system control, audio/video codecs, data protocols

<table>
<tr><td>Video</td><td>Audio</td><td colspan="4">Control and Management</td><td>Data</td></tr>
<tr><td>H.261
H.263</td><td>G.711, G.722,
G.723.1, G.728,
G.729</td><td rowspan="2">RTCP</td><td>H.225.0
RAS</td><td>H.225.0
Signaling</td><td>H.245
Control</td><td>T.124</td></tr>
<tr><td colspan="2">RTP</td><td colspan="3">X.224 Class 0</td><td>T.125</td></tr>
<tr><td colspan="3">UDP</td><td colspan="3">TCP</td><td rowspan="2">T.123</td></tr>
<tr><td colspan="6">Network (IP)</td></tr>
<tr><td colspan="7">Datalink (IEEE 802.3)</td></tr>
</table>

Session Initiation Protocol (SIP)

Application level signaling protocol

Allows creating, modifying & terminating sessions with more participants

Supports user location, call setup, call transfers, mobility (by proxying & redirection)

Used by the gateways to setup connections

SIP works in conjunction with IP protocols:

- RSVP for reserving resources
- RTP/RTCP/RTSP for transporting real-time data
- SAP (Session Announcement Protocol) – advertise multimedia session
- SDP (Session Description Protocol)

Can use TCP/UDP

Text based