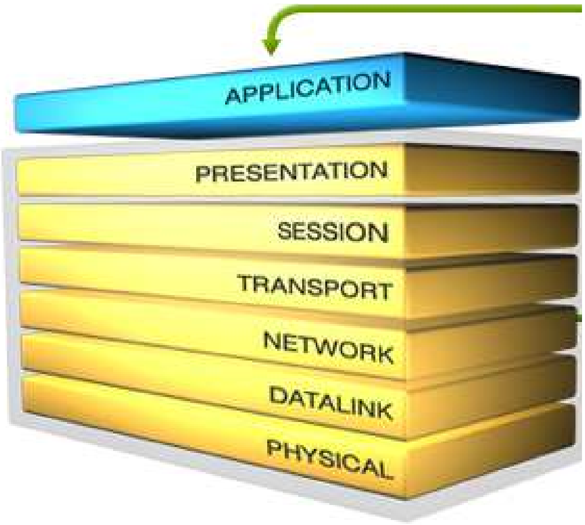
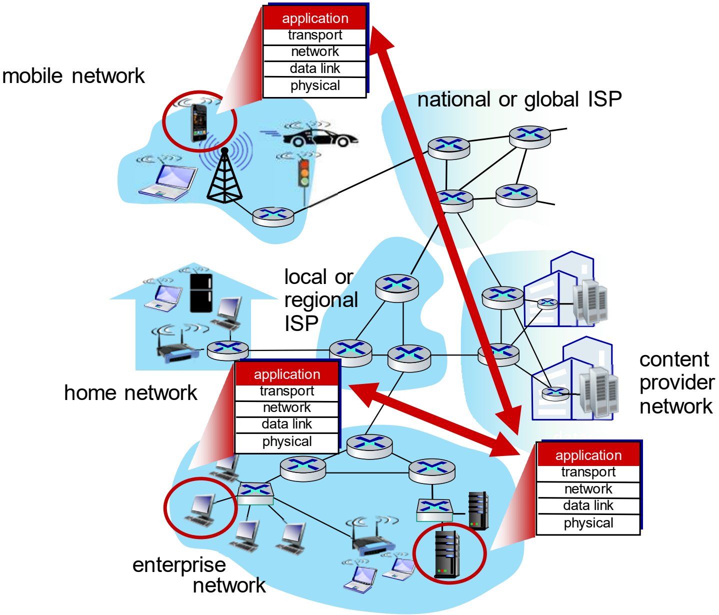
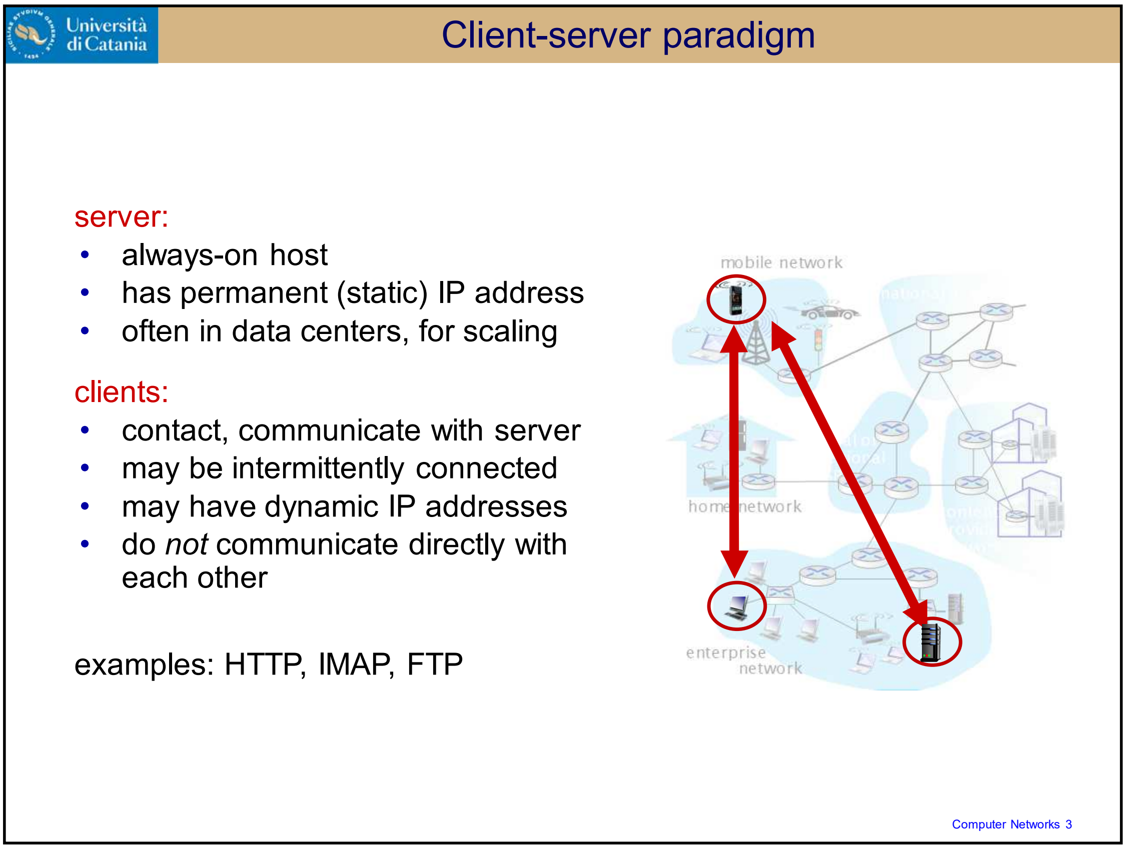
Application level



Interprocess communications



Applications require a bi-directional, reliable channel to exchange messages.



Client-server paradigm

server:

- always-on host
- has permanent (static) IP address
- often in data centers, for scaling

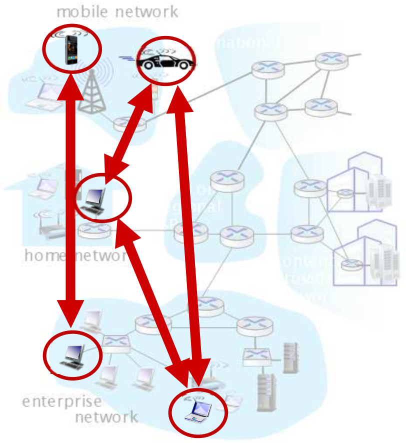
clients:

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

examples: HTTP, IMAP, FTP

Peer-peer architecture

- *no* always-on server
- arbitrary end systems directly communicate
- peers request service from other peers, provide service in return to other peers
  - *self scalability* – new peers bring new service capacity, as well as new service demands
- peers are intermittently connected and change IP addresses
  - complex management



example: P2P file sharing

Università di Catania

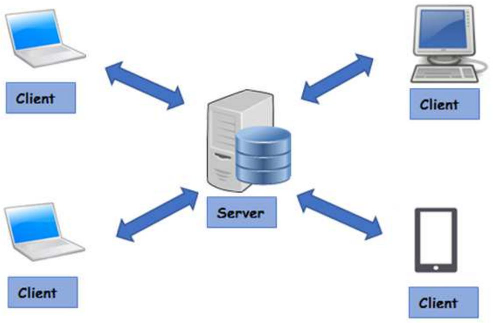
Client-server paradigm

process:

a 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

Message exchange allows the synchronization between processes



clients, servers

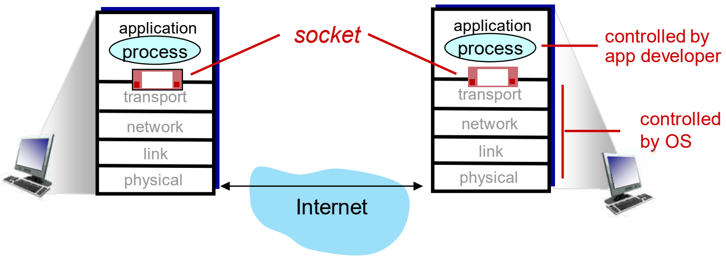
client process: process that initiates communication

server process: process that waits to be contacted

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 to deliver message to socket at receiving process
  - two sockets involved: one on each side



Addressing processes

To receive messages, process must have an *identifier.*

Every host device has unique 32-bit (128-bit) IP address.

Q: does IP address of host on which process runs suffice for identifying the process?

A: no, *many* processes can be running on same host!

Identifier includes both IP address and port numbers associated with process on host.

- example port numbers:
  HTTP server: 80
  mail server: 25
- to send HTTP message to gaia.cs.umass.edu web server:
  IP address: 128.119.245.12
  port number: 80

An application-layer protocol defines:

- types of messages exchanged,
  - e.g., request, response
- message syntax:
  - what fields in messages & how fields are delineated
- message semantics
  - meaning of information in fields
- rules for when and how processes send & respond to messages

open protocols:

- defined in RFCs, everyone has access to protocol definition
- allows for interoperability

e.g., HTTP, SMTP

proprietary protocols:

- e.g., Skype

What transport service does an app need?

data integrity

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

throughput

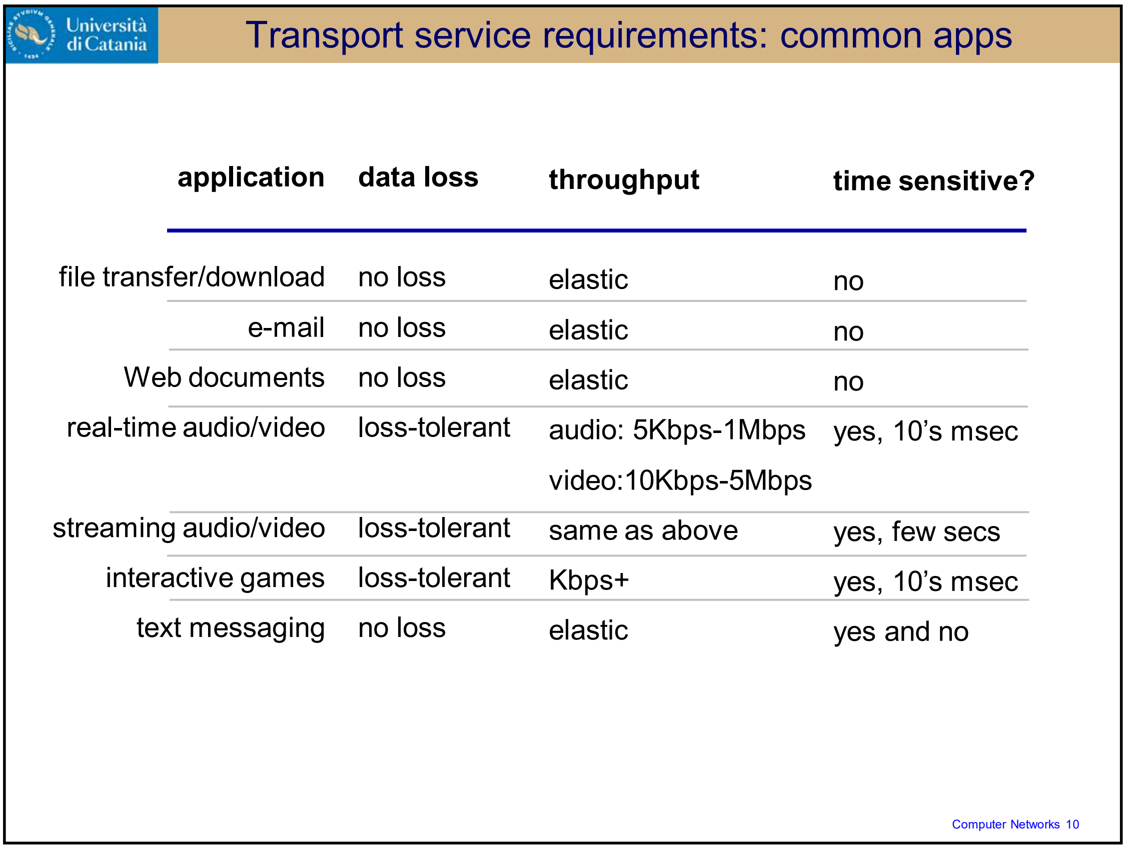
- some apps (e.g., multimedia) require minimum amount of throughput to be “effective”
- other apps (“elastic apps”) make use of whatever throughput they get

timing

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

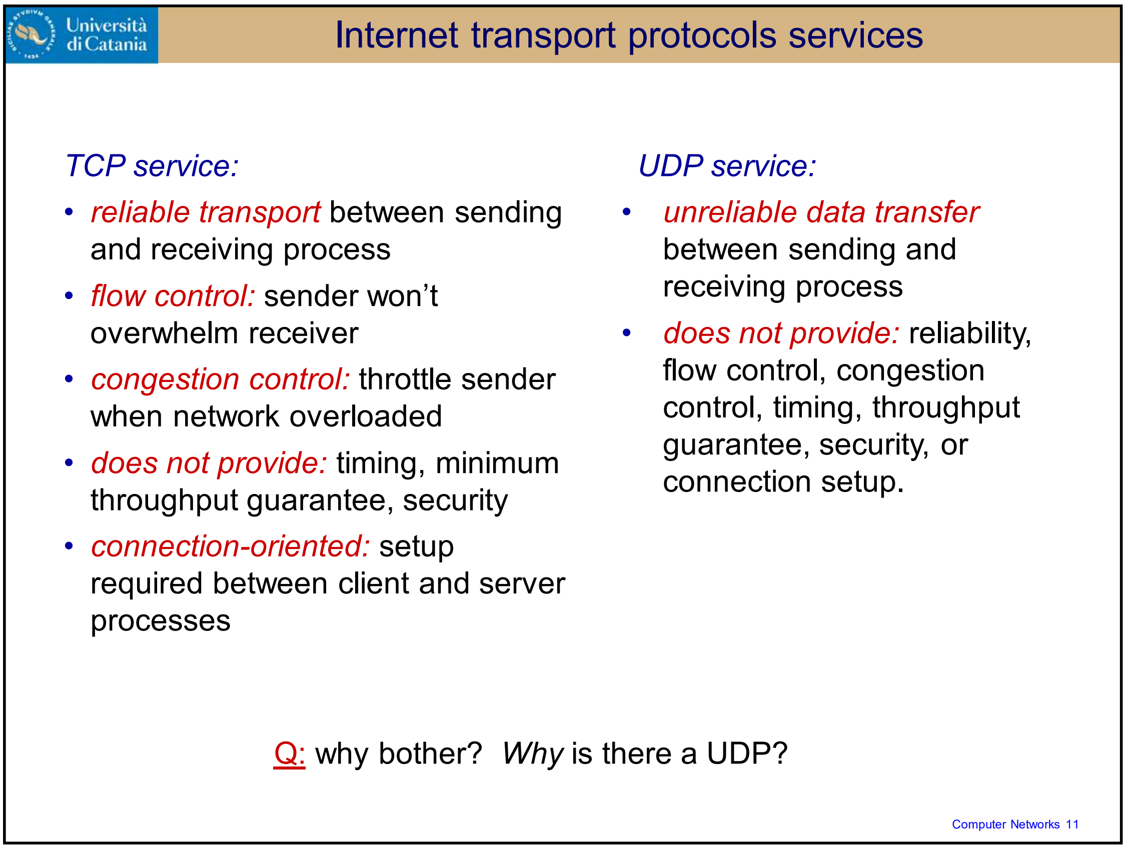
security

- encryption, data integrity, …



Transport service requirements: common apps

application	data loss	throughput	time sensitive?
file transfer/download	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, 10's msec
streaming audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	Kbps+	yes, 10's msec
text messaging	no loss	elastic	yes and no



Internet transport protocols services

TCP service:

- *reliable transport* between sending and receiving process
- *flow control:* sender won’t overwhelm receiver
- *congestion control:* throttle sender when network overloaded
- *does not provide:* timing, minimum throughput guarantee, security
- *connection-oriented:* setup required between client and server processes

UDP service:

- *unreliable data transfer* between sending and receiving process
- *does not provide:* reliability, flow control, congestion control, timing, throughput guarantee, security, or connection setup.

Q: why bother? *Why* is there a UDP?

Internet transport protocols services

application	application layer protocol	transport protocol
file transfer/download	FTP [RFC 959]	TCP
e-mail	SMTP [RFC 5321]	TCP
Web documents	HTTP 1.1 [RFC 7320]	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC	TCP or UDP
	3550], or proprietary HTTP	
streaming audio/video	[RFC 7320], DASH	TCP
interactive games	WOW, FPS (proprietary)	UDP or TCP

Port numbers

Internet ports can be divided in 3 groups:

Well Known Ports: (0 – 1023) - for system services

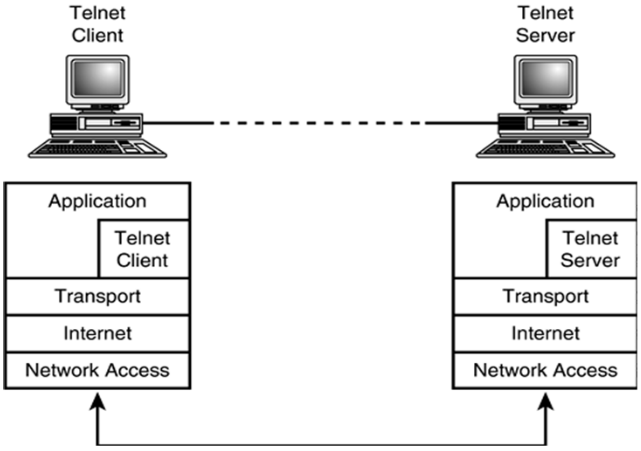
Registered Ports: (1024 – 49151) These numbers are assigned by Internet Corporation for Assigned Names and Numbers for some specific use

Dynamic and/or Private Ports: (49152 – 65535)

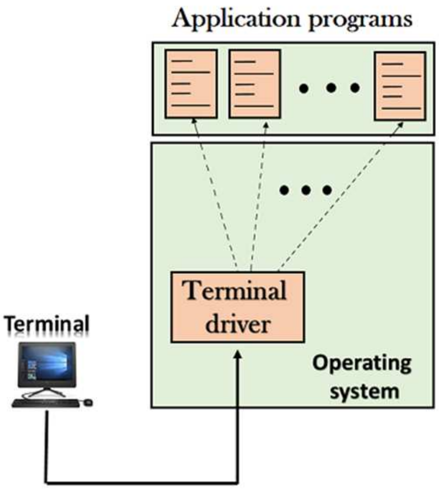
Ports

20/tcp	FTP - data
21/tcp	FTP - control
22/tcp	SSH - Secure login
23/tcp	Telnet
25/tcp	SMTP
53/tcp	DNS
53/udp	DNS
67/udp	BOOTP (Server) and DHCP (Server)
68/udp	BOOTP (Client) and DHCP (Client)
69/udp	TFTP
70/tcp	Gopher
80/tcp	HTTP
88/tcp	Kerberos Authenticating agent
110/tcp	POP3
123/udp	NTP
143/tcp	IMAP4
161/udp	SNMP (Agent)
162/udp	SNMP (Manager)
443/tcp	HTTPS
465/tcp	SMTP over SSL
993/tcp	IMAP4 over SSL
995/tcp	POP3 over SSL

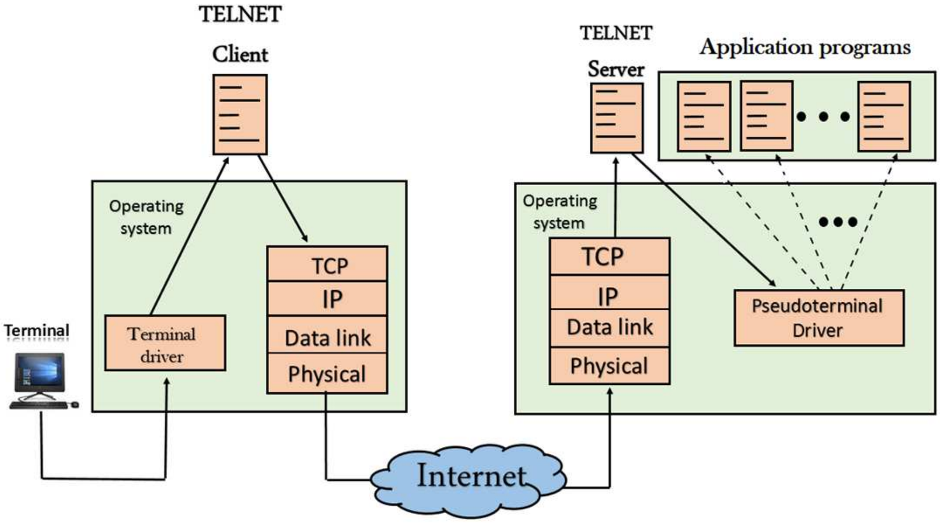
Telnet


Telnet Client
Telnet Server
Application
Telnet Client
Transport
Internet
Network Access
Application
Telnet Server
Transport
Internet
Network Access

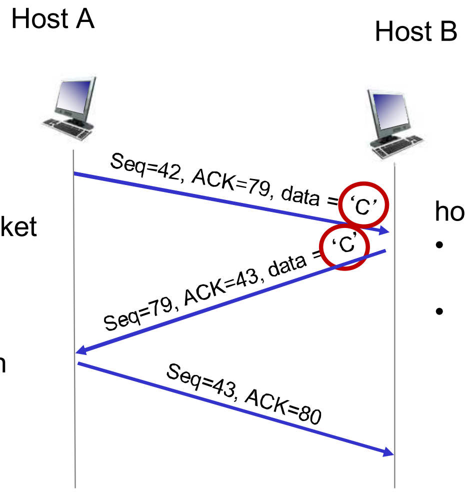
OS Terminal


Application programs
• • •
• • •
Terminal driver
Operating system
Terminal

Telnet


TELNET
Client
Operating system
TCP
IP
Data link
Physical
Terminal
Terminal driver
Internet
TELNET
Server
Application programs
Operating system
TCP
IP
Data link
Physical
Pseudoterminal Driver

Telnet



Host A

Host B

User
- types ‘C’

host A
- sends a packet

Seq=42, ACK=79, data = ‘C’

host B
- inserts ‘C’ in the standard input,
- echoes back ‘C’

Seq=79, ACK=43, data = ‘C’

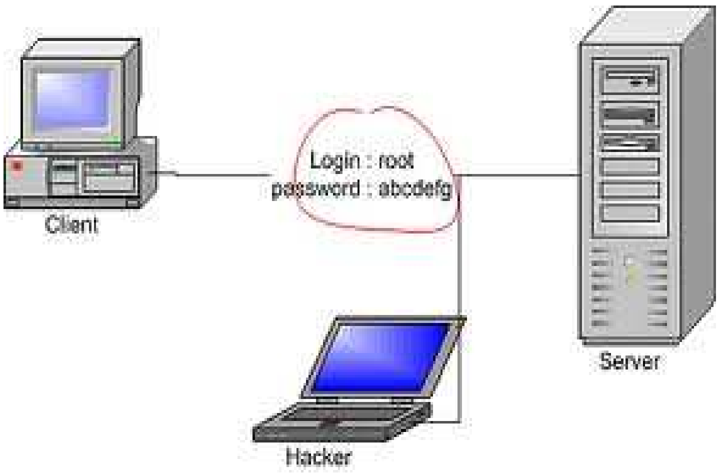
host A
- shows ‘C’ on the terminal

Seq=43, ACK=80

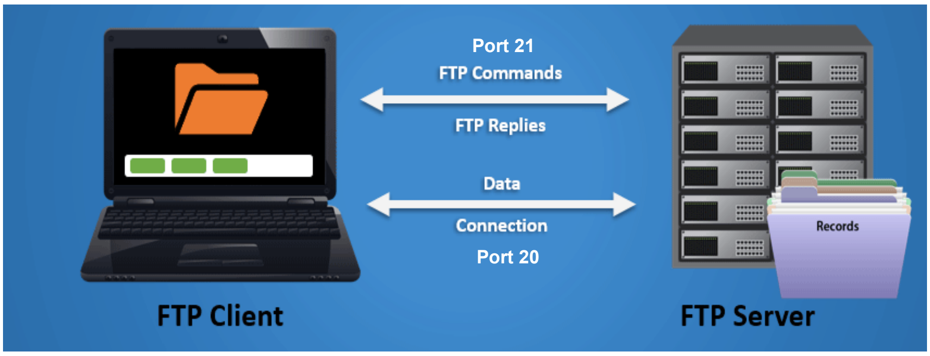
simple telnet scenario

Università di Catania

Telnet

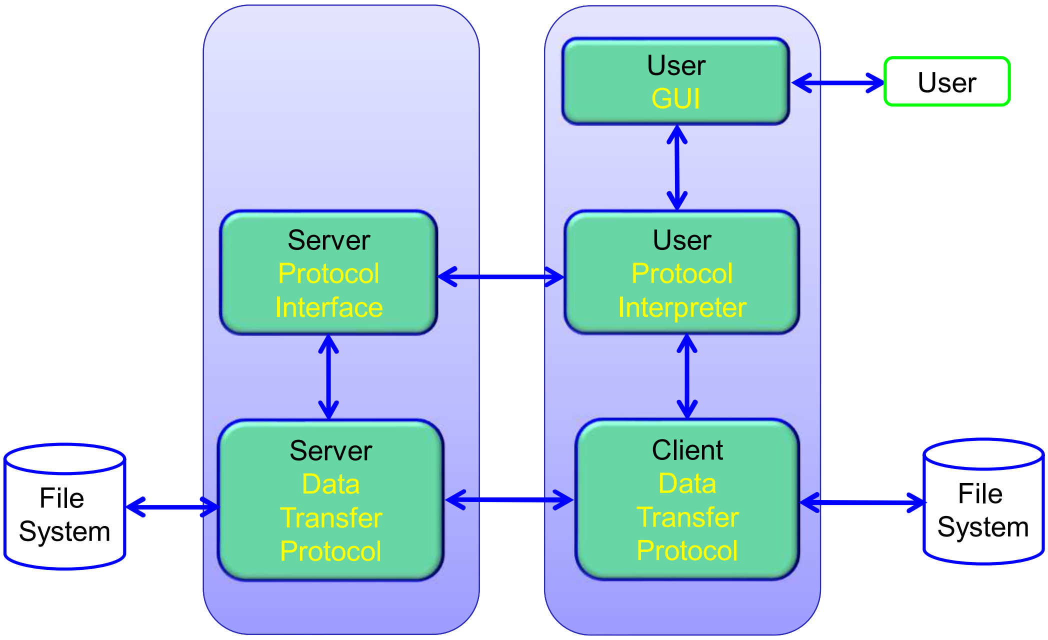


FTP


Port 21
FTP Commands
FTP Replies
Data
Connection
Port 20
Records
FTP Client
FTP Server

Università di Catania

FTP



User

User GUI

User Protocol Interpreter

Server Protocol Interface

Server Data Transfer Protocol

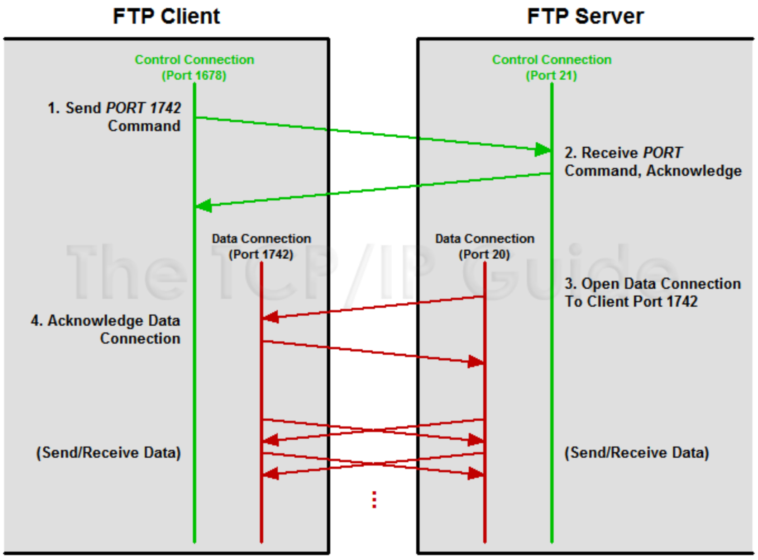
Client Data Transfer Protocol

File System

File System

Università di Catania

FTP



FTP Client

FTP Server

Control Connection (Port 1678)

Control Connection (Port 21)

1. Send *PORT 1742* Command

2. Receive *PORT* Command, Acknowledge

Data Connection (Port 1742)

Data Connection (Port 20)

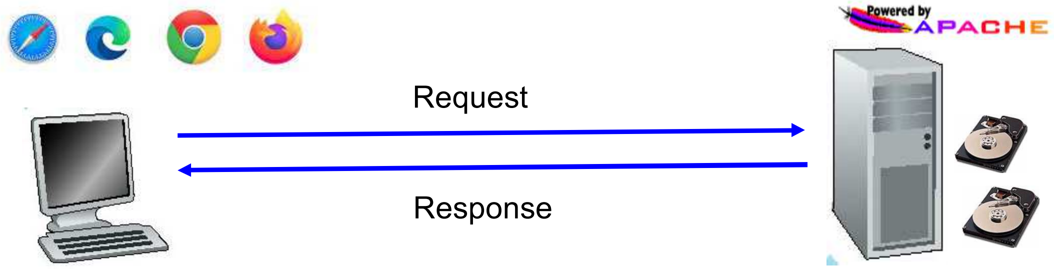
3. Open Data Connection To Client Port 1742

4. Acknowledge Data Connection

(Send/Receive Data)

(Send/Receive Data)

HTTP



HTTP

First, a quick review…

- a web page consists of objects, each of which can be stored on different Web servers
- an 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 (Uniform Resource Locator) , e.g.,

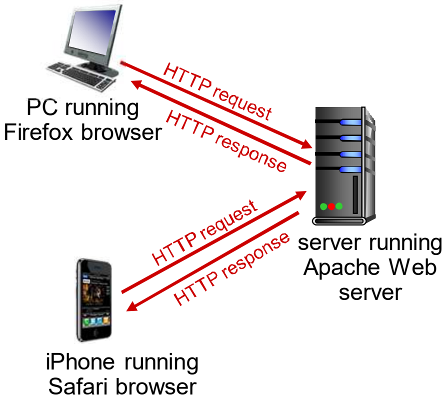
$$\underbrace{\texttt{www.someschool.edu}}_{\text{host name}}\underbrace{\texttt{/someDept/pic.gif}}_{\text{path name}}$$

Università di Catania

HTTP overview

HTTP: hypertext transfer protocol

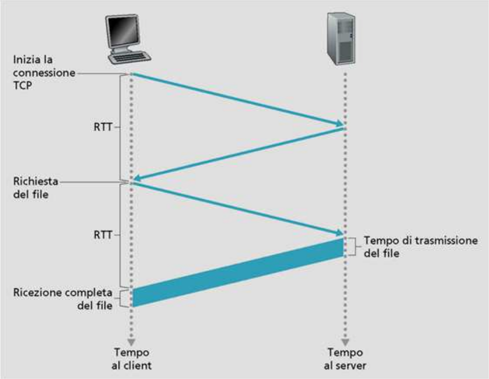
- Web's application layer protocol
- client/server model:
  - *client:* browser that requests, receives, (using HTTP protocol) and "displays" Web objects
  - *server:* Web server sends (using HTTP protocol) objects in response to requests



HTTP overview (continued)

HTTP uses TCP:

- client initiates TCP connection (creates socket) to server on 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


Inizia la connessione TCP
RTT
Richiesta del file
RTT
Tempo di trasmissione del file
Ricezione completa del file
Tempo al client
Tempo al server

Stateless vs Stateful

Both print notes (tickets)

HTTP overview (continued)

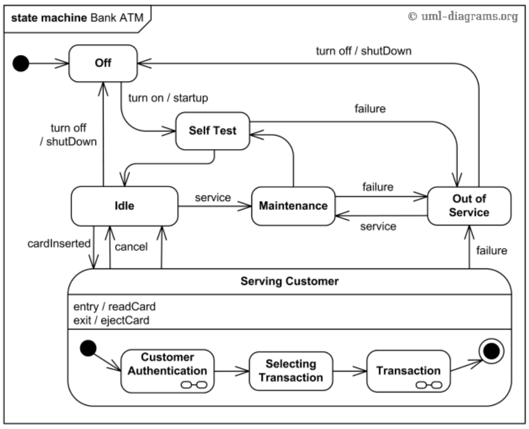
HTTP is “stateless”

- 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

If the HTML page contains reference to other objects, they can be required in a second time using HTTP.

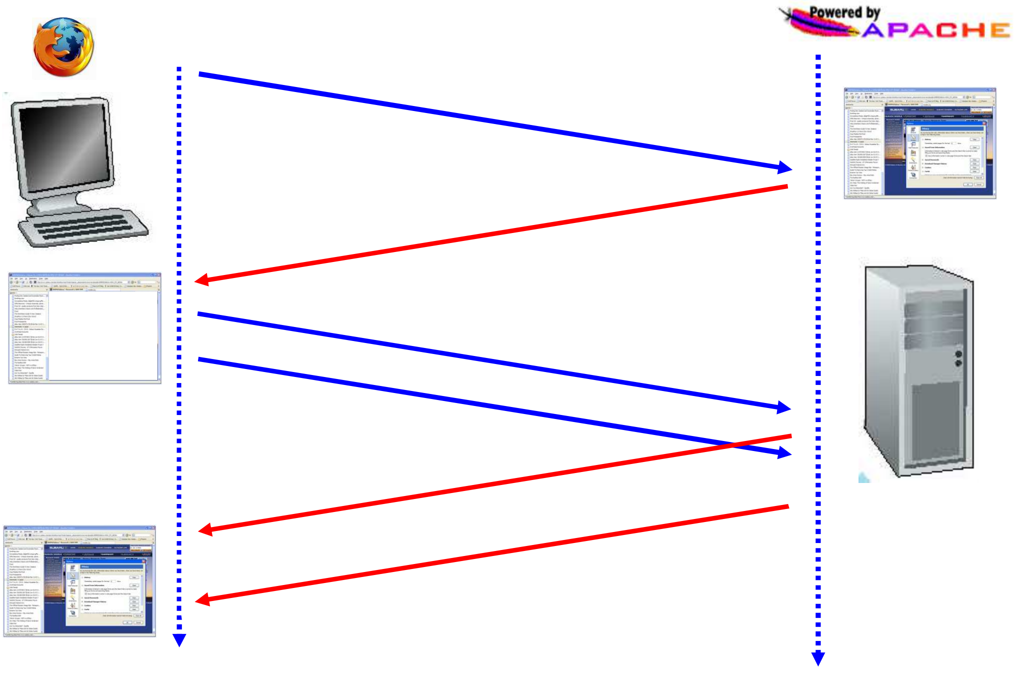
HTTP/1.0 closes every TPC connection after the correspondent object transefer.

HTTP/1.1 uses persistent TCP connections. Different objects can be sent in pipeline using the same TCP connection, that will be closed only at the end.

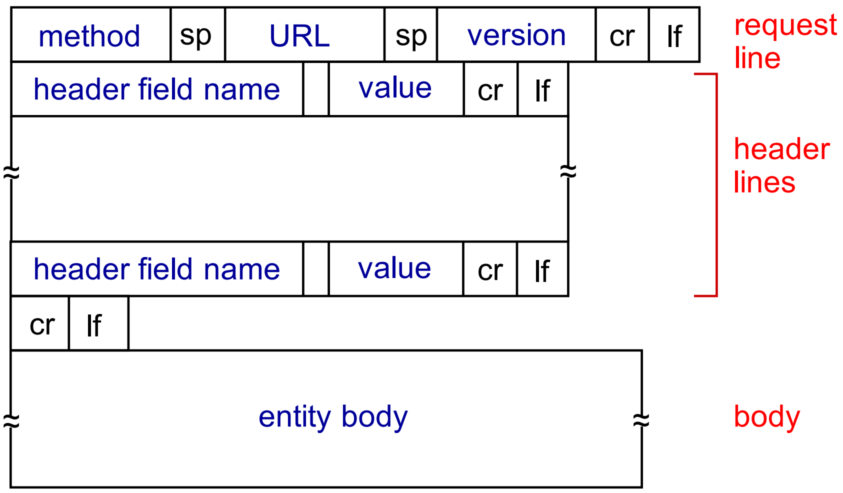
HTTP/2 uses header compression, permits "push" dispatch and parallel download (RFC7540)

HTTP/3 ???

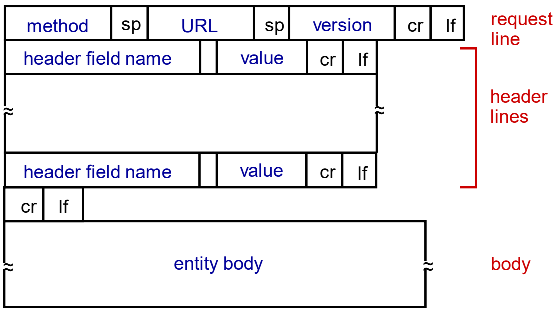
HTTP



HTTP Request



HTTP Request



method sp URL sp version cr lf

request line

header field name value cr lf

header field name value cr lf

header lines

cr lf

entity body

body

carriage return character

line-feed character

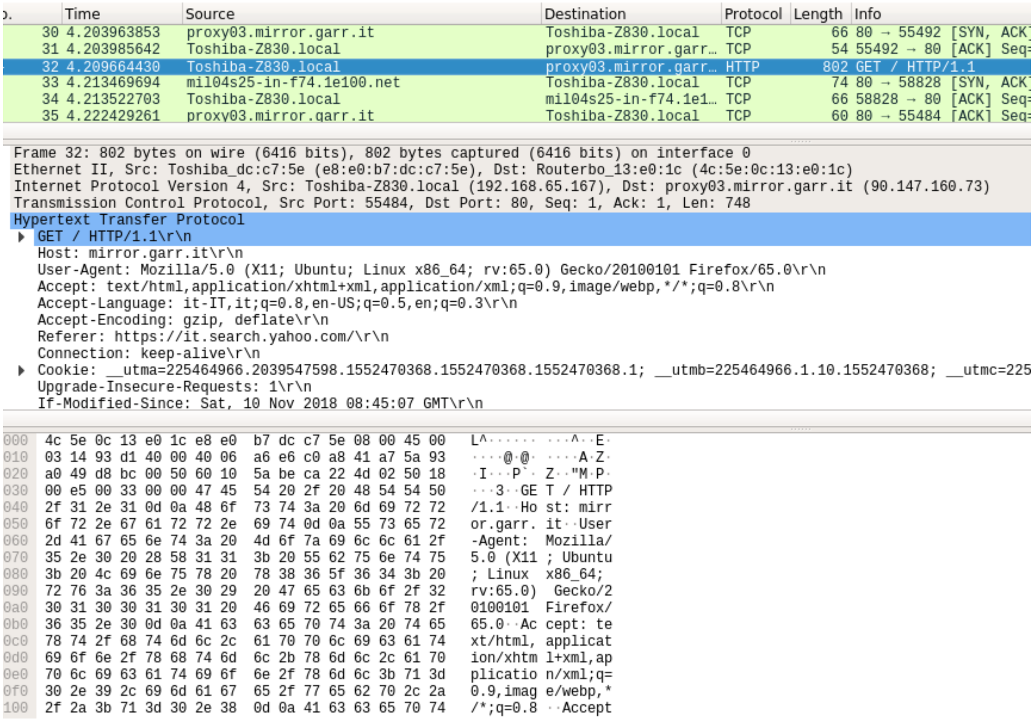
request line (GET, POST, HEAD commands)

header lines

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

```
GET /index.html HTTP/1.1\r\n
Host: www-net.cs.umass.edu\r\n
User-Agent: Firefox/3.6.10\r\n
Accept: text/html,application/xhtml+xml\r\n
Accept-Language: en-us,en;q=0.5\r\n
Accept-Encoding: gzip,deflate\r\n
Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
Keep-Alive: 115\r\n
Connection: keep-alive\r\n
\r\n
```

HTTP Request



```
o.    Time           Source                       Destination              Protocol  Length  Info
   30 4.203963853    proxy03.mirror.garr.it       Toshiba-Z830.local       TCP         66 80 → 55492 [SYN, ACK]
   31 4.203985642    Toshiba-Z830.local           proxy03.mirror.garr…     TCP         54 55492 → 80 [ACK] Seq=
   32 4.209664430    Toshiba-Z830.local           proxy03.mirror.garr…     HTTP       802 GET / HTTP/1.1
   33 4.213469694    mil04s25-in-f74.1e100.net    Toshiba-Z830.local       TCP         74 80 → 58828 [SYN, ACK]
   34 4.213522703    Toshiba-Z830.local           mil04s25-in-f74.1e1…     TCP         66 58828 → 80 [ACK] Seq=
   35 4.222429261    proxy03.mirror.garr.it       Toshiba-Z830.local       TCP         60 80 → 55484 [ACK] Seq=
```

```
Frame 32: 802 bytes on wire (6416 bits), 802 bytes captured (6416 bits) on interface 0
Ethernet II, Src: Toshiba_dc:c7:5e (e8:e0:b7:dc:c7:5e), Dst: Routerbo_13:e0:1c (4c:5e:0c:13:e0:1c)
Internet Protocol Version 4, Src: Toshiba-Z830.local (192.168.65.167), Dst: proxy03.mirror.garr.it (90.147.160.73)
Transmission Control Protocol, Src Port: 55484, Dst Port: 80, Seq: 1, Ack: 1, Len: 748
Hypertext Transfer Protocol
 ▸ GET / HTTP/1.1\r\n
   Host: mirror.garr.it\r\n
   User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:65.0) Gecko/20100101 Firefox/65.0\r\n
   Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8\r\n
   Accept-Language: it-IT,it;q=0.8,en-US;q=0.5,en;q=0.3\r\n
   Accept-Encoding: gzip, deflate\r\n
   Referer: https://it.search.yahoo.com/\r\n
   Connection: keep-alive\r\n
 ▸ Cookie: __utma=225464966.2039547598.1552470368.1552470368.1552470368.1; __utmb=225464966.1.10.1552470368; __utmc=225
   Upgrade-Insecure-Requests: 1\r\n
   If-Modified-Since: Sat, 10 Nov 2018 08:45:07 GMT\r\n
```

```
000  4c 5e 0c 13 e0 1c e8 e0  b7 dc c7 5e 08 00 45 00   L^······ ···^··E·
010  03 14 93 d1 40 00 40 06  a6 e6 c0 a8 41 a7 5a 93   ····@·@· ····A·Z·
020  a0 49 d8 bc 00 50 60 10  5a be ca 22 4d 02 50 18   ·I···P`· Z··"M·P·
030  00 e5 00 33 00 00 47 45  54 20 2f 20 48 54 54 50   ···3··GE T / HTTP
040  2f 31 2e 31 0d 0a 48 6f  73 74 3a 20 6d 69 72 72   /1.1··Ho st: mirr
050  6f 72 2e 67 61 72 72 2e  69 74 0d 0a 55 73 65 72   or.garr. it··User
060  2d 41 67 65 6e 74 3a 20  4d 6f 7a 69 6c 6c 61 2f   -Agent:  Mozilla/
070  35 2e 30 20 28 58 31 31  3b 20 55 62 75 6e 74 75   5.0 (X11 ; Ubuntu
080  3b 20 4c 69 6e 75 78 20  78 38 36 5f 36 34 3b 20   ; Linux  x86_64;
090  72 76 3a 36 35 2e 30 29  20 47 65 63 6b 6f 2f 32   rv:65.0)  Gecko/2
0a0  30 31 30 30 31 30 31 20  46 69 72 65 66 6f 78 2f   0100101  Firefox/
0b0  36 35 2e 30 0d 0a 41 63  63 65 70 74 3a 20 74 65   65.0··Ac cept: te
0c0  78 74 2f 68 74 6d 6c 2c  61 70 70 6c 69 63 61 74   xt/html, applicat
0d0  69 6f 6e 2f 78 68 74 6d  6c 2b 78 6d 6c 2c 61 70   ion/xhtm l+xml,ap
0e0  70 6c 69 63 61 74 69 6f  6e 2f 78 6d 6c 3b 71 3d   plicatio n/xml;q=
0f0  30 2e 39 2c 69 6d 61 67  65 2f 77 65 62 70 2c 2a   0.9,imag e/webp,*
100  2f 2a 3b 71 3d 30 2e 38  0d 0a 41 63 63 65 70 74   /*;q=0.8 ··Accept
```

Other HTTP request messages

GET method (for sending data to server):

- include user data in URL field of HTTP GET request message (following a ‘**?**’):

POST method:

web page often includes form input

- user input sent from client to server in entity body of HTTP POST request message

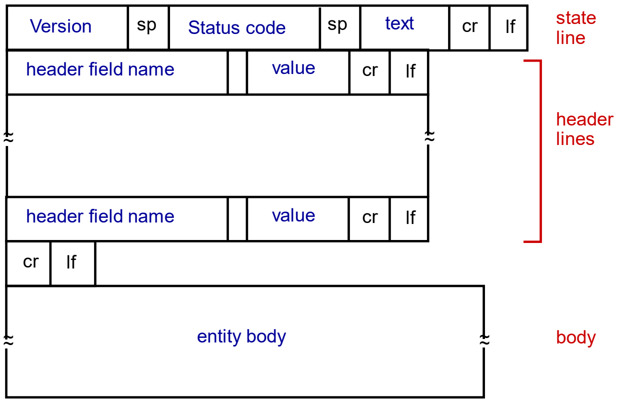
HEAD method:

- requests headers (only) that would be returned *if* specified URL were requested with an HTTP GET method.

PUT method:

- uploads new file (object) to server
- completely replaces file that exists at specified URL with content in entity body of POST HTTP request message

HTTP Response



HTTP Response

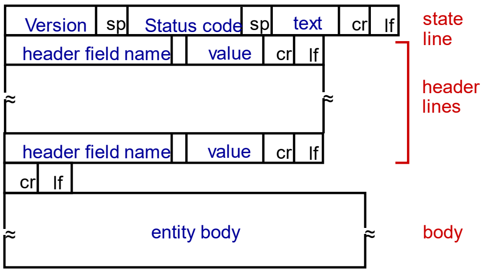
status line (protocol status code status phrase)

header lines

data, e.g., requested HTML file

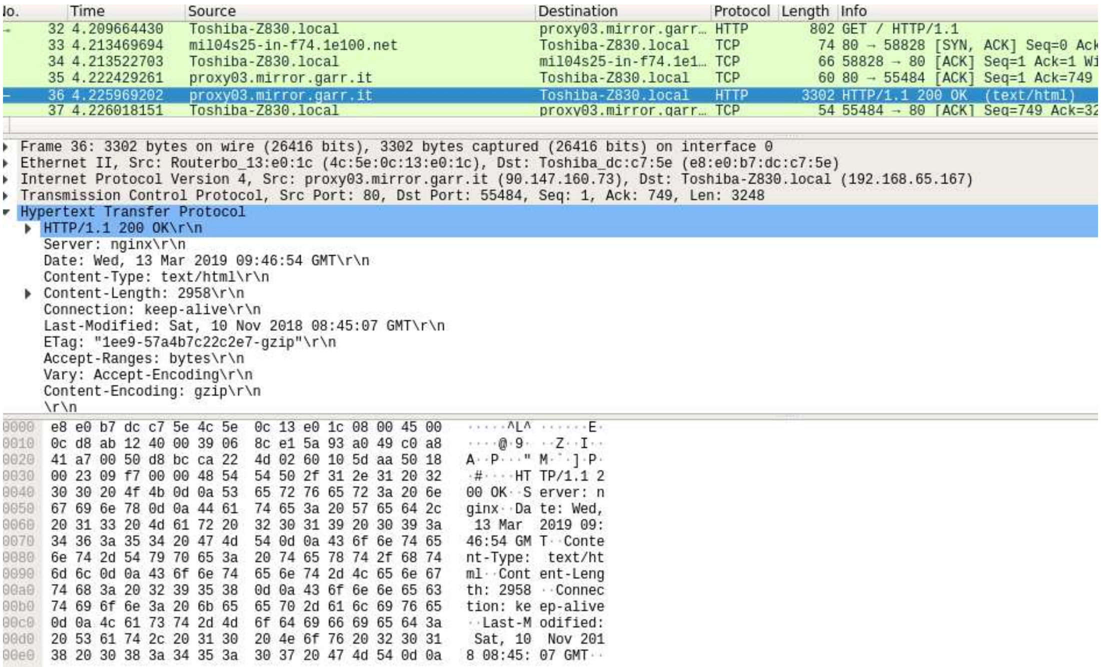
```
HTTP/1.1 200 OK\r\n
Date: Sun, 26 Sep 2010 20:09:20
   GMT\r\n
Server: Apache/2.0.52 (CentOS)\r\n
Last-Modified: Tue, 30 Oct 2007
   17:00:02 GMT\r\n
ETag: "17dc6-a5c-bf716880"\r\n
Accept-Ranges: bytes\r\n
Content-Length: 2652\r\n
Keep-Alive: timeout=10, max=100\r\n
Connection: Keep-Alive\r\n
Content-Type: text/html; charset=ISO-
   8859-1\r\n
\r\n
data data data data data ...
```

HTTP Response



```
HTTP/1.1 400 Bad Request
Date: Tue, 30 Feb 2022 13:28:52 GMT
Server: Apache/2.3.26 (Unix) Debian GNU/Linux
PHP/8.1.2
Connection: close
Content-Type: text/html; charset=iso-8859-1

Connessione all'host perduta.
```

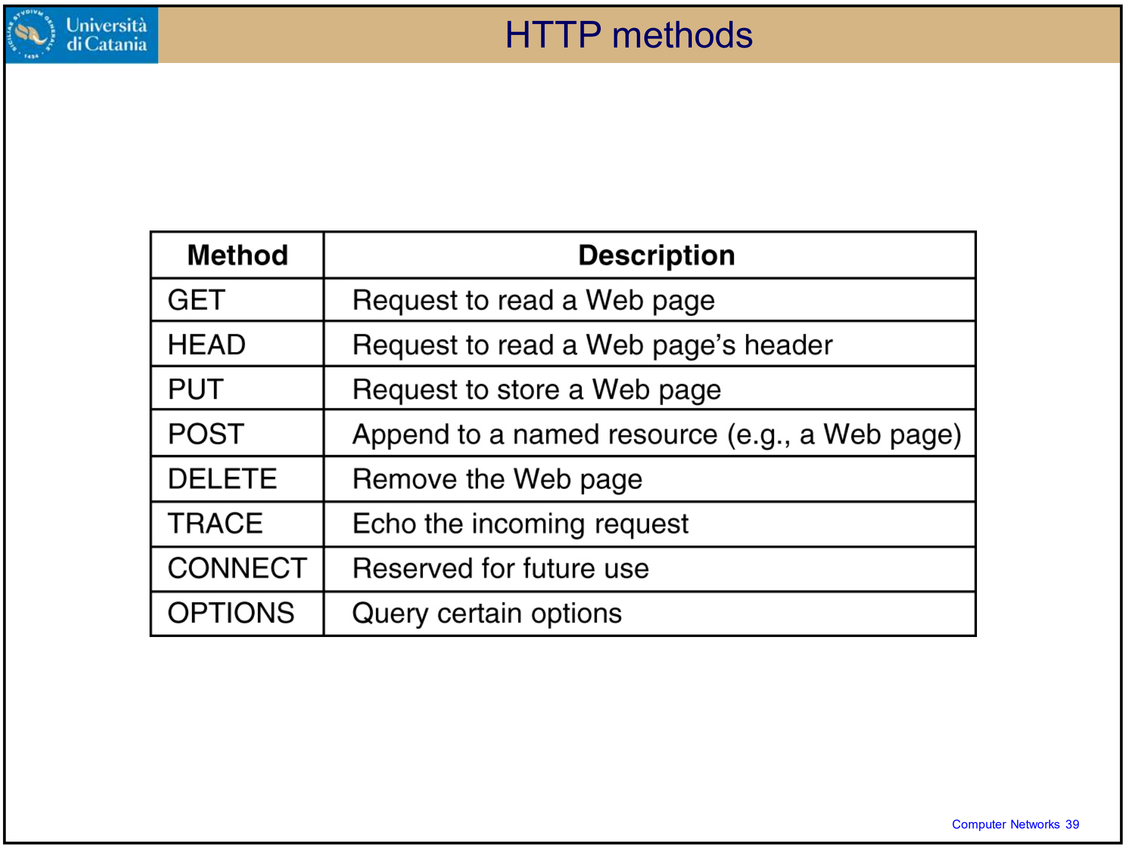
HTTP Response



```
lo.   Time          Source                    Destination          Protocol Length Info
 32 4.209664430   Toshiba-Z830.local        proxy03.mirror.garr…  HTTP      802 GET / HTTP/1.1
 33 4.213469694   mil04s25-in-f74.1e100.net Toshiba-Z830.local    TCP        74 80 → 58828 [SYN, ACK] Seq=0 Ack
 34 4.213522703   Toshiba-Z830.local        mil04s25-in-f74.1e1…  TCP        66 58828 → 80 [ACK] Seq=1 Ack=1 Wi
 35 4.222429261   proxy03.mirror.garr.it    Toshiba-Z830.local    TCP        60 80 → 55484 [ACK] Seq=1 Ack=749
 36 4.225969202   proxy03.mirror.garr.it    Toshiba-Z830.local    HTTP     3302 HTTP/1.1 200 OK  (text/html)
 37 4.226018151   Toshiba-Z830.local        proxy03.mirror.garr…  TCP        54 55484 → 80 [ACK] Seq=749 Ack=32
```

```
▸ Frame 36: 3302 bytes on wire (26416 bits), 3302 bytes captured (26416 bits) on interface 0
▸ Ethernet II, Src: Routerbo_13:e0:1c (4c:5e:0c:13:e0:1c), Dst: Toshiba_dc:c7:5e (e8:e0:b7:dc:c7:5e)
▸ Internet Protocol Version 4, Src: proxy03.mirror.garr.it (90.147.160.73), Dst: Toshiba-Z830.local (192.168.65.167)
▸ Transmission Control Protocol, Src Port: 80, Dst Port: 55484, Seq: 1, Ack: 749, Len: 3248
▾ Hypertext Transfer Protocol
  ▸ HTTP/1.1 200 OK\r\n
    Server: nginx\r\n
    Date: Wed, 13 Mar 2019 09:46:54 GMT\r\n
    Content-Type: text/html\r\n
  ▸ Content-Length: 2958\r\n
    Connection: keep-alive\r\n
    Last-Modified: Sat, 10 Nov 2018 08:45:07 GMT\r\n
    ETag: "1ee9-57a4b7c22c2e7-gzip"\r\n
    Accept-Ranges: bytes\r\n
    Vary: Accept-Encoding\r\n
    Content-Encoding: gzip\r\n
    \r\n
```

```
0000  e8 e0 b7 dc c7 5e 4c 5e  0c 13 e0 1c 08 00 45 00   ·····^L^ ······E·
0010  0c d8 ab 12 40 00 39 06  8c e1 5a 93 a0 49 c0 a8   ····@·9· ··Z··I··
0020  41 a7 00 50 d8 bc ca 22  4d 02 60 10 5d aa 50 18   A··P···" M·`·]·P·
0030  00 23 09 f7 00 00 48 54  54 50 2f 31 2e 31 20 32   ·#····HT TP/1.1 2
0040  30 30 20 4f 4b 0d 0a 53  65 72 76 65 72 3a 20 6e   00 OK··S erver: n
0050  67 69 6e 78 0d 0a 44 61  74 65 3a 20 57 65 64 2c   ginx··Da te: Wed,
0060  20 31 33 20 4d 61 72 20  32 30 31 39 20 30 39 3a    13 Mar  2019 09:
0070  34 36 3a 35 34 20 47 4d  54 0d 0a 43 6f 6e 74 65   46:54 GM T··Conte
0080  6e 74 2d 54 79 70 65 3a  20 74 65 78 74 2f 68 74   nt-Type:  text/ht
0090  6d 6c 0d 0a 43 6f 6e 74  65 6e 74 2d 4c 65 6e 67   ml··Cont ent-Leng
00a0  74 68 3a 20 32 39 35 38  0d 0a 43 6f 6e 6e 65 63   th: 2958 ··Connec
00b0  74 69 6f 6e 3a 20 6b 65  65 70 2d 61 6c 69 76 65   tion: ke ep-alive
00c0  0d 0a 4c 61 73 74 2d 4d  6f 64 69 66 69 65 64 3a   ··Last-M odified:
00d0  20 53 61 74 2c 20 31 30  20 4e 6f 76 20 32 30 31    Sat, 10  Nov 201
00e0  38 20 30 38 3a 34 35 3a  30 37 20 47 4d 54 0d 0a   8 08:45: 07 GMT··
```



HTTP methods

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page’s header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options

HTTP codes

Information Codes

100	Continue

Success Codes

200	OK
203	Non-Authoritative Information
204	No Content

Redirection Codes

305	Use Proxy

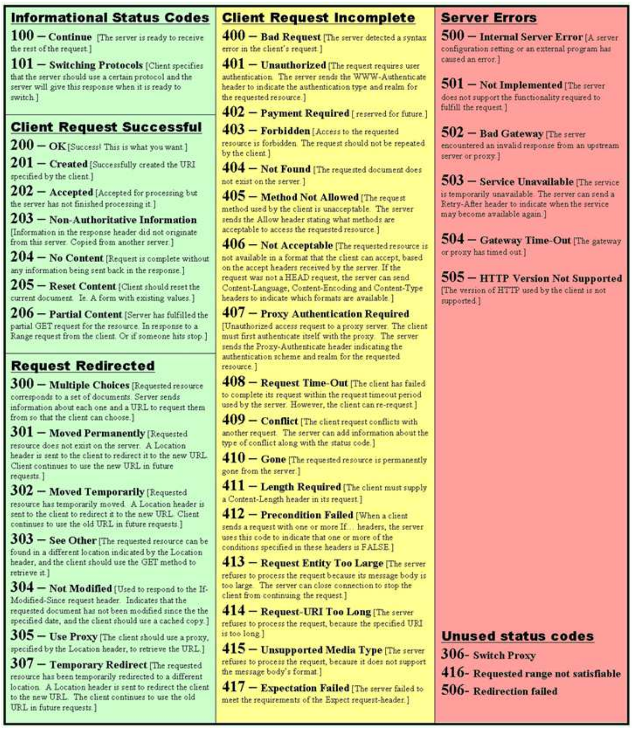
Client Error Codes

400	Bad Request
403	Forbidden
404	Not Found
405	Method Not Allowed

Server Error Codes

500	Internal Server Error
505	HTTP Version not supported

HTTP codes



Informational Status Codes

100 – Continue [The server is ready to receive the rest of the request.]

101 – Switching Protocols [Client specifies that the server should use a certain protocol and the server will give this response when it is ready to switch.]

Client Request Successful

200 – OK [Success! This is what you want.]

201 – Created [Successfully created the URI specified by the client.]

202 – Accepted [Accepted for processing but the server has not finished processing it.]

203 – Non-Authoritative Information [Information in the response header did not originate from this server. Copied from another server.]

204 – No Content [Request is complete without any information being sent back in the response.]

205 – Reset Content [Client should reset the current document. Ie. A form with existing values.]

206 – Partial Content [Server has fulfilled the partial GET request for the resource. In response to a Range request from the client. Or if someone hits stop.]

Request Redirected

300 – Multiple Choices [Requested resource corresponds to a set of documents. Server sends information about each one and a URL to request them from so that the client can choose.]

301 – Moved Permanently [Requested resource does not exist on the server. A Location header is sent to the client to redirect it to the new URL. Client continues to use the new URL in future requests.]

302 – Moved Temporarily [Requested resource has temporarily moved. A Location header is sent to the client to redirect it to the new URL. Client continues to use the old URL in future requests.]

303 – See Other [The requested resource can be found in a different location indicated by the Location header, and the client should use the GET method to retrieve it.]

304 – Not Modified [Used to respond to the If-Modified-Since request header. Indicates that the requested document has not been modified since the the specified date, and the client should use a cached copy.]

305 – Use Proxy [The client should use a proxy, specified by the Location header, to retrieve the URL.]

307 – Temporary Redirect [The requested resource has been temporarily redirected to a different location. A Location header is sent to redirect the client to the new URL. The client continues to use the old URL in future requests.]

Client Request Incomplete

400 – Bad Request [The server detected a syntax error in the client's request.]

401 – Unauthorized [The request requires user authentication. The server sends the WWW-Authenticate header to indicate the authentication type and realm for the requested resource.]

402 – Payment Required [reserved for future.]

403 – Forbidden [Access to the requested resource is forbidden. The request should not be repeated by the client.]

404 – Not Found [The requested document does not exist on the server.]

405 – Method Not Allowed [The request method used by the client is unacceptable. The server sends the Allow header stating what methods are acceptable to access the requested resource.]

406 – Not Acceptable [The requested resource is not available in a format that the client can accept, based on the accept headers received by the server. If the request was not a HEAD request, the server can send Content-Language, Content-Encoding and Content-Type headers to indicate which formats are available.]

407 – Proxy Authentication Required [Unauthorized access request to a proxy server. The client must first authenticate itself with the proxy. The server sends the Proxy-Authenticate header indicating the authentication scheme and realm for the requested resource.]

408 – Request Time-Out [The client has failed to complete its request within the request timeout period used by the server. However, the client can re-request.]

409 – Conflict [The client request conflicts with another request. The server can add information about the type of conflict along with the status code.]

410 – Gone [The requested resource is permanently gone from the server.]

411 – Length Required [The client must supply a Content-Length header in its request.]

412 – Precondition Failed [When a client sends a request with one or more If... headers, the server uses this code to indicate that one or more of the conditions specified in these headers is FALSE.]

413 – Request Entity Too Large [The server refuses to process the request because its message body is too large. The server can close connection to stop the client from continuing the request.]

414 – Request-URI Too Long [The server refuses to process the request, because the specified URI is too long.]

415 – Unsupported Media Type [The server refuses to process the request, because it does not support the message body's format.]

417 – Expectation Failed [The server failed to meet the requirements of the Expect request-header.]

Server Errors

500 – Internal Server Error [A server configuration setting or an external program has caused an error.]

501 – Not Implemented [The server does not support the functionality required to fulfill the request.]

502 – Bad Gateway [The server encountered an invalid response from an upstream server or proxy.]

503 – Service Unavailable [The service is temporarily unavailable. The server can send a Retry-After header to indicate when the service may become available again.]

504 – Gateway Time-Out [The gateway or proxy has timed out.]

505 – HTTP Version Not Supported [The version of HTTP used by the client is not supported.]

Unused status codes

306- Switch Proxy

416- Requested range not satisfiable

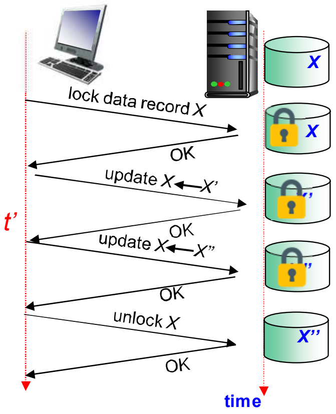
506- Redirection failed

Maintaining user/server state: cookies

Recall: HTTP GET/response interaction is *stateless*

- no notion of multi-step exchanges of HTTP messages to complete a Web "transaction"
  - no need for client/server to track "state" of multi-step exchange
  - all HTTP requests are independent of each other
  - no need for client/server to "recover" from a partially-completed-but-never-completely-completed transaction

a stateful protocol: client makes two changes to X, or none at all



Q: what happens if network connection or client crashes at *t'* ?

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

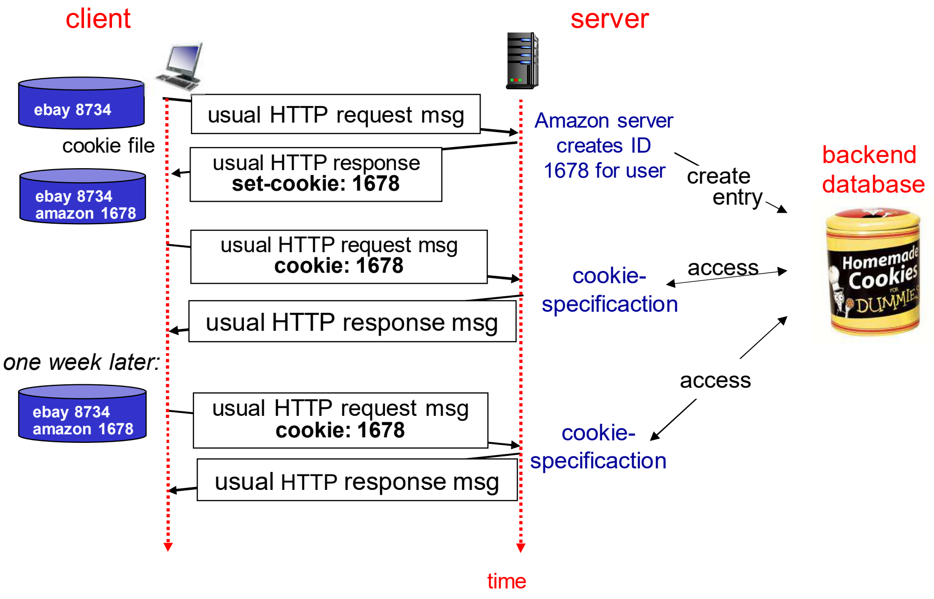
3) cookie file kept on user's host, managed by user's browser

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



client

server

ebay 8734

cookie file

ebay 8734
amazon 1678

usual HTTP request msg

Amazon server
creates ID
1678 for user

create
entry

backend
database

usual HTTP response
set-cookie: 1678

usual HTTP request msg
cookie: 1678

cookie-
specificaction

access

usual HTTP response msg

one week later:

ebay 8734
amazon 1678

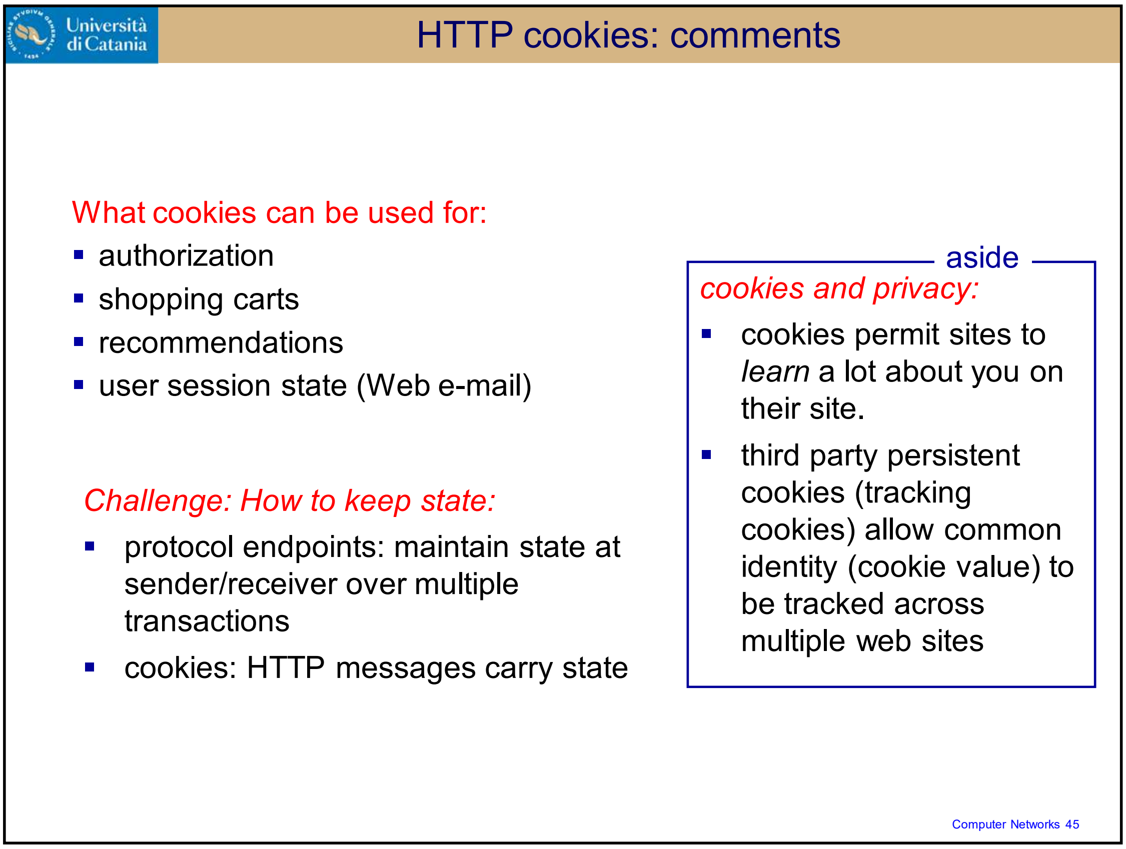
usual HTTP request msg
cookie: 1678

access

cookie-
specificaction

usual HTTP response msg

time



HTTP cookies: comments

What cookies can be used for:

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

Challenge: How to keep state:

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

aside

cookies and privacy:

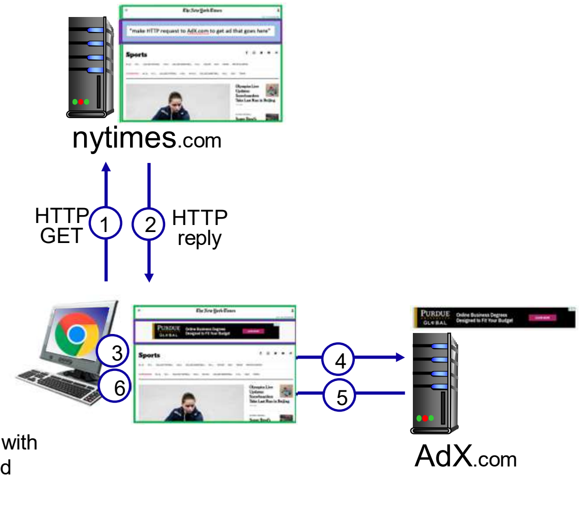
- cookies permit sites to *learn* a lot about you on their site.
- third party persistent cookies (tracking cookies) allow common identity (cookie value) to be tracked across multiple web sites

Example: displaying a NY Times web page

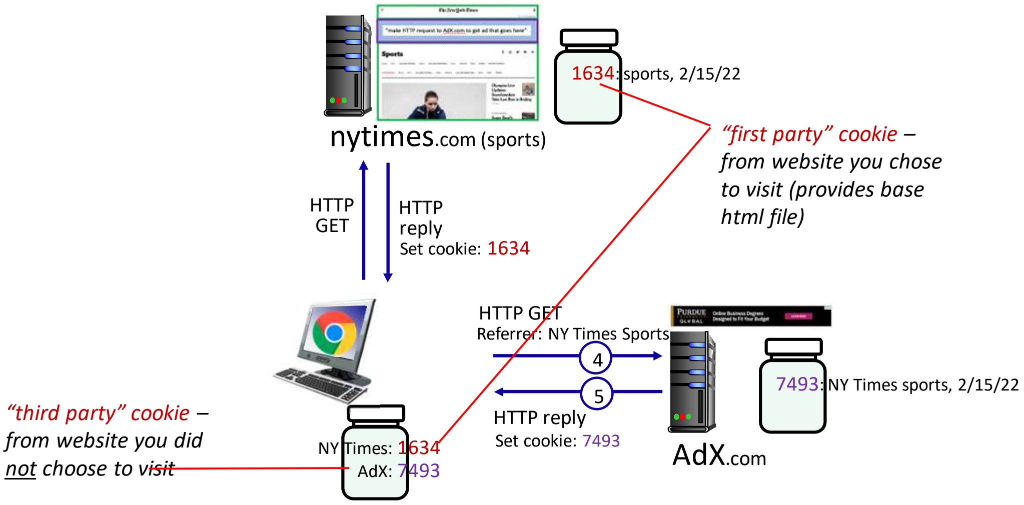
1 2 GET base html file from nytimes.com

4 5 fetch ad from AdX.com

7 display composed page


"make HTTP request to AdX.com to get ad that goes here"
Sports
nytimes.com
HTTP GET
1
2
HTTP reply
PURDUE GLOBAL
Online Business Degrees Designed to Fit Your Budget
3
6
4
5
AdX.com
NY times page with embedded ad displayed

Cookies: tracking a user’s browsing behavior



nytimes.com (sports)

“make HTTP request to AdX.com to get ad that goes here”

1634: sports, 2/15/22

“first party” cookie – from website you chose to visit (provides base html file)

HTTP GET

HTTP reply
Set cookie: 1634

HTTP GET
Referrer: NY Times Sports

4

5

HTTP reply
Set cookie: 7493

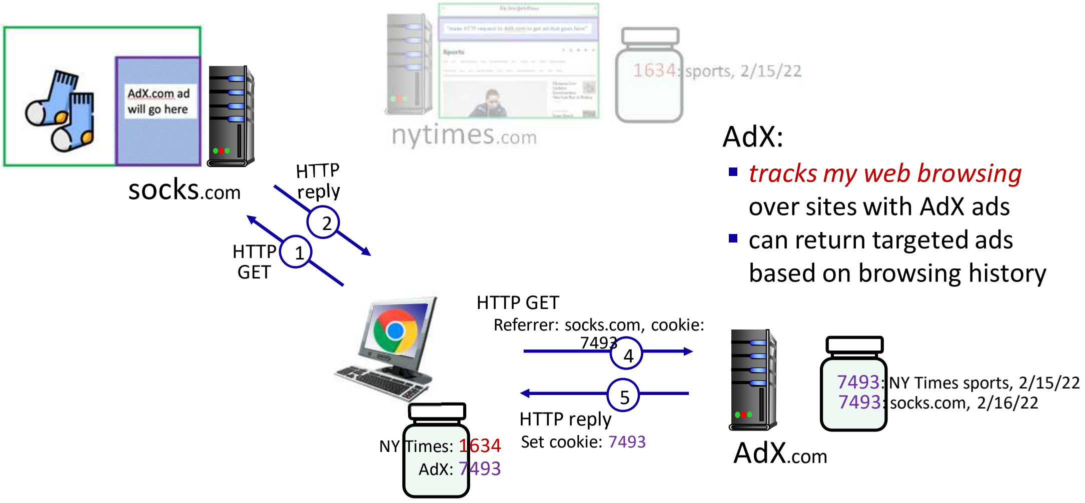
AdX.com

7493: NY Times sports, 2/15/22

NY Times: 1634
AdX: 7493

“third party” cookie – from website you did <u>not</u> choose to visit

Cookies: tracking a user's browsing behavior

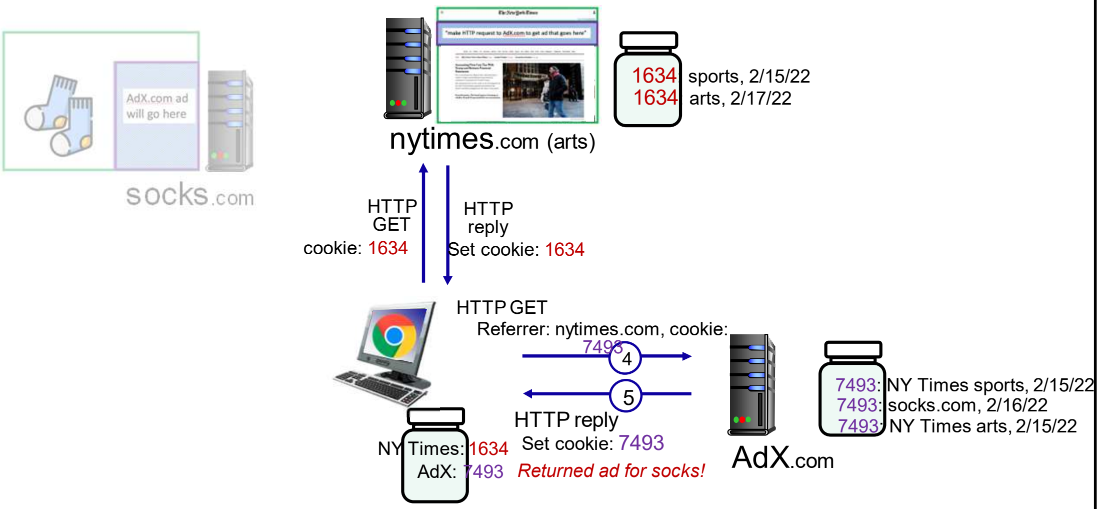


AdX:

- *tracks my web browsing* over sites with AdX ads
- can return targeted ads based on browsing history

Cookies: tracking a user's browsing behavior

(one day later)



nytimes.com (arts)

1634 sports, 2/15/22
1634 arts, 2/17/22

HTTP GET
cookie: 1634

HTTP reply
Set cookie: 1634

HTTP GET
Referrer: nytimes.com, cookie: 7493

4

5

HTTP reply
Set cookie: 7493

Returned ad for socks!

NY Times: 1634
AdX: 7493

AdX.com

7493: NY Times sports, 2/15/22
7493: socks.com, 2/16/22
7493: NY Times arts, 2/15/22

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

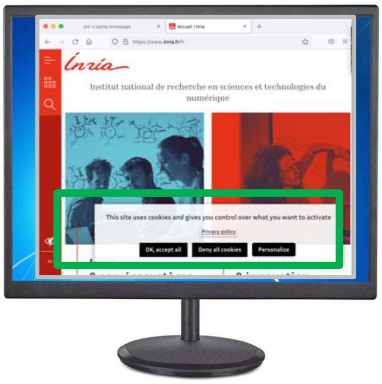
GDPR and cookies

GDPR (EU General Data Protection Regulation)

> "Natural persons may be associated with online identifiers […] such as internet protocol addresses, cookie identifiers or other identifiers […].
>
> This may leave traces which, in particular when combined with unique identifiers and other information received by the servers, may be used to create profiles of the natural persons and identify them."

GDPR, recital 30 (May 2018)

when cookies can identify an individual, cookies are considered personal data, subject to GDPR personal data regulations

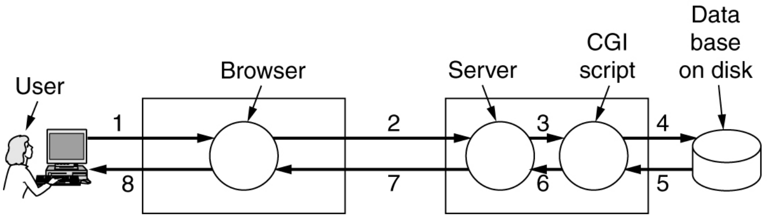


Institut national de recherche en sciences et technologies du numérique

This site uses cookies and gives you control over what you want to activate

User has explicit control over whether or not cookies are allowed

DHTML



1. User fills in form
2. Form sent back
3. Handed to CGI
4. CGI queries DB
5. Record found
6. CGI builds page
7. Page returned
8. Page displayed

HTTP/2

Key goal: decreased delay in multi-object HTTP requests.

HTTP1.1: introduced multiple, pipelined GETs over single TCP connection

- server responds *in-order* (FCFS: first-come- first-served scheduling) to GET requests
- with FCFS, small object may have to wait for transmission (head-of-line (HOL) blocking) behind large object(s)
- loss recovery (retransmitting lost TCP segments) stalls object transmission

HTTP/2

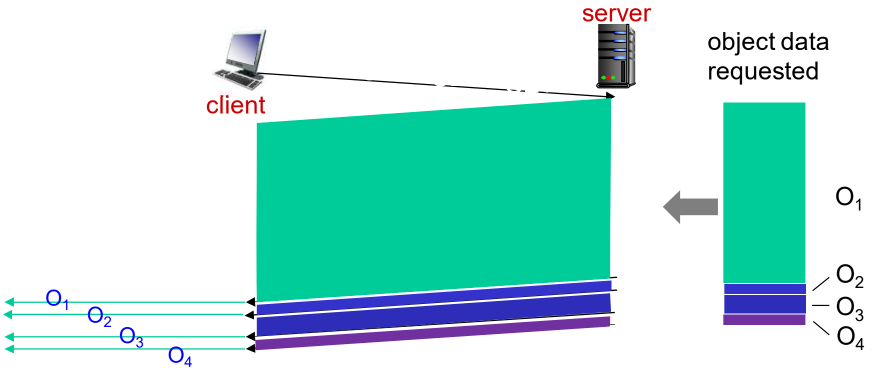
Key goal: decreased delay in multi-object HTTP requests.

HTTP/2: [RFC 7540, 2015] increased flexibility at *server* in sending objects to client:

- methods, status codes, most header fields unchanged from HTTP 1.1
- transmission order of requested objects based on client-specified object priority (not necessarily FCFS)
- *push* unrequested objects to client
- divide objects into frames, schedule frames to mitigate HOL blocking

HTTP/2: mitigating HOL blocking

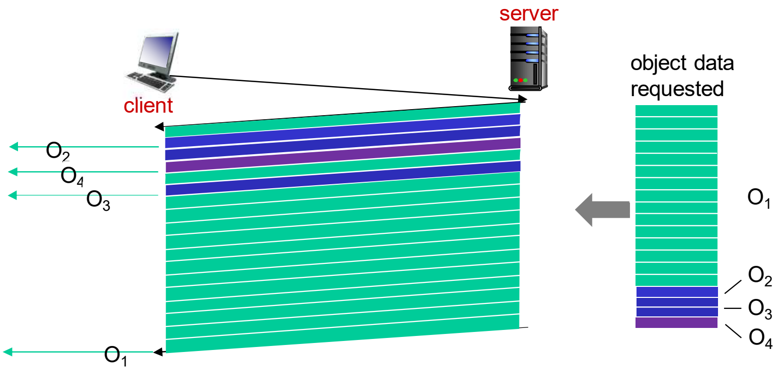
HTTP 1.1: client requests 1 large object (e.g., video file, and 3 smaller objects)



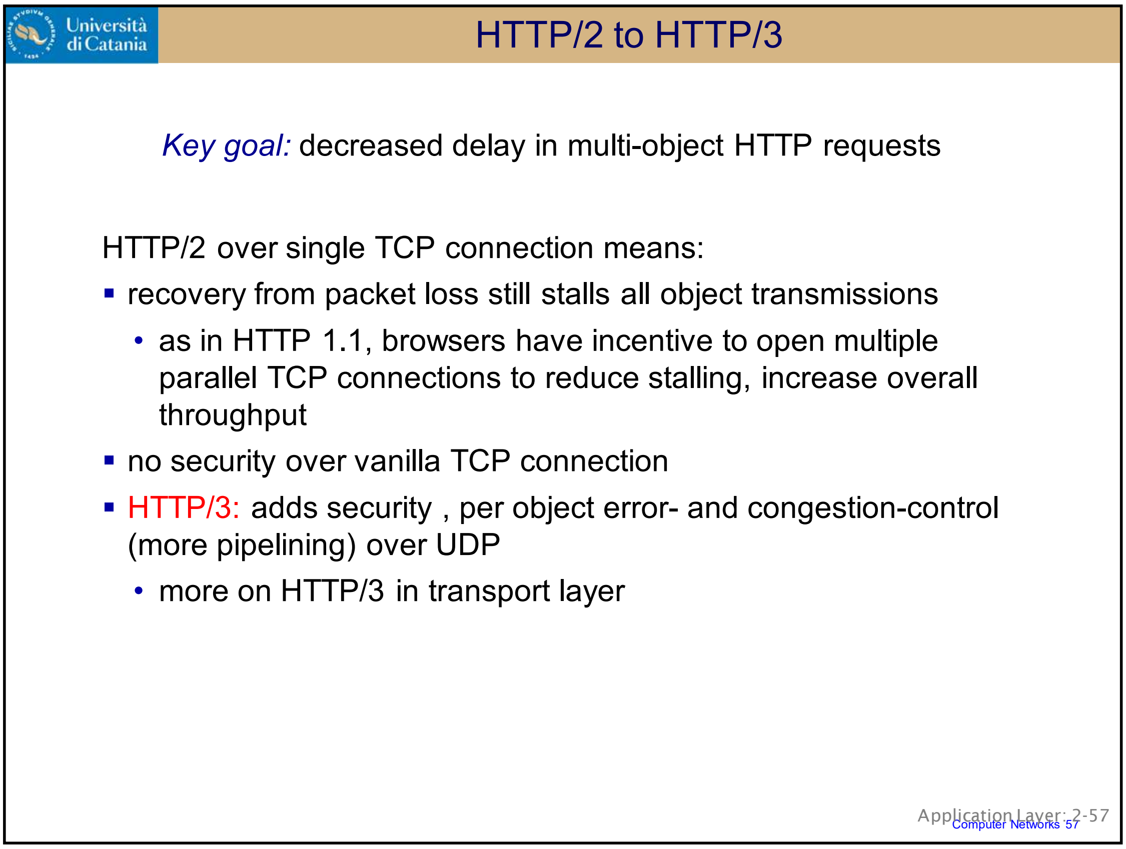
objects delivered in order requested: O_2, O_3, O_4 wait behind O_1

HTTP/2: mitigating HOL blocking

HTTP/2: objects divided into frames, transmitted in an interleaved way



O_2, O_3, O_4 delivered quickly, O_1 slightly delayed



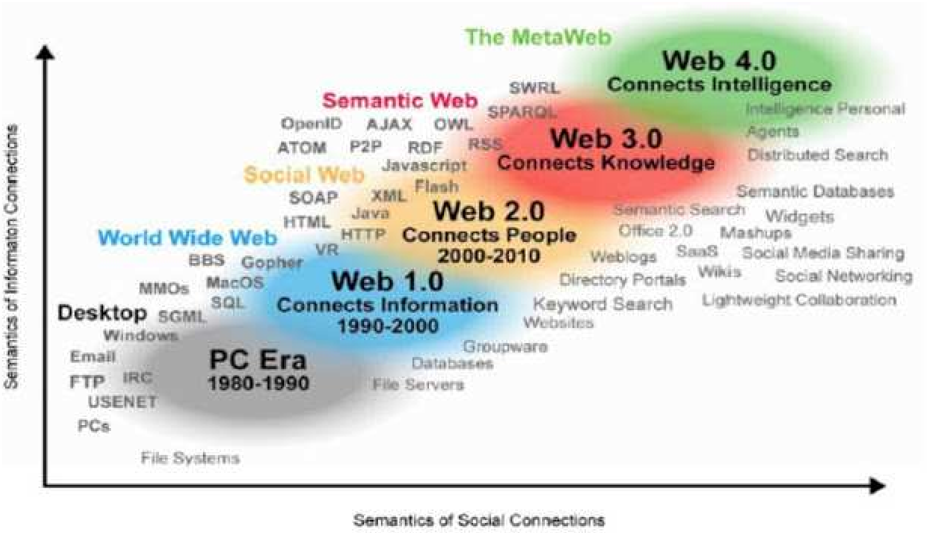
HTTP/2 to HTTP/3

Key goal: decreased delay in multi-object HTTP requests

HTTP/2 over single TCP connection means:

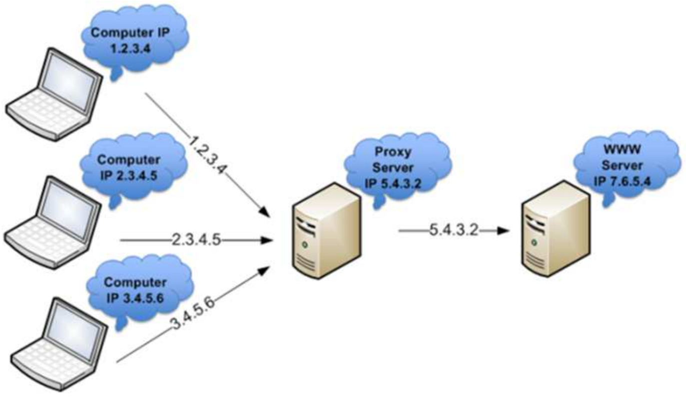
- recovery from packet loss still stalls all object transmissions
  - as in HTTP 1.1, browsers have incentive to open multiple parallel TCP connections to reduce stalling, increase overall throughput
- no security over vanilla TCP connection
- HTTP/3: adds security , per object error- and congestion-control (more pipelining) over UDP
  - more on HTTP/3 in transport layer

Web Evolution

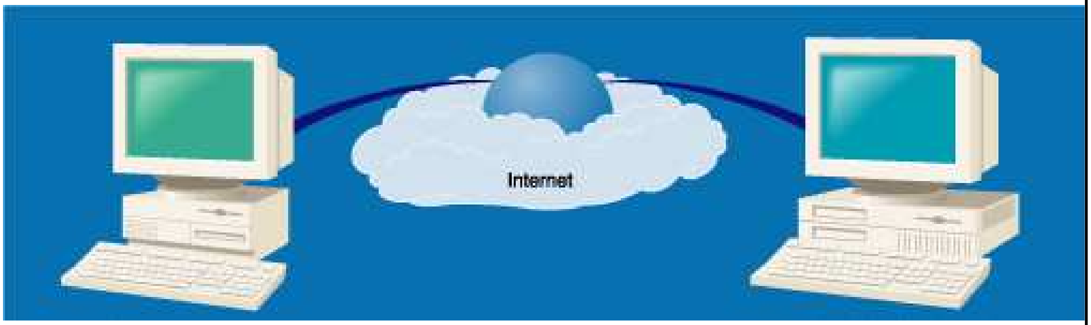


Università di Catania

Proxy server



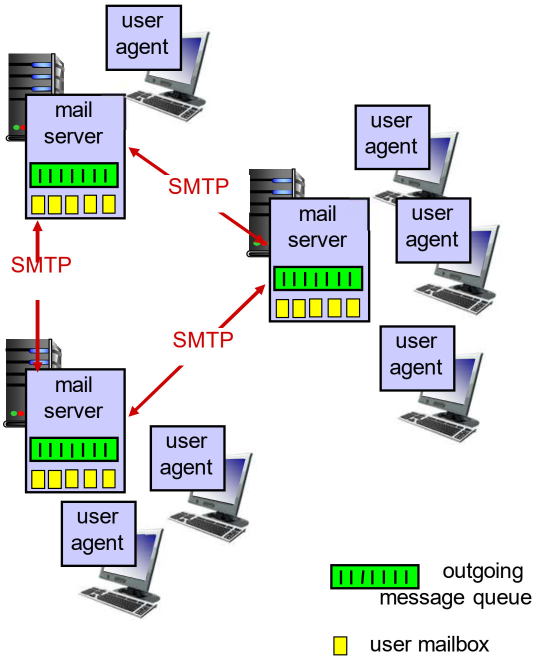
SMTP

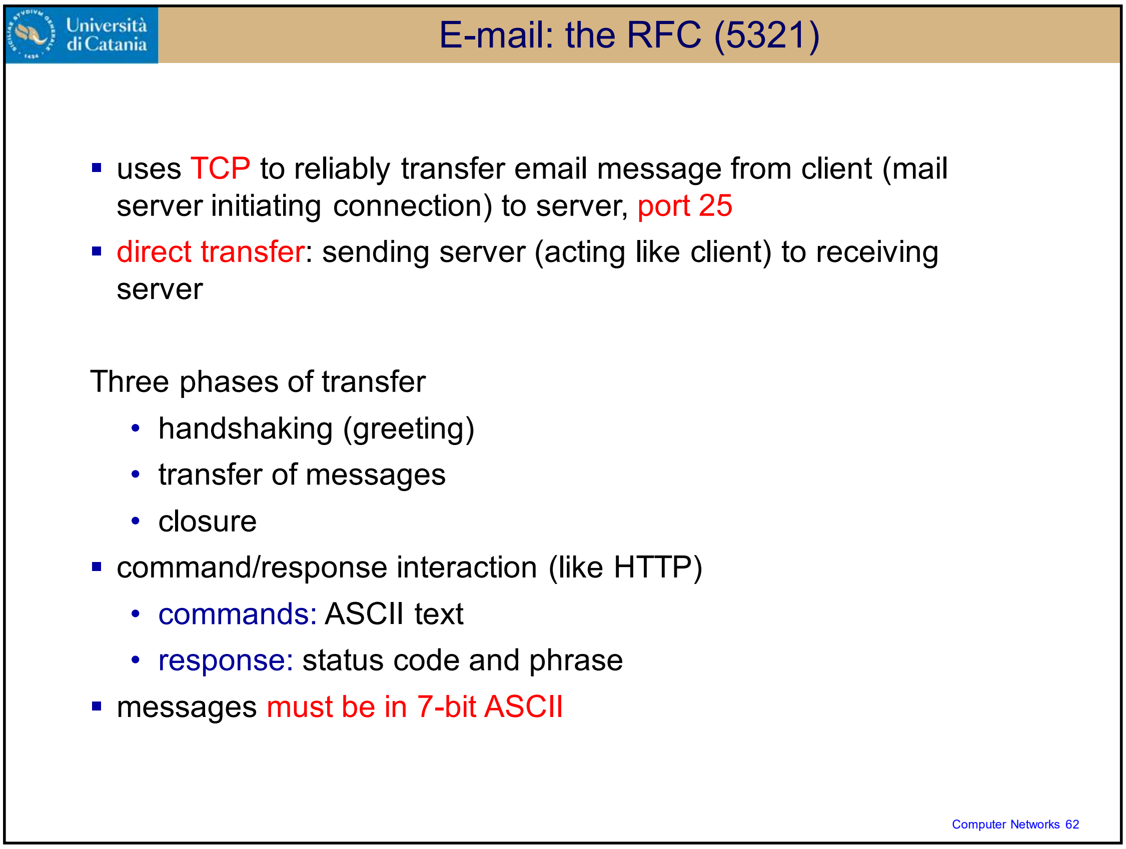


SMTP

Three major components:

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





E-mail: the 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

- handshaking (greeting)
- transfer of messages
- closure

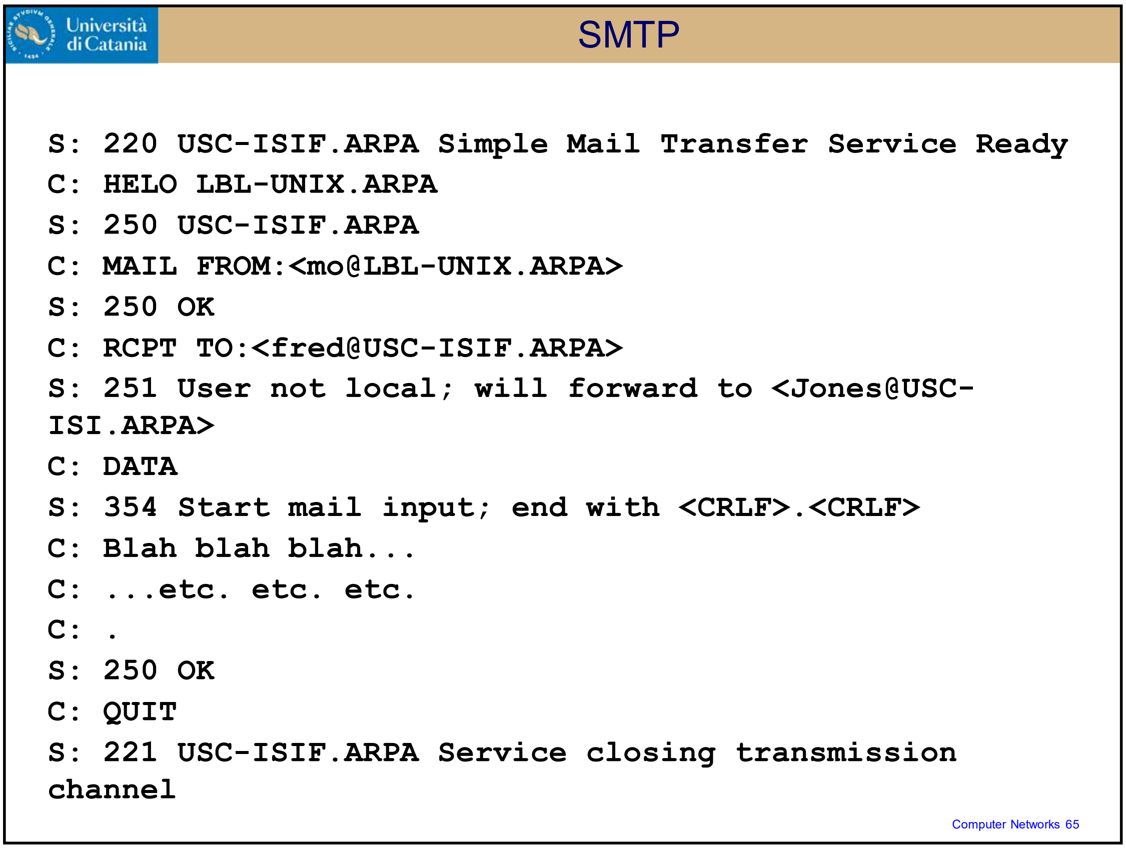
- command/response interaction (like HTTP)
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCII

SMTP

```
S: 220 BBN-UNIX.ARPA Simple Mail Transfer Service Ready
C: HELO USC-ISIF.ARPA
S: 250 Hello BBN-UNIX.ARPA, pleased to meet you
C: MAIL FROM:<Smith@USC-ISIF.ARPA>
S: 250 OK
C: RCPT TO:<Jones@BBN-UNIX.ARPA>
S: 250 OK
C: RCPT TO:<Green@BBN-UNIX.ARPA>
S: 550 No such user here
C: RCPT TO:<Brown@BBN-UNIX.ARPA>
S: 250 OK
```

SMTP

```
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: Blah blah blah...
C: ...etc. etc. etc.
C: .
S: 250 OK
C: QUIT
S: 221 BBN-UNIX.ARPA Service closing transmission
channel
```



SMTP

```
S: 220 USC-ISIF.ARPA Simple Mail Transfer Service Ready
C: HELO LBL-UNIX.ARPA
S: 250 USC-ISIF.ARPA
C: MAIL FROM:<mo@LBL-UNIX.ARPA>
S: 250 OK
C: RCPT TO:<fred@USC-ISIF.ARPA>
S: 251 User not local; will forward to <Jones@USC-
ISI.ARPA>
C: DATA
S: 354 Start mail input; end with <CRLF>.<CRLF>
C: Blah blah blah...
C: ...etc. etc. etc.
C: .
S: 250 OK
C: QUIT
S: 221 USC-ISIF.ARPA Service closing transmission
channel
```

SMTP

```
                  S: 220 xyz.com SMTP service ready
C: HELO abcd.com
                  S: 250 xyz.com says hello to abcd.com
C: MAIL FROM: <elinor@abcd.com>
                  S: 250 sender ok
C: RCPT TO: <carolyn@xyz.com>
                  S: 250 recipient ok
C: DATA
                  S: 354 Send mail; end with "." on a line by itself
C: From: elinor@abcd.com
C: To: carolyn@xyz.com
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@abcd.com>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: Earth orbits sun integral number of times
C:
C: This is the preamble. The user agent ignores it. Have a nice day.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/enriched
C:
C: Happy birthday to you
C: Happy birthday to you
C: Happy birthday dear <bold> Carolyn </bold>
C: Happy birthday to you
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: message/external-body;
C:         access-type="anon-ftp";
C:         site="bicycle.abcd.com";
C:         directory="pub";
C:         name="birthday.snd"
C:
C: content-type: audio/basic
C: content-transfer-encoding: base64
C: --qwertyuiopasdfghjklzxcvbnm
C: .
                  S: 250 message accepted
C: QUIT
                  S: 221 xyz.com closing connection
```

Try SMTP interaction for yourself:

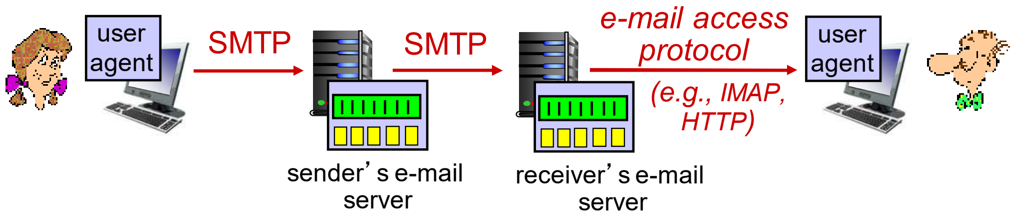
```
telnet <servername> 25
```

- see 220 reply from server
- enter HELO, MAIL FROM:, RCPT TO:, DATA, QUIT commands

above lets you send email without using e-mail client (reader)

Note: this will only work if <servername> allows telnet connections to port 25 (this is becoming increasingly rare because of security concerns)

SMTP


user agent
SMTP
SMTP
e-mail access protocol
(e.g., IMAP, HTTP)
user agent
sender’s e-mail server
receiver’s e-mail server

SMTP

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

SMTP

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

POP

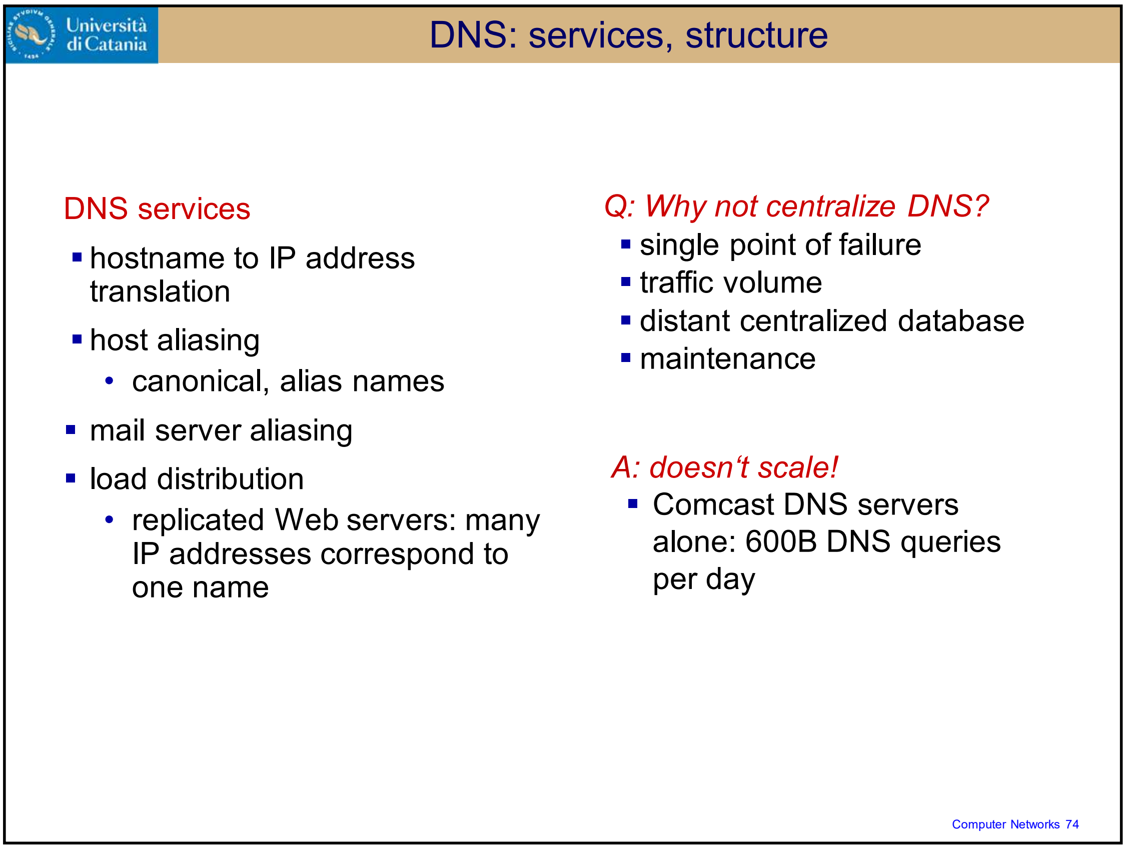
```
                        S: +OK POP3 server ready
C: USER carolyn
                        S: +OK
C: PASS vegetables
                        S: +OK login successful
C: LIST
                        S: 1 2505
                        S: 2 14302
                        S: 3 8122
                        S: .
C: RETR 1
                        S: (sends message 1)
C: DELE 1
C: RETR 2
                        S: (sends message 2)
C: DELE 2
C: RETR 3
                        S: (sends message 3)
C: DELE 3
C: QUIT
                        S: +OK POP3 server disconnecting
```

POP vs IMAP

Feature	POP3	IMAP
Where is protocol defined?	RFC 1939	RFC 2060
Which TCP port is used?	110	143
Where is e-mail stored?	User's PC	Server
Where is e-mail read?	Off-line	On-line
Connect time required?	Little	Much
Use of server resources?	Minimal	Extensive
Multiple mailboxes?	No	Yes
Who backs up mailboxes?	User	ISP
Good for mobile users?	No	Yes
User control over downloading?	Little	Great
Partial message downloads?	No	Yes
Are disk quotas a problem?	No	Could be in time
Simple to implement?	Yes	No
Widespread support?	Yes	Growing

DNS

151.97.240.18	www.dmi.unict.it
151.97.240.4	www.unict.it
151.97.6.236	www.ing.unict.it
151.97.252.132	galileo.dmi.unict.it



DNS: services, structure

DNS services

- hostname to IP address translation
- host aliasing
  - canonical, alias names
- mail server aliasing
- load distribution
  - replicated Web servers: many IP addresses correspond to one name

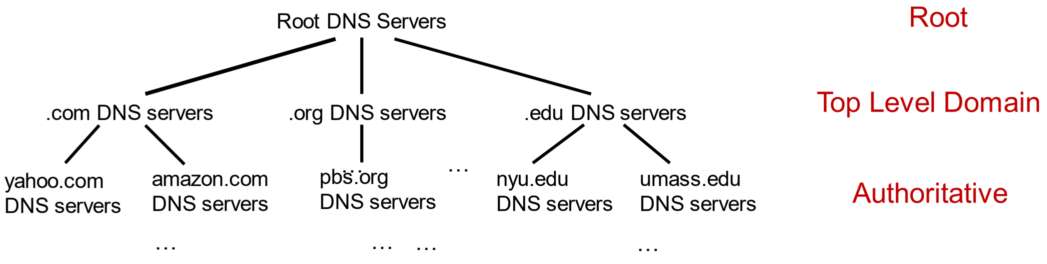
Q: Why not centralize DNS?

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

A: doesn‘t scale!

- Comcast DNS servers alone: 600B DNS queries per day

DNS: a distributed, hierarchical database



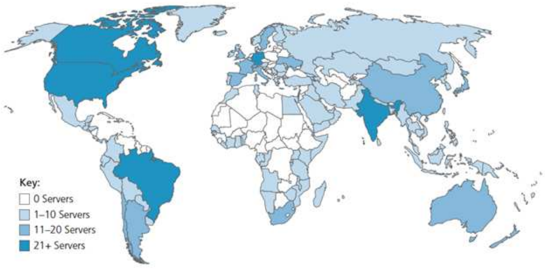
Client wants IP address for www.amazon.com; 1st approximation:

- 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

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 and 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)



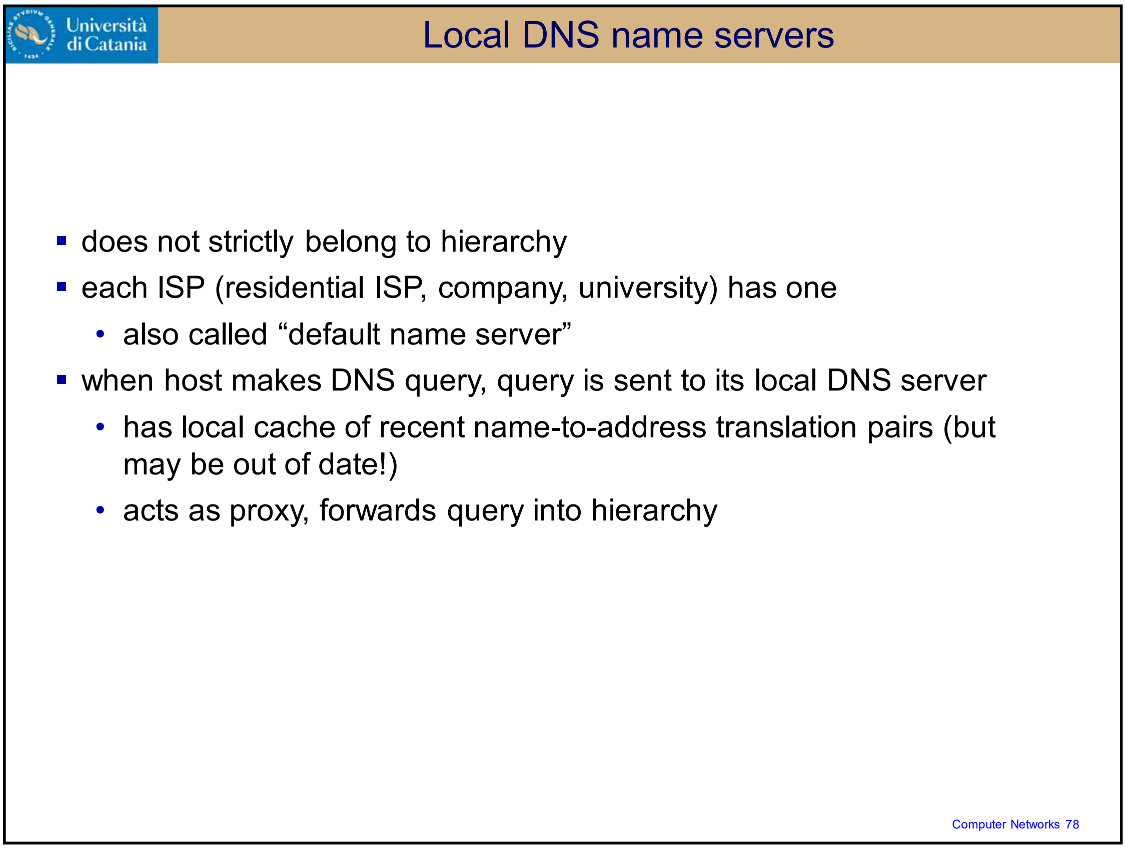
TLD: authoritative servers

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
- Network Solutions: authoritative registry for .com, .net 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

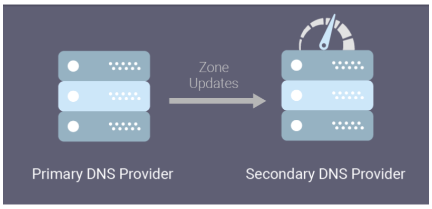
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
  - also called “default name server”
- when host makes DNS query, query is sent to its local DNS server
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy

Primary vs secondary DNS

Primary DNS is the main authoritative DNS server (or nameserver) that serves as the initial stop for a query as the user-entered domain name is translated into an IP address.

Secondary DNS service affords you an extra set of authoritative nameservers to answer queries for your domain. The information that is stored on both nameservers is identical. Secondary DNS allows your domain zone file to be backed up automatically and stored as a copy on a secondary server.

Administrators can modify only records on primary DNS!

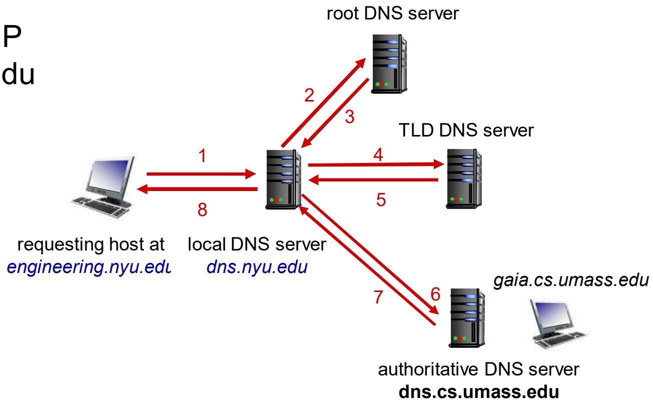


DNS name resolution: iterated query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Iterated query:

- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"

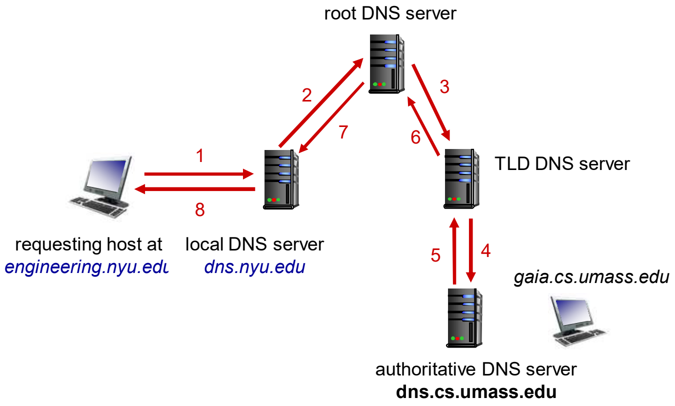


DNS name resolution: recursive query

Example: host at engineering.nyu.edu wants IP address for gaia.cs.umass.edu

Recursive query:

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



Caching, Updating DNS Records

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time (TTL)
  - TLD servers typically cached in local name servers
    - thus root name servers not often visited
- cached entries may be *out-of-date* (best-effort name-to-address translation!)
  - if name host changes IP address, may not be known Internet-wide until all TTLs expire!
- update/notify mechanisms proposed IETF standard
  - RFC 2136

DNS records

DNS: distributed database storing resource records (RR)

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

type=A
- **`name`** is hostname
- **`value`** is IP address

type=NS
- **`name`** is domain (e.g., foo.com)
- **`value`** is hostname of authoritative name server for this domain

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

type=MX
- **`value`** is name of mailserver associated with name

Record DNS

```
;; QUESTION SECTION:;
www.google.it.          IN  A
;; ANSWER SECTION:
www.google.it.      190592  IN  CNAME   www.google.com.
www.google.com.     431593  IN  CNAME   www.l.google.com.
www.l.google.com.       111 IN  A   74.125.39.147
www.l.google.com.       111 IN  A   74.125.39.99
www.l.google.com.       111 IN  A   74.125.39.103
www.l.google.com.       111 IN  A   74.125.39.104
;; AUTHORITY SECTION:
l.google.com.       7217    IN  NS  e.l.google.com.
l.google.com.       7217    IN  NS  f.l.google.com.
l.google.com.       7217    IN  NS  g.l.google.com.
l.google.com.       7217    IN  NS  a.l.google.com.
l.google.com.       7217    IN  NS  b.l.google.com.
l.google.com.       7217    IN  NS  c.l.google.com.
l.google.com.       7217    IN  NS  d.l.google.com.
```

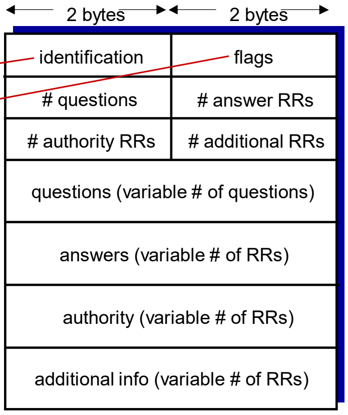
Record DNS

```
nslookup -q=mx google.com (windows)
Server:  DCA.dmi.unict.it
Address:  151.97.252.165

Risposta da un server non autorevole:
google.com      MX preference = 50, mail exchanger = alt4.aspmx.l.google.com
google.com      MX preference = 40, mail exchanger = alt3.aspmx.l.google.com
google.com      MX preference = 10, mail exchanger = aspmx.l.google.com
google.com      MX preference = 30, mail exchanger = alt2.aspmx.l.google.com
google.com      MX preference = 20, mail exchanger = alt1.aspmx.l.google.com

alt4.aspmx.l.google.com internet address = 173.194.202.26
alt4.aspmx.l.google.com AAAA IPv6 address = 2607:f8b0:400e:c00::1a
alt3.aspmx.l.google.com internet address = 142.250.157.26
alt3.aspmx.l.google.com AAAA IPv6 address = 2404:6800:4008:c13::1b
aspmx.l.google.com      internet address = 108.177.127.26
aspmx.l.google.com      AAAA IPv6 address = 2a00:1450:4013:c01::1a
alt2.aspmx.l.google.com internet address = 74.125.200.26
alt2.aspmx.l.google.com AAAA IPv6 address = 2404:6800:4003:c00::1a
alt1.aspmx.l.google.com internet address = 142.250.150.27
alt1.aspmx.l.google.com AAAA IPv6 address = 2a00:1450:4010:c1c::1a
```

DNS protocol messages

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

- identification:
- flags:



2 bytes

2 bytes

identification

flags

# questions

# answer RRs

# authority RRs

additional RRs

questions (variable # of questions)

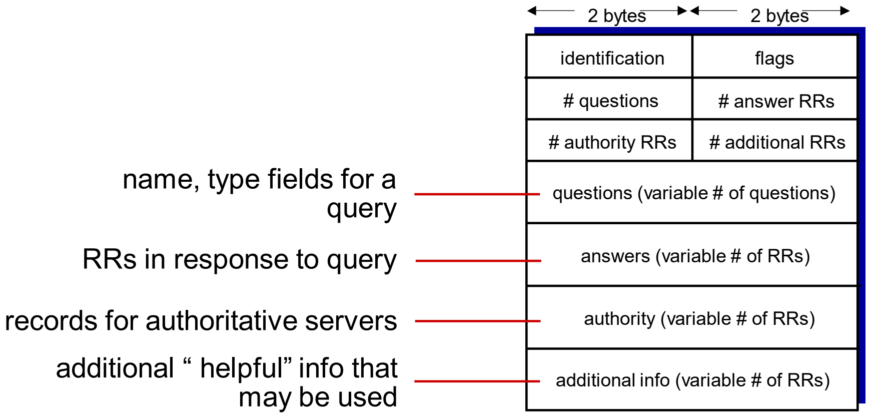
answers (variable # of RRs)

authority (variable # of RRs)

additional info (variable # of RRs)

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

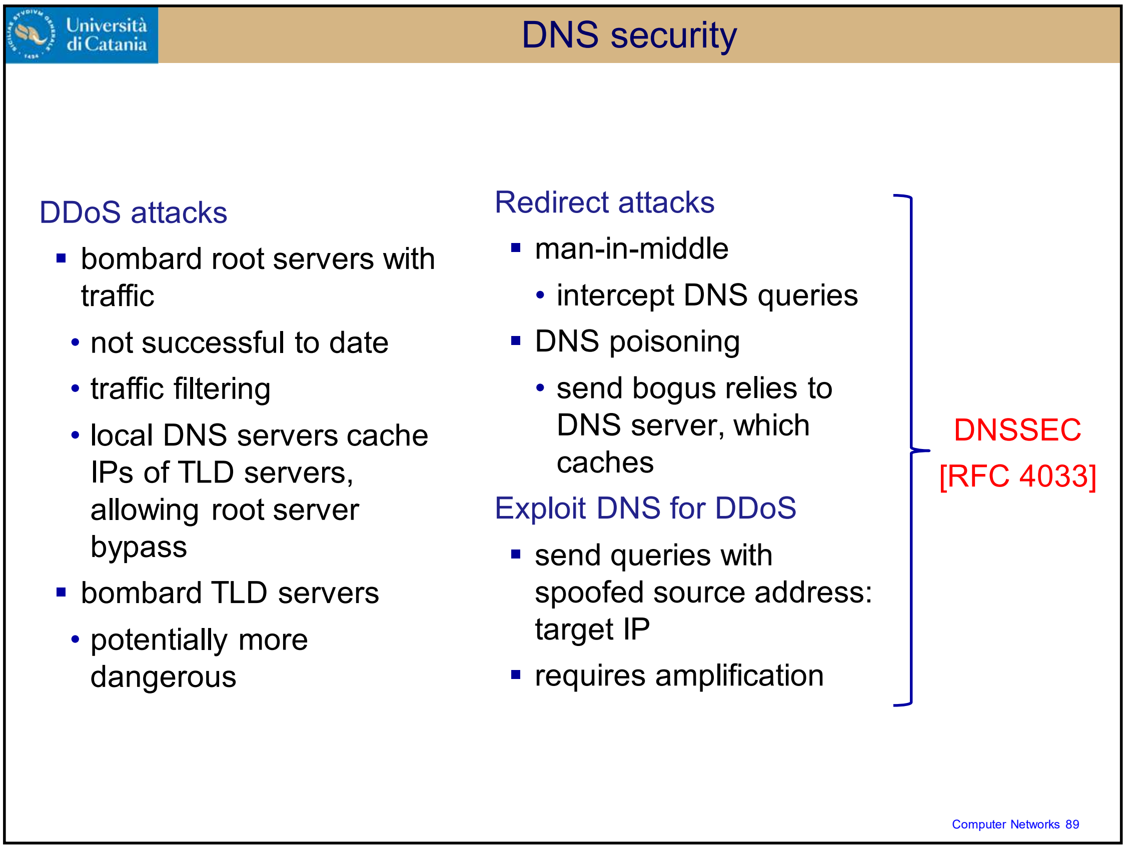
Inserting records into DNS

Example: new startup “**`Network Utopia`**”

- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts NS, A RRs into .com TLD server:

```
(networkutopia.com, dns1.networkutopia.com, NS)
 (dns1.networkutopia.com, 212.212.212.1, A)
```

- create authoritative server locally with IP address 212.212.212.1
  - type A record for **`www.networkuptopia.com`**
  - type MX record for **`networkutopia.com`**



DNS security

DDoS attacks

- bombard root servers with traffic
  - not successful to date
  - traffic filtering
  - local DNS servers cache IPs of TLD servers, allowing root server bypass
- bombard TLD servers
  - potentially more dangerous

Redirect attacks

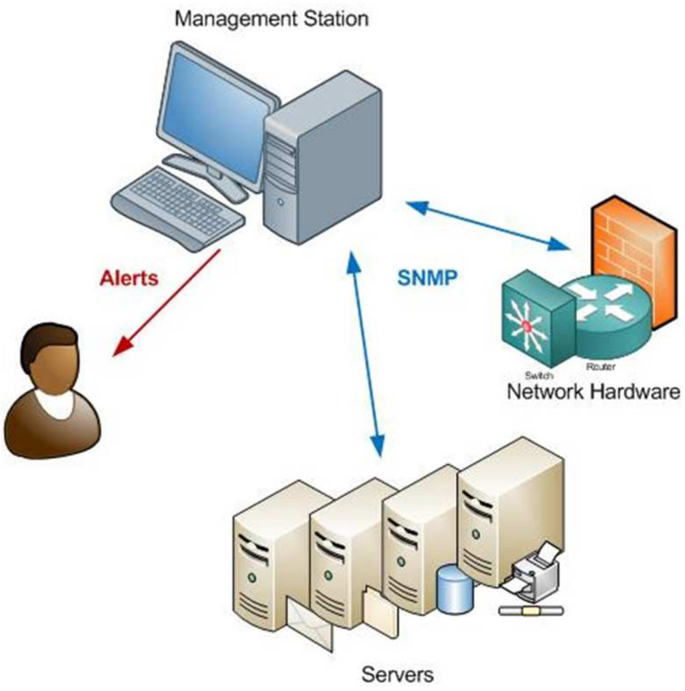
- man-in-middle
  - intercept DNS queries
- DNS poisoning
  - send bogus relies to DNS server, which caches

Exploit DNS for DDoS

- send queries with spoofed source address: target IP
- requires amplification

DNSSEC [RFC 4033]

Network management



SNMP

SNMP **S**imple **N**etwork **M**anagement **P**rotocol

SNMP has been subject to many update and re-design processes:

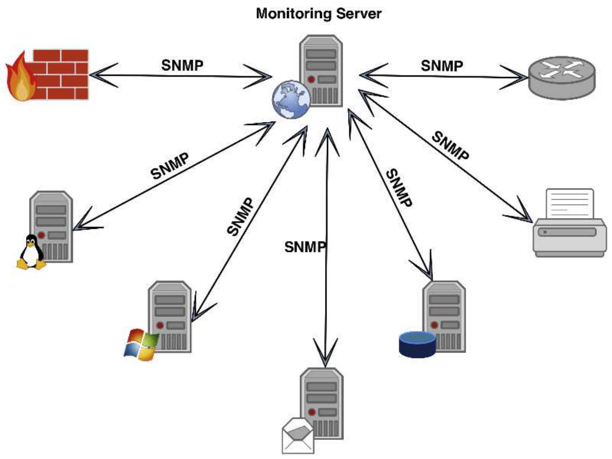
- to introduce security issues
- to have more flexible handler model
- to maintain compatibility towards legacy systems
- …
- to handle not only network systems.

SNMP

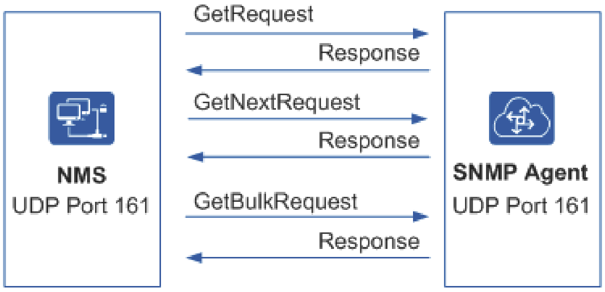
It is composed by a manager and some agents which handle variables that represent system objects to handling/monitoring

The manager performs *get* and *set* operations

The agents normally wait for requests, but can send *trap*



SNMP



NMS
UDP Port 161

GetRequest

Response

GetNextRequest

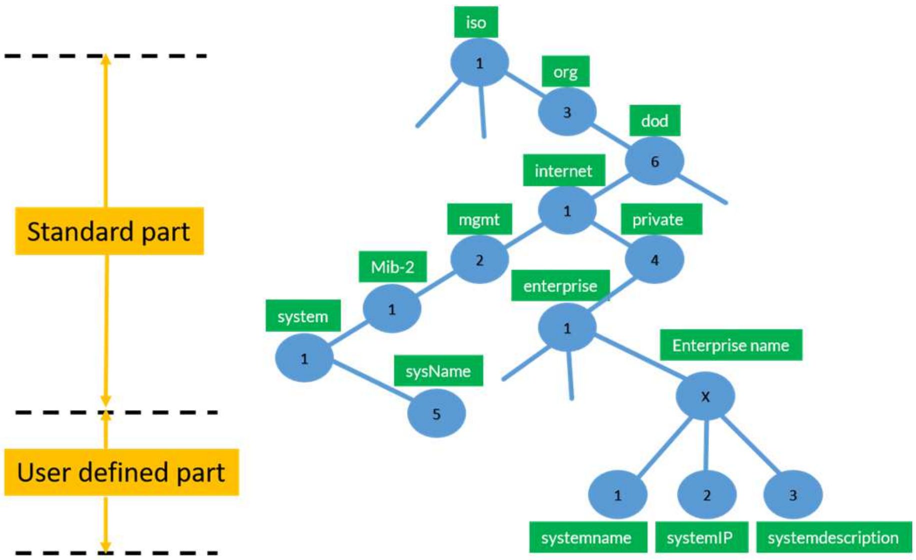
Response

GetBulkRequest

Response

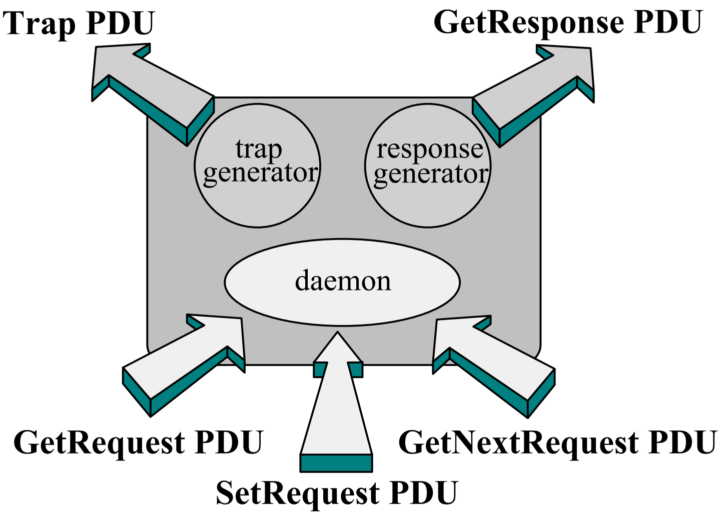
SNMP Agent
UDP Port 161

SNMP Mib


Standard part
User defined part
iso
1
org
3
dod
6
internet
1
mgmt
2
private
4
Mib-2
1
enterprise
1
system
1
sysName
5
Enterprise name
x
1
2
3
systemname
systemIP
systemdescription

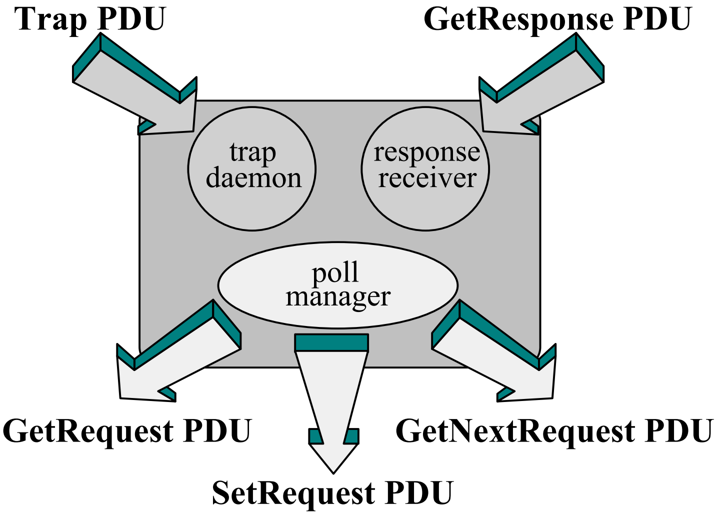
Agent SNMP

An agent SNMP

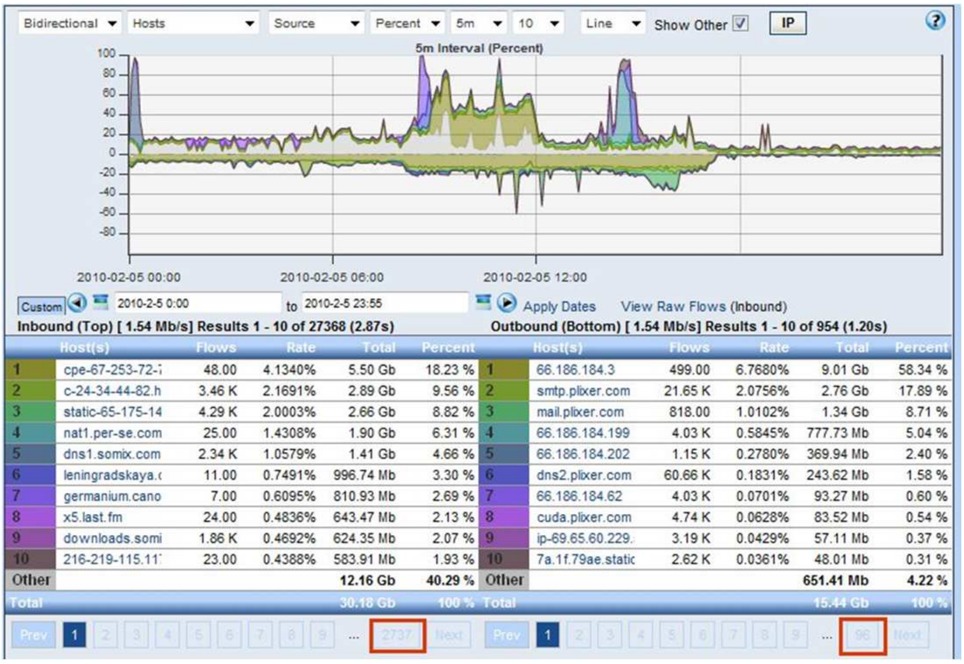


Manager SNMP

A manager SNMP



SNMP



2010-2-5 0:00 to 2010-2-5 23:55

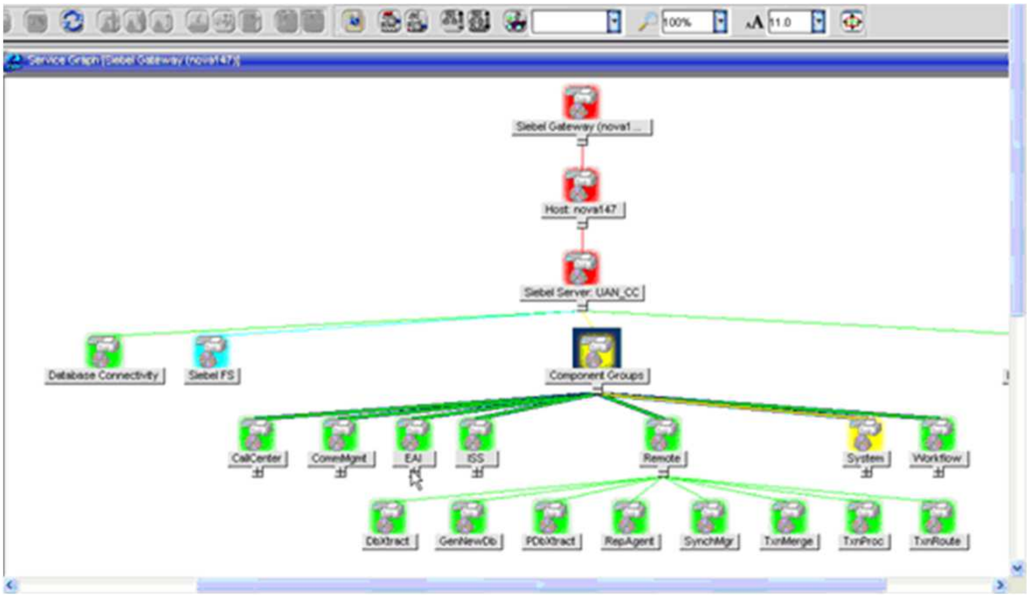
Inbound (Top) [1.54 Mb/s] Results 1 - 10 of 27368 (2.87s)

	Host(s)	Flows	Rate	Total	Percent
1	cpe-67-253-72-1	48.00	4.1340%	5.50 Gb	18.23 %
2	c-24-34-44-82.h	3.46 K	2.1691%	2.89 Gb	9.56 %
3	static-65-175-14	4.29 K	2.0003%	2.66 Gb	8.82 %
4	nat1.per-se.com	25.00	1.4308%	1.90 Gb	6.31 %
5	dns1.somix.com	2.34 K	1.0579%	1.41 Gb	4.66 %
6	leningradskaya.(	11.00	0.7491%	996.74 Mb	3.30 %
7	germanium.cano	7.00	0.6095%	810.93 Mb	2.69 %
8	x5.last.fm	24.00	0.4836%	643.47 Mb	2.13 %
9	downloads.somi	1.86 K	0.4692%	624.35 Mb	2.07 %
10	216-219-115.11	23.00	0.4388%	583.91 Mb	1.93 %
Other				**12.16 Gb**	**40.29 %**
Total				30.18 Gb	100 %

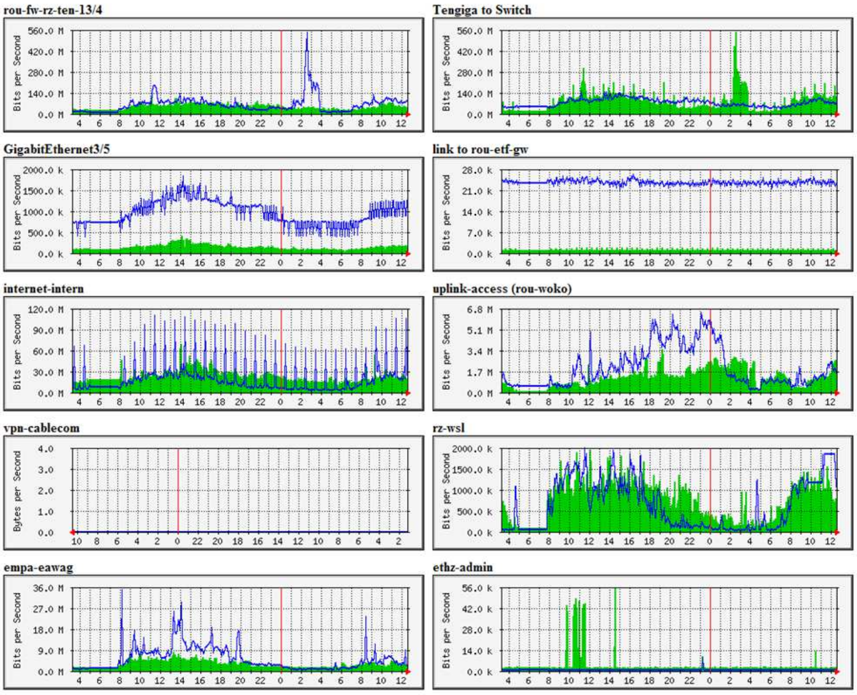
Outbound (Bottom) [1.54 Mb/s] Results 1 - 10 of 954 (1.20s)

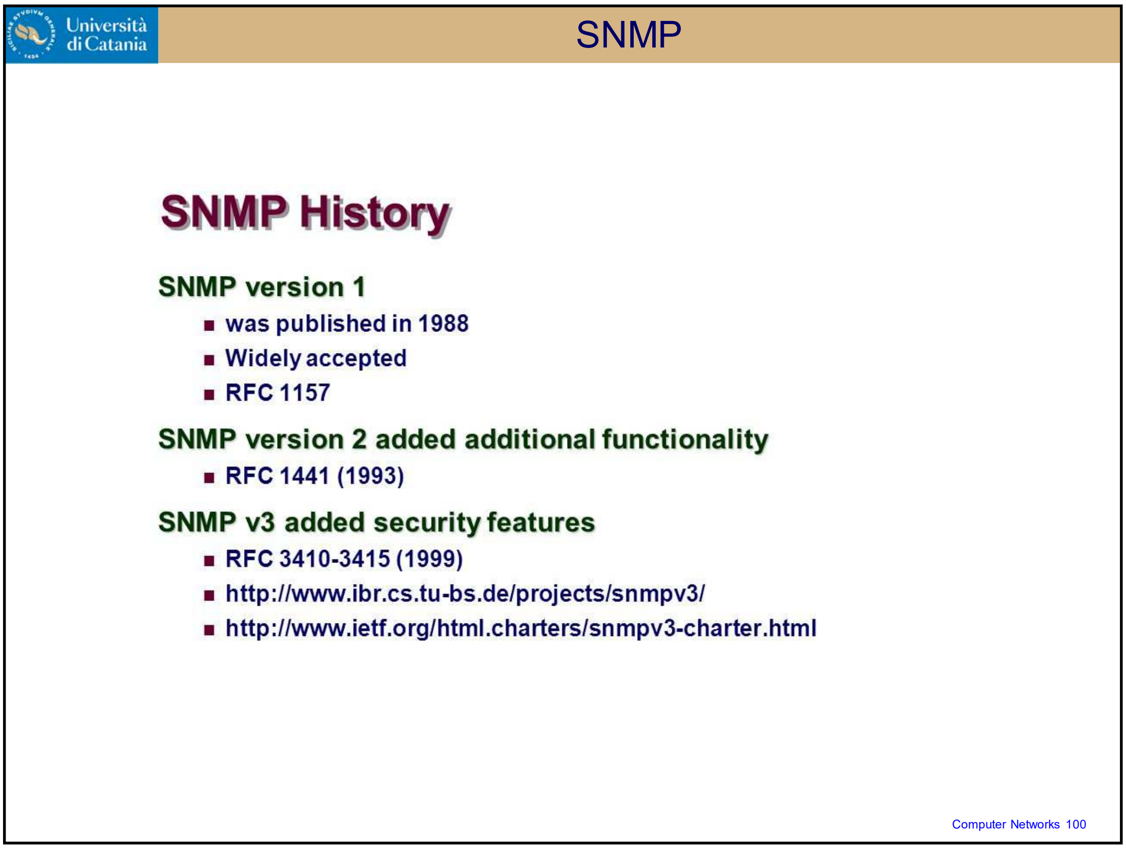
	Host(s)	Flows	Rate	Total	Percent
1	66.186.184.3	499.00	6.7680%	9.01 Gb	58.34 %
2	smtp.plixer.com	21.65 K	2.0756%	2.76 Gb	17.89 %
3	mail.plixer.com	818.00	1.0102%	1.34 Gb	8.71 %
4	66.186.184.199	4.03 K	0.5845%	777.73 Mb	5.04 %
5	66.186.184.202	1.15 K	0.2780%	369.94 Mb	2.40 %
6	dns2.plixer.com	60.66 K	0.1831%	243.62 Mb	1.58 %
7	66.186.184.62	4.03 K	0.0701%	93.27 Mb	0.60 %
8	cuda.plixer.com	4.74 K	0.0628%	83.52 Mb	0.54 %
9	ip-69.65.60.229.	3.19 K	0.0429%	57.11 Mb	0.37 %
10	7a.1f.79ae.static	2.62 K	0.0361%	48.01 Mb	0.31 %
Other				**651.41 Mb**	**4.22 %**
Total				15.44 Gb	100 %

SNMP


100%
11.0
Service Graph [Siebel Gateway (nova147)]
Siebel Gateway (nova1 ...
Host: nova147
Siebel Server: UAN_CC
Database Connectivity
Siebel FS
Component Groups
CallCenter
CommMgmt
EAI
ISS
Remote
System
Workflow
DbXtract
GenNewDb
PDbXtract
RepAgent
SynchMgr
TxnMerge
TxnProc
TxnRoute

MRTG


rou-fw-rz-ten-13/4
Tengiga to Switch
GigabitEthernet3/5
link to rou-etf-gw
internet-intern
uplink-access (rou-woko)
vpn-cablecom
rz-wsl
empa-eawag
ethz-admin
Bits per Second
Bytes per Second
560.0 M
420.0 M
280.0 M
140.0 M
0.0 M
2000.0 k
1500.0 k
1000.0 k
500.0 k
0.0 k
28.0 k
21.0 k
14.0 k
7.0 k
120.0 M
90.0 M
60.0 M
30.0 M
6.8 M
5.1 M
3.4 M
1.7 M
4.0
3.0
2.0
1.0
0.0
36.0 M
27.0 M
18.0 M
9.0 M
56.0 k
42.0 k
28.0 k
14.0 k
4 6 8 10 12 14 16 18 20 22 0 2 4 6 8 10 12
10 8 6 4 2 0 22 20 18 16 14 12 10 8 6 4 2



Università di Catania

SNMP

SNMP History

SNMP version 1

- was published in 1988
- Widely accepted
- RFC 1157

SNMP version 2 added additional functionality

- RFC 1441 (1993)

SNMP v3 added security features

- RFC 3410-3415 (1999)
- http://www.ibr.cs.tu-bs.de/projects/snmpv3/
- http://www.ietf.org/html.charters/snmpv3-charter.html