

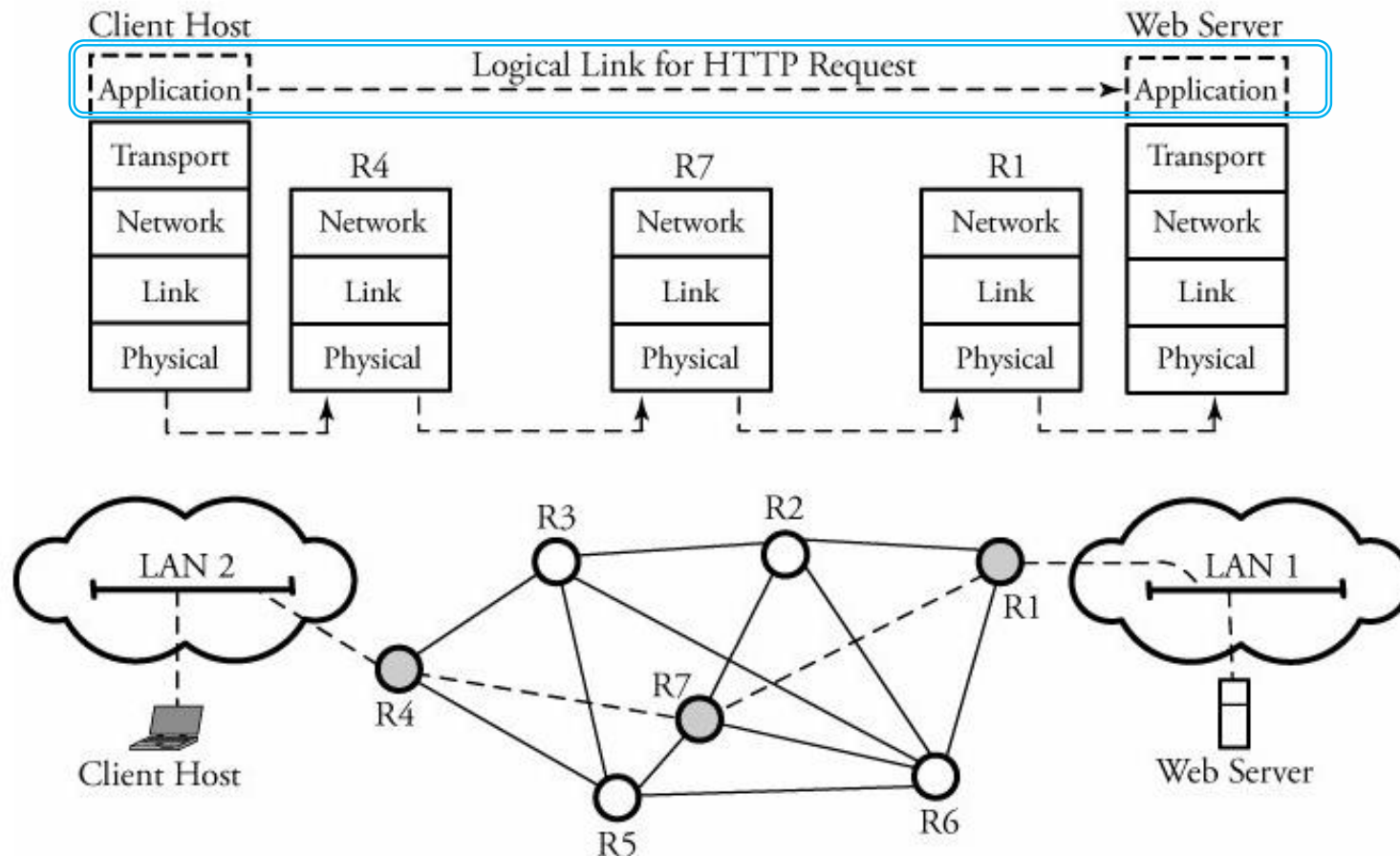
Application Level

Lenuta Alboaie (adria@info.uaic.ro)
Andrei Panu (andrei.panu@info.uaic.ro)

Content

- **Application Level protocols**
 - Preliminaries
 - Design Features
 - Access to the remote terminal
 - Electronic mail
 - SMTP (Simple Mail Transfer Protocol)
 - POP (Post Office Protocol)
 - File transfer
 - TFTP (Trivial File Transfer Protocol)
 - FTP (File Transfer Protocol)
 - World-Wide Web (HTTP)

Preliminaries



Communication between two *end-systems*

[Computer and Communication Networks , Nader F. Mir, 2006]

Preliminaries

| Application | | Application layer protocol | Underlying transport protocol |
|------------------------|----------------------|-------------------------------------|-------------------------------|
| remote terminal access | e-mail | SMTP [RFC 2821] | TCP |
| | Web | Telnet [RFC 854] | TCP |
| | file transfer | HTTP [RFC 2616] | TCP |
| | streaming multimedia | FTP [RFC 959] | TCP |
| | Internet telephony | HTTP (eg Youtube), RTP [RFC 1889] | TCP or UDP |
| | | SIP, RTP, proprietary (e.g., Skype) | typically UDP |

Preliminaries

- **At the application layer, many services are provided :**
 - Remote Terminal (TELNET, SSH, ...)
 - Mail (SMTP, IMAP, POP, ...)
 - File Transfer(TFTP, FTP, ...)
 - World-Wide Web (HTTP)
 - Instant conversations (ICQ, XMPP (from May 2014 -> no longer supported in Google Voice), Hangouts IM, WhatsApp, ...)
- **It offers the protocols to solve system tasks - /etc/services, /etc/protocols**
 - Network file system (NFS)
 - Connectivity with other file systems (SMB)
 - Database Services (MySQL, PostgreSQL, ..., Hive, ...)

Design Features

- **Types of protocols depending on the nature of the transferred data**
 - **Character streams generated by user**
 - Used for interactive applications (*telnet, rlogin, IRC, ...*)
 - The traffic is composed of uninterpretable data
 - It may include control sequences (e.g., terminal control, color codes) – ANSI codes
- (Example: **CSI *n* E** -> called: CNL – Cursor Next Line
Moves cursor to beginning of the *n*-th (default 1) following line)

Design Features

- **Types of protocols depending on the nature of the transferred data**
 - **Question/Answer ASCII messages**
 - The server and the client send character streams that can be read by human users (SMTP, FTP, HTTP/1.1, XMPP, SIP, ...)
 - Usually, consist of lines of text
 - **Binary formats**
 - Used for lower-level protocols (SNMP – Simple Network Management Protocol) or high level protocols (NFS over RPC, HTTP/2.0)
 - Problems in data representation may appear (e.g., byte order)
 - **Ad-hoc protocols used by applications (non-standard) written by users**
 - Can adopt some of the previous types

Design Features

- Requirements relating to the protocol design
 - Critical parameters: command length name, buffer size, addressing mode
 - Defining allowed operations (e.g., create, read, write, delete, update)
 - Reporting errors: error codes, messages
 - Message formats: source, destination, parameters, data encoding, fixed/variable length, ...

Design Features

- The usual scenario
 - Server – reads operation code (opcodes) and reports status using error codes
 - Client – constructs messages using the allowed opcodes

Design Features

- *Reliability* issue
 - The network may loss messages
 - Approaches:
 - *post-office*
 - Does not expect any confirmations
 - *Handshaking* – all messages are confirmed
 - *Acknowledged reply*
 - An answer is expected and the sender waits for a confirmation response
 - *Request/reply* – the sender waits for the response (for a period of time) (e.g., RPC, SOAP)

Access to the remote terminal

- An “ancient” standard Internet service
- Used through commands such as *rlogin*, *telnet*, *ssh* (secure version of *telnet*)
- Uses client/server model:
 - The client interacts with the user
 - The server provides access to a *shell* (e.g., *bash*)



*remote login
application*

Access to the remote terminal

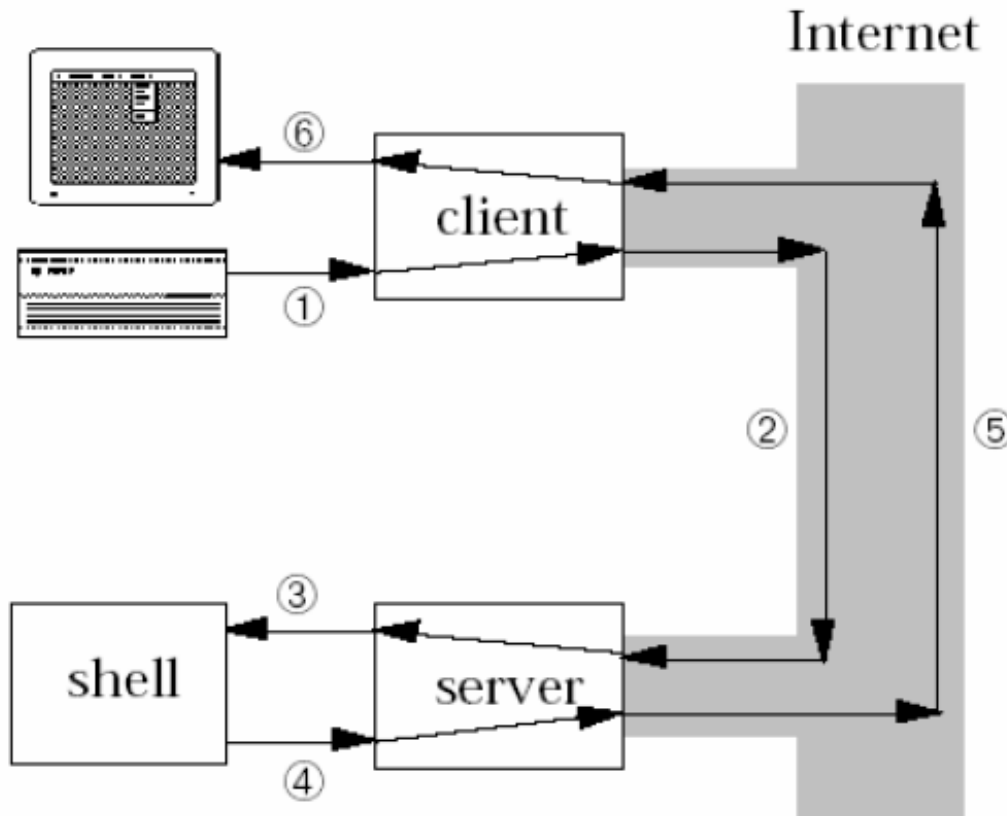


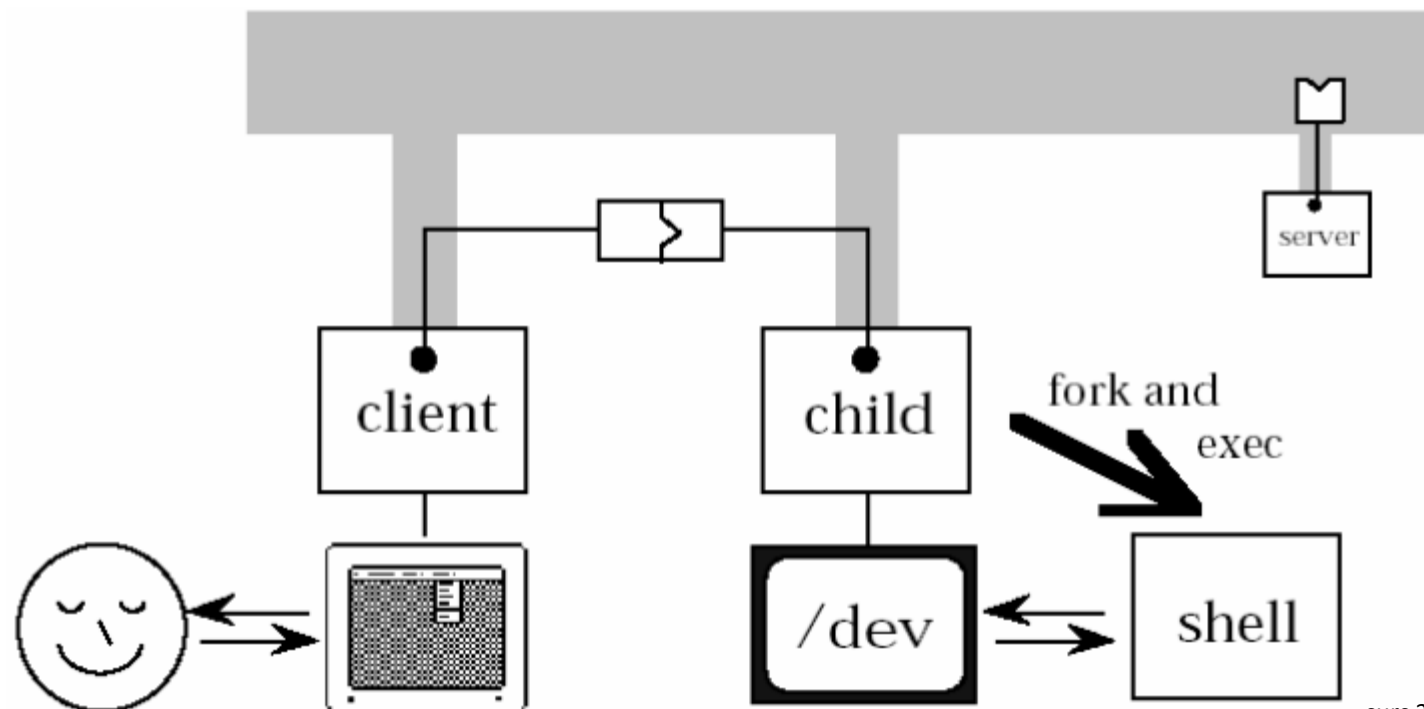
Figure: The functionality of a *remote login* application

Access to the remote terminal

- Implementation – general mechanism

For remote connection, each client will be serviced by a server child process

The child process will create another process that will connect the customer to a pseudo-terminal and execute the *shell*



[Retele de calculatoare –
curs 2007-2008, Sabin Buraga]

Access to the remote terminal

- **Issues**

- Initialization and authentication
 - How is the client identified?
 - How do we know that the server is official?
- Who processes tasks such as: line editing, display typed characters (*echoing*), suspending the terminal (CTRL + S), etc.?
- Communication between client and the server
 - Interruptions from the user
 - The control over the window size

Access to the remote terminal

- **rlogin**

- Simple protocol for remote access
- Used exclusively for connecting UNIX machines
- RFC 1258: *“The rlogin facility provides a remote-echoed, locally flow-controlled virtual terminal with proper flushing of output”*

Functionality:

- **rlogin** communicates with a **rlogind** daemon from the remote host
- Authentication is performed through the so called “reliable” hosts (*“trusted” hosts*)
 - rlogind allows logging without password if the remote host appears in the file */etc/hosts.equiv* or if the user has a *.rlogin* file in its *home* directory

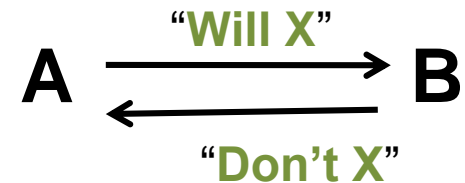
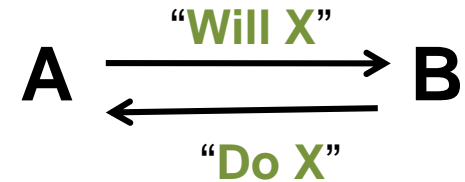
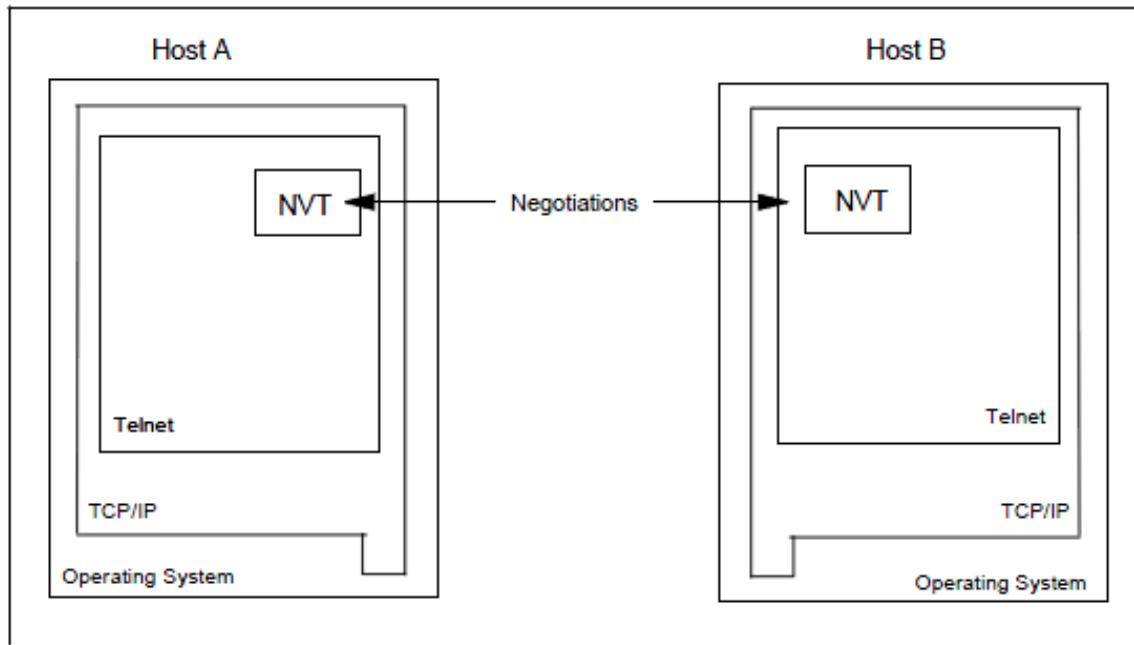
Access to the remote terminal

- **telnet** (*terminal network*)
 - TCP/IP protocol for remote access (RFC 854, 855)
 - It is platform independent
 - It can be used as a generic client without knowing details about the server
 - Client authentication is not performed by the protocol, but by the application
 - The protocol is based on:
 - **NVT** (**Network Virtual Terminal**) concept: a virtual device with a general structure compatible with a wide range of terminals; each host maps its own characteristics to those of NVT terminal;
 - Once a connection has been established through TELNET, both ends are treated symmetrically

Access to the remote terminal

- **telnet** (*terminal network*)

- The protocol is based on:
 - Both sides of communication can negotiate the use of additional options that reflect the used hardware
 - Options: line editing, window size, etc.



telnet offers compatibility with old terminals (vt52, vt100,...)

Access to the remote terminal

- **telnet** (*terminal network*)
 - Communication between client and server is carried out through commands such as:
 - **IP** (**I**nterrupt **P**rocess; 244) -> the running program ends
 - **AO** (**A**bort **O**utput; 245) -> release any output buffer
 - **AYT** (**A**re **y**ou **t**here; 246) -> allows the client to send an OOB probe to check if the remote end-point is alive
 - **EC** (**E**rase **c**haracter; 247) -> delete previous character
 - **EL** (**E**rase **L**ine; 248) -> delete the entire line
 - ... (RFC 854)
 - **Sending a command:** the command (1 byte) is preceded by a byte with value 255 - IAC (*Interpret As Command*)

SSH

- **SSH** (*secure shell*)
 - Provides a secure communication (TCP based) through encrypted messages and authentication messages
 - SSH uses the client/server model
 - A SSH client is used to establish a connection with a SSH *daemon*
 - Uses:
 - Logging into a remote machine and execute commands
 - Support for *tunneling* (future course)
 - Allows file transfer in combination with SFTP or SCP protocols
 - It has support in most modern operation systems

Electronic Mail (E-mail)

- **TCP based protocols:**
 - **SMTP** (*Simple Mail Transfer Protocol*)
 - RFC 821 (specifies how mail is exchanged between two hosts)
 - **POP** (*Post Office Protocol*)
 - RFC 1939
 - **POP3S** – secure version of **POP3**
 - See also: RFC 822 (specifications regarding mail header), RFC 2049 (specifications over documents different by *plain text ASCII* that may be contained in an email), RFC 974 (standard on mail routing using DNS)
 - RFC 822 and 974 -> consolidated in RFC 2821, 2822

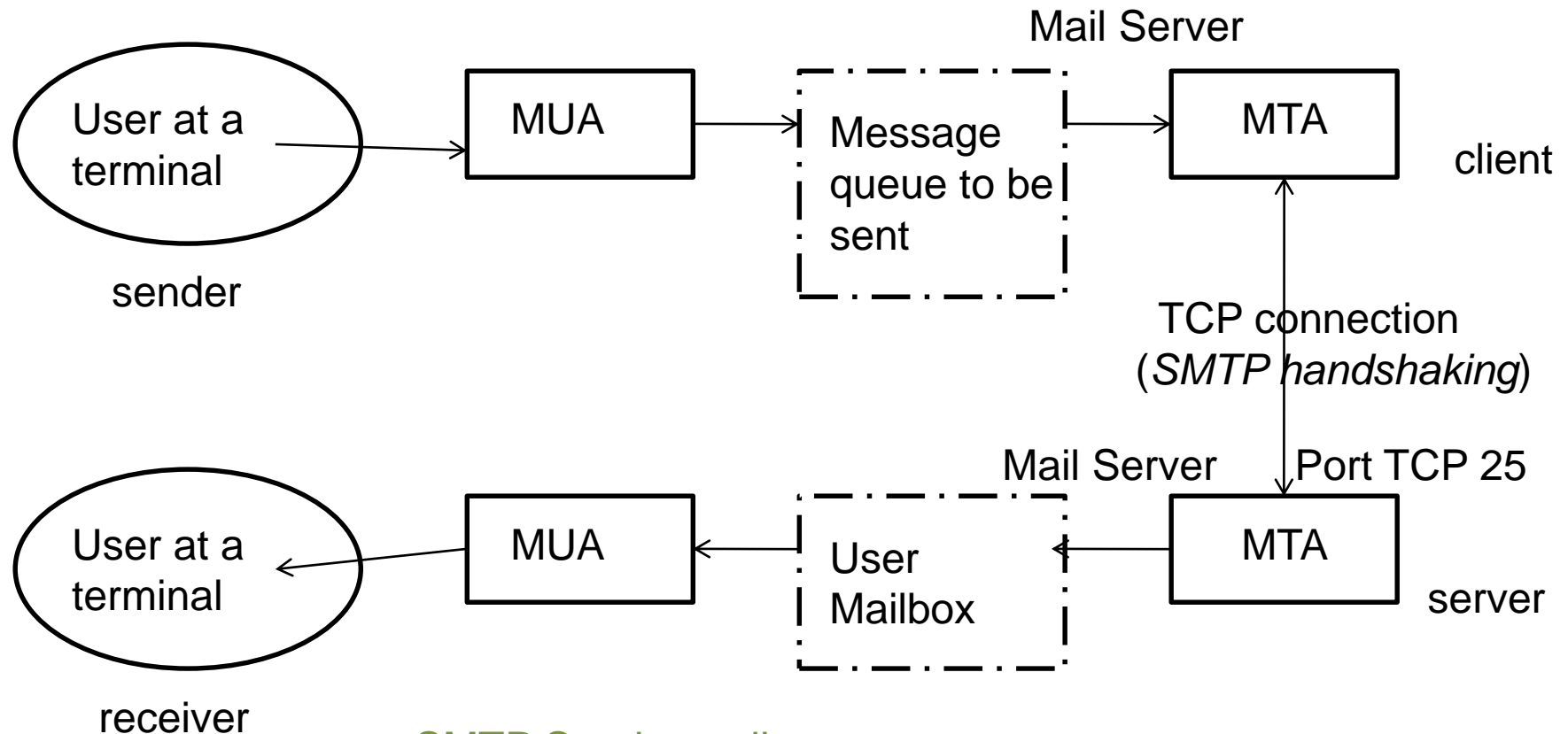
E-mail

- **Terminology**

- **MUA** – *Mail User Agent*: (local) client for mail
Ex: alpine, mutt, Mozilla Thunderbird, Kmail, Outlook etc.
- **MTA** – *Mail Transport Agent*
responsible for communicating with remote hosts and sending / receiving mail (client & server); Ex.: sendmail, qmail
- **MDA** - *Mail Distribution Agent* or LDA (*Local Delivery Agent*)
directs incoming messages to the user's mailbox; Ex: maildrop, Sieve, procmail
- **Mail exchanger (MX)** – responsible host for the e-mails of a domain

E-mail | SMTP

- Used in the exchange of mail messages between mail servers (MTAs)

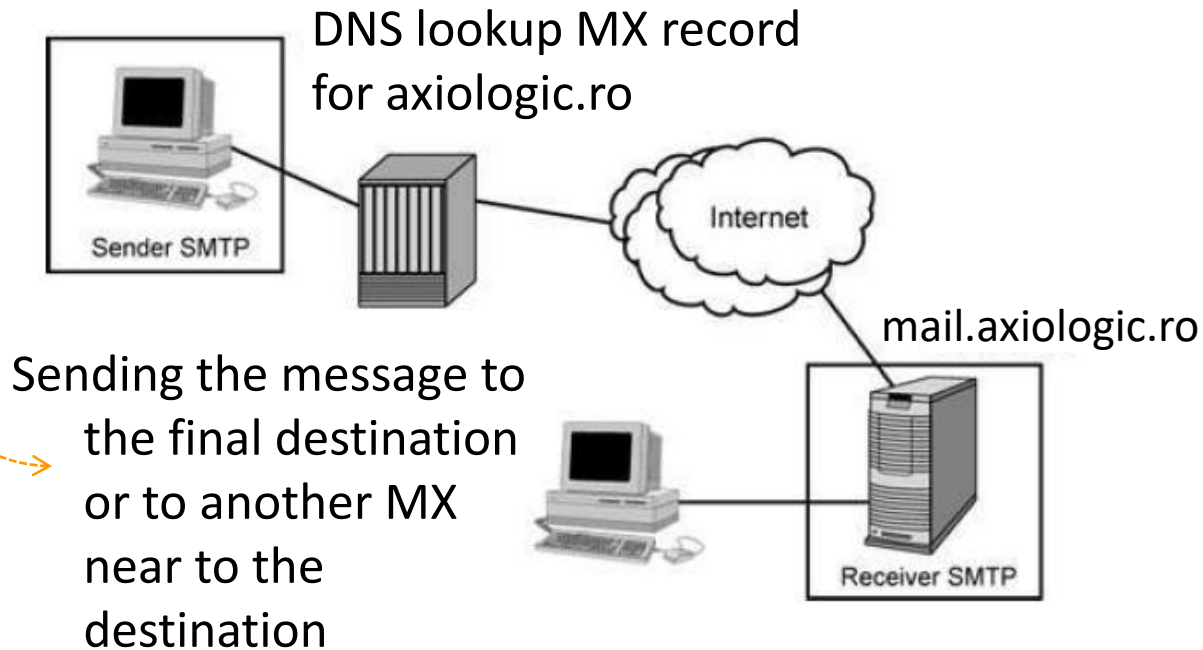


SMTP Sender = client
SMTP Receiver = server

E-mail | SMTP

- DNS and e-mail

- The MX resource-record from DNS identifies the host that processes and forwards mails for the specified domain



- General mechanism:

- The SMTP Server verifies the MX record of the domain specified in the email address (e.g., axiologic.ro for b@axiologic.ro address) and let's say that this record is mail.axiologic.ro.
- This mail will be send to the SMTP server on the mail.axiologic.ro machine.

E-mail

- **Characteristics**

- The distinction between **envelope** and **content**
 - **Envelope** encapsulates the message, contains data necessary to carry messages: recipient address, priority, security, ...
 - The envelope is used to route the message to the recipient
 - **The message** in the envelope contains a **header** (control data for MUA) and a **body** (data for user)
- Each user is identified by an e-mail address:
mailbox@location (**account@internetAddress**)

E-mail | SMTP

- **Components:**

- *envelope* – used by the MTA for delivery

Example:

MAIL From: <adria@info.uaic.ro>

RCPT to: <adria@info.uaic.ro>

- *headers* – used by MUA

Example: Received, Message-ID, From, Date, Reply-To,
Subject,...

- *message body*

- Mechanism: MUA takes the content, adds headers and forwards to the MTA; MTA adds header, adds envelope, and sends it to another MTA

E-mail | SMTP

- Header fields used in emails:

| Header | Meaning |
|--------------|---|
| To: | Email address(es) of primary recipient(s) |
| Cc: | Email address(es) of secondary recipient(s) |
| Bcc: | Email address(es) for blind carbon copies |
| From: | Person or people who created the message |
| Sender: | Email 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 |

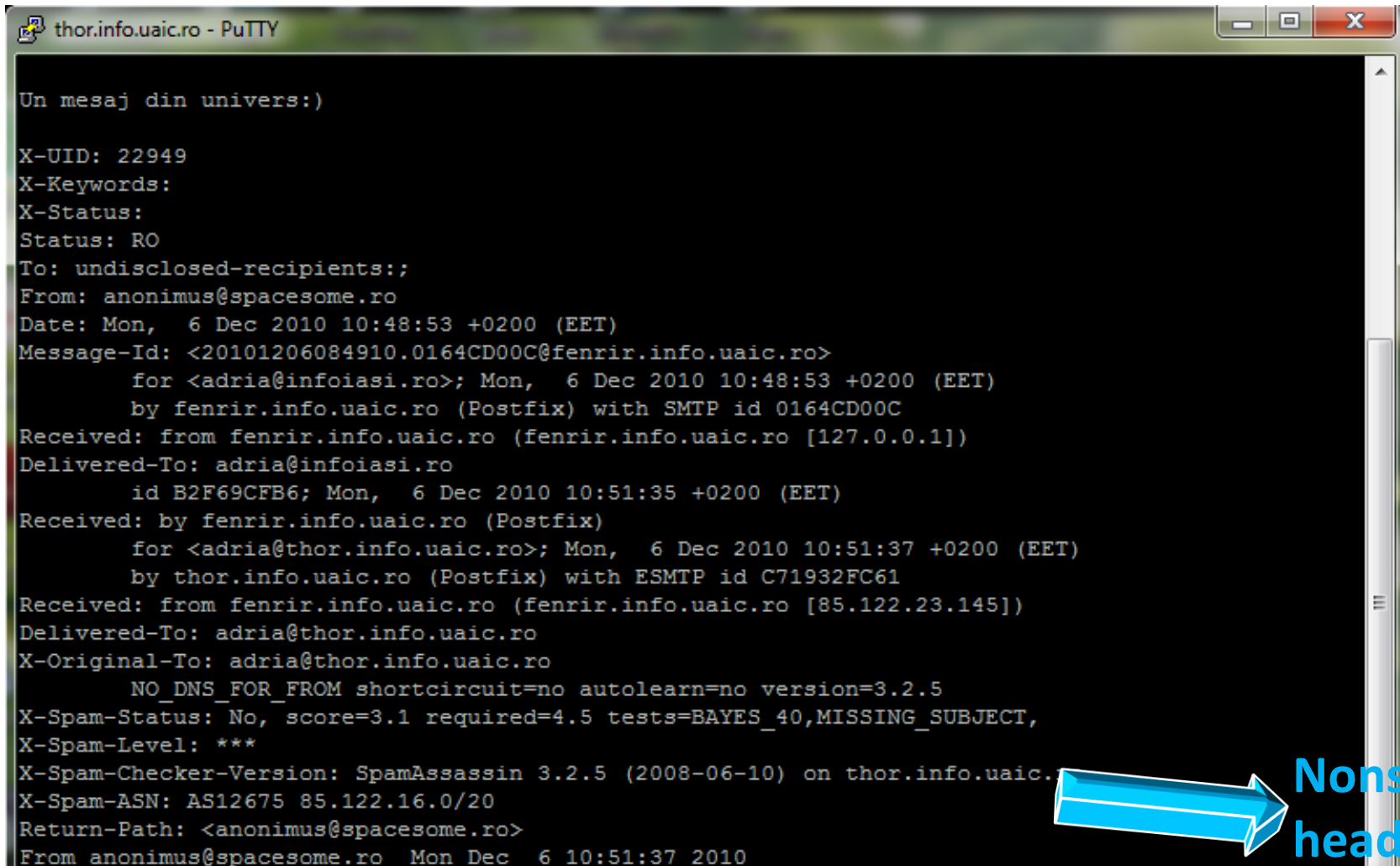
E-mail | SMTP

- Header fields used in emails:

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

E-mail | SMTP

Example



```
thor.info.uaic.ro - PuTTY

Un mesaj din univers:)

X-UID: 22949
X-Keywords:
X-Status:
Status: RO
To: undisclosed-recipients;;
From: anonimus@spacesome.ro
Date: Mon,  6 Dec 2010 10:48:53 +0200 (EET)
Message-Id: <20101206084910.0164CD00C@fenrir.info.uaic.ro>
    for <adria@infoiasi.ro>; Mon,  6 Dec 2010 10:48:53 +0200 (EET)
    by fenrir.info.uaic.ro (Postfix) with SMTP id 0164CD00C
Received: from fenrir.info.uaic.ro (fenrir.info.uaic.ro [127.0.0.1])
Delivered-To: adria@infoiasi.ro
    id B2F69CFB6; Mon,  6 Dec 2010 10:51:35 +0200 (EET)
Received: by fenrir.info.uaic.ro (Postfix)
    for <adria@thor.info.uaic.ro>; Mon,  6 Dec 2010 10:51:37 +0200 (EET)
    by thor.info.uaic.ro (Postfix) with ESMTP id C71932FC61
Received: from fenrir.info.uaic.ro (fenrir.info.uaic.ro [85.122.23.145])
Delivered-To: adria@thor.info.uaic.ro
X-Original-To: adria@thor.info.uaic.ro
    NO_DNS_FOR_FROM shortcircuit=no autolearn=no version=3.2.5
X-Spam-Status: No, score=3.1 required=4.5 tests=BAYES_40,MISSING_SUBJECT,
X-Spam-Level: ***
X-Spam-Checker-Version: SpamAssassin 3.2.5 (2008-06-10) on thor.info.uaic.
X-Spam-ASN: AS12675 85.122.16.0/20
Return-Path: <anonimus@spacesome.ro>
From anonimus@spacesome.ro  Mon Dec  6 10:51:37 2010
```

Nonstandard
headers

E-mail | SMTP

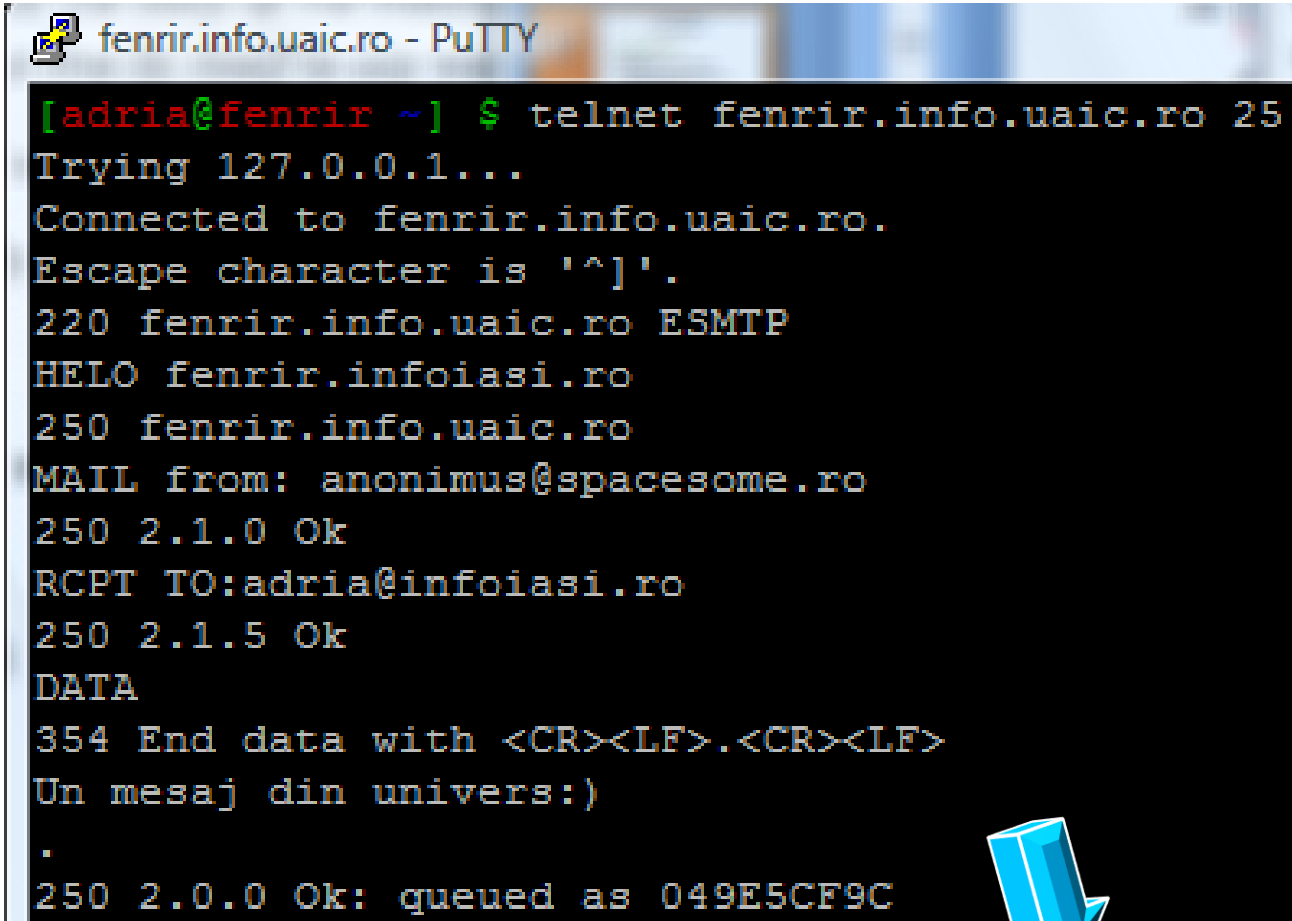
- Communication:
 - It creates a TCP connection between a *Sender SMTP* and a *Receiver SMTP* (between MTAs). Obs.: SMTP Receiver may be the final destination or an intermediate (*mail gateway*)
 - The client sends SMTP commands, and the server responds with status codes
 - Status messages include numeric codes NNN and texts
 - The commands' order is important
 - Port 25 is used

E-mail | SMTP

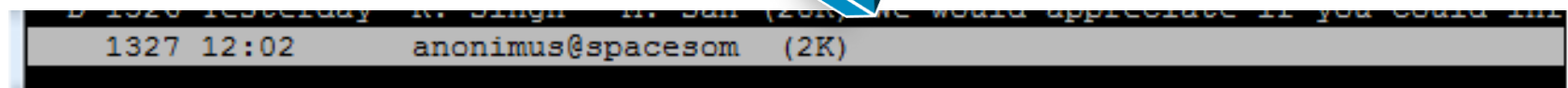
- Common commands
 - **HELO**: identifies the sending host
 - **MAIL FROM**: starts a transaction and identifies the e-mail origin
 - **RCPT TO**: identifies individual recipients of the message (e-mail); **RCPT TO**: multiple receivers can be specified
 - **DATA** represents text lines ended with \r\n; the last line contains only “.”
 - **QUIT**

E-mail | SMTP

- Example:



```
fenrir.info.uaic.ro - PuTTY
[adria@fenrir ~] $ telnet fenrir.info.uaic.ro 25
Trying 127.0.0.1...
Connected to fenrir.info.uaic.ro.
Escape character is '^]'.
220 fenrir.info.uaic.ro ESMTP
HELO fenrir.infoiasi.ro
250 fenrir.info.uaic.ro
MAIL from: anonimus@spacesome.ro
250 2.1.0 Ok
RCPT TO:adria@infoiasi.ro
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Un mesaj din univers:)
.
250 2.0.0 Ok: queued as 049E5CF9C
```



| | | | |
|------|-------|-------------------|------|
| 1327 | 12:02 | anonimus@spacesom | (2K) |
|------|-------|-------------------|------|

E-mail | SMTP

- **Other commands:**
 - **VRFY:** verifying the validity of a *recipient*
 - **NOOP:** forcing the server to respond with an OK code (200)
 - **TURN:** recipient and the sender are swapped without having to create a new TCP connection
 - **RSET:** drop the current transaction

E-mail | SMTP

- RFC 822: SMTP is limited to 7-bit ASCII text
- RFC 1521: defines a standard to solve previous limitations -> MIME (*Multipurpose Internet Mail Extensions*)
 - Encoding Standard for non-ASCII messages
 - Languages with accents, with non-Latin alphabets, without alphabet, non-textual messages
 - Allows to enclose any type of files to the email
 - The used field:

Content-Type: type/subtype

Example: Mime-Version: 1.0

Content-Type: TEXT/PLAIN

E-mail | SMTP

- MIME types:

application defines client application
(application/executable)

text defines text formats
(text/plain, text/html)

image specifies graphic formats
(image/gif, image/jpeg)

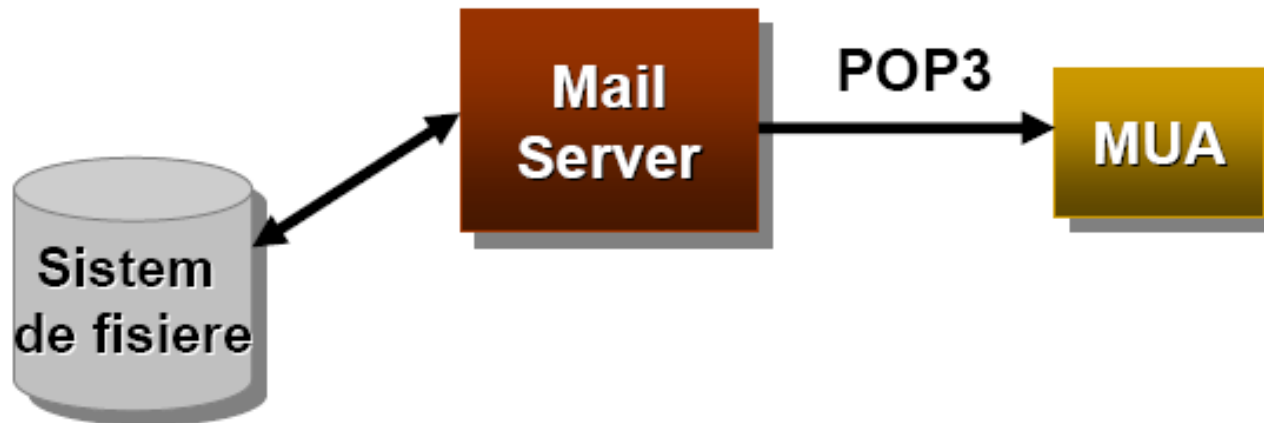
audio specifies audio formats (audio/basic)

video specifies video formats (video/mpeg)

multipart used for compound data transport
(multipart/mixed, multipart/alternative)

E-mail | POP

- POP (Post Office Protocol) – RFC 1939
- Used to transfer messages from a mail server to a MUA – 110 port
- Commands and responses are ASCII messages
- The responses begin with +OK or -ERR



[Retele de calculatoare –
curs 2007-2008, Sabin Buraga]

E-mail | POP

- Common commands:
 - **USER** specifies the account name
 - **PASS** specifies the password
 - **STAT** provides the number of messages from the *mailbox*
 - **LIST** displays the message list and the length one per line
 - **RETR** retrieves a message
 - **DELE** resets the transaction
 - **QUIT** deletes marked messages and closes the connection

```
adria@thor:~$ telnet thor 110
Trying 85.122.23.1...
Connected to thor.info.uaic.ro.
Escape character is '^]'.
+OK POP3 thor.info.uaic.ro 2007b.104 server ready
user adria
+OK User name accepted, password please
pass [REDACTED]
+OK Mailbox open, 1108 messages
stat
+OK 1108 194519602
retr 1108
+OK 1115 octets
Return-Path: <anonymus@spacesome.ro>
X-Spam-ASN: AS12675 85.122.16.0/20
X-Spam-Checker-Version: SpamAssassin 3.2.5 (2008-06-10) on thor.info.uaic.ro
X-Spam-Level: ***
X-Spam-Status: No, score=3.1 required=4.5 tests=BAYES_40,MISSING_SUBJECT,
NO_DNS_FOR_FROM shortcircuit=no autolearn=no version=3.2.5
X-Original-To: adria@thor.info.uaic.ro
Delivered-To: adria@thor.info.uaic.ro
Received: from fenrir.info.uaic.ro (fenrir.info.uaic.ro [85.122.23.145])
    by thor.info.uaic.ro (Postfix) with ESMTP id C71932FC61
    for <adria@thor.info.uaic.ro>; Mon,  6 Dec 2010 10:51:37 +0200 (EET)
Received: by fenrir.info.uaic.ro (Postfix)
    id B2F69CFB6; Mon,  6 Dec 2010 10:51:35 +0200 (EET)
Delivered-To: adria@infoiasi.ro
Received: from fenrir.info.uaic.ro (fenrir.info.uaic.ro [127.0.0.1])
    by fenrir.info.uaic.ro (Postfix) with SMTP id 0164CD00C
    for <adria@infoiasi.ro>; Mon,  6 Dec 2010 10:48:53 +0200 (EET)
Message-Id: <20101206084910.0164CD00C@fenrir.info.uaic.ro>
Date: Mon,  6 Dec 2010 10:48:53 +0200 (EET)
From: anonymus@spacesome.ro
To: undisclosed-recipients:;
Status: O

Un mesaj din univers:)
```

E-mail | POP

Exemplu



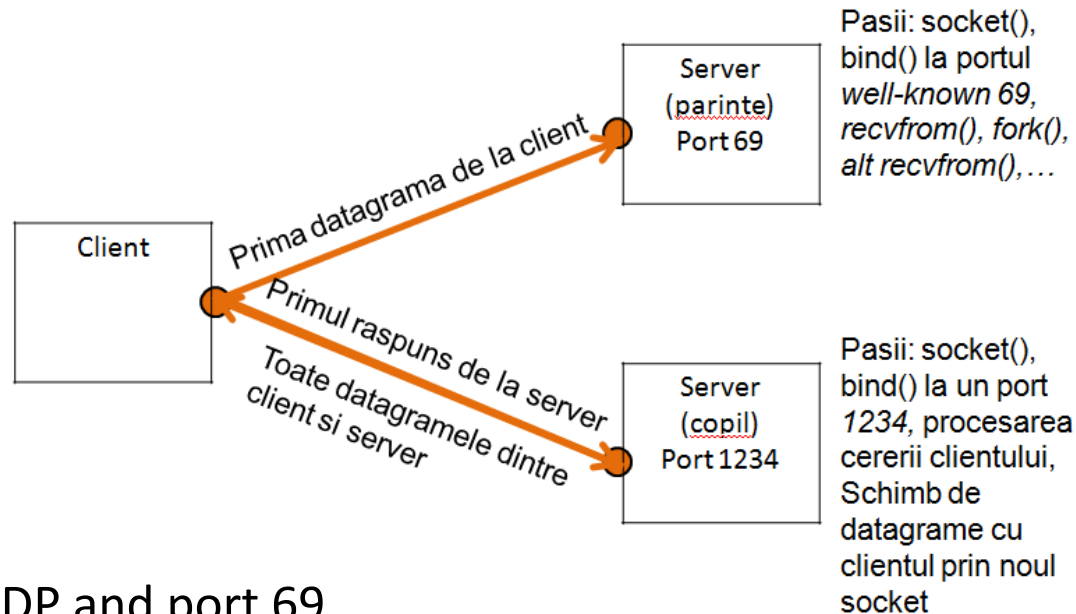
unencrypted
password

E-mail | POP

- POP 3 - features:
 - Generally, if the user changes the client he can no longer access the emails; obs.: Some clients have a *'keep a copy of the email on the server'* option
 - Uses the mechanism *"download-and-keep"*: copies of the messages on different clients
 - POP3 is stateless between sessions
- Other solutions:
 - IMAP (interactive Mail Access Protocol) – RFC 1730
 - Keeps all messages in one place: the server
 - Allows the user to organize messages in folders
 - Stores user status between sessions
 - The folder names and mapping with the messages IDs

File Transfer | TFTP

- TFTP (Trivial File Transfer Protocol) -> ...Course 6 & RFC 1350



- uses UDP and port 69
- often used to initialize diskless workstations and other devices
- does not have authentication and encryption mechanisms => used in local networks
- RFC 1785, 2347, 2348, 2349

File Transfer | TFTP

- TFTP (Trivial File Transfer Protocol)

TFTP implementations use commands such as:

| | |
|---|--|
| <code>Connect <host></code> | Specifies the destination host ID. |
| <code>Mode <ascii binary></code> | Specifies the type of transfer mode. |
| <code>Get <remote filename> [<local filename>]</code> | Retrieves a file. |
| <code>Put <remote filename> [<local filename>]</code> | Stores a file. |
| <code>Verbose</code> | Toggles verbose mode, which displays additional information during file transfer, on or off. |
| <code>Quit</code> | Exits TFTP. |

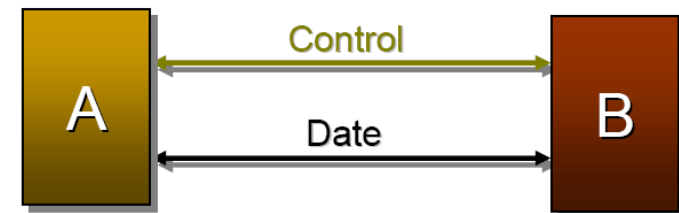
– RFC 1785, 2347, 2348, 2349

[TCP/IP Tutorial and Technical
Overview, IBM, 2006]

File Transfer | FTP

FTP – characterization

- Used both interactive and by programs
- Ensures safe and efficient file transfer
- It is based on the client/server model
- FTP uses two TCP connections to transfer files:
 - **Control Connection**
 - used to send commands and receive status codes
 - The control connection uses port 21
 - **Data connection**
 - Used for the actual transfer
 - The data connection uses port 20 or a random one ($P > 1023$)
 - It is not required in a FTP session



File Transfer | FTP

FTP – characterization

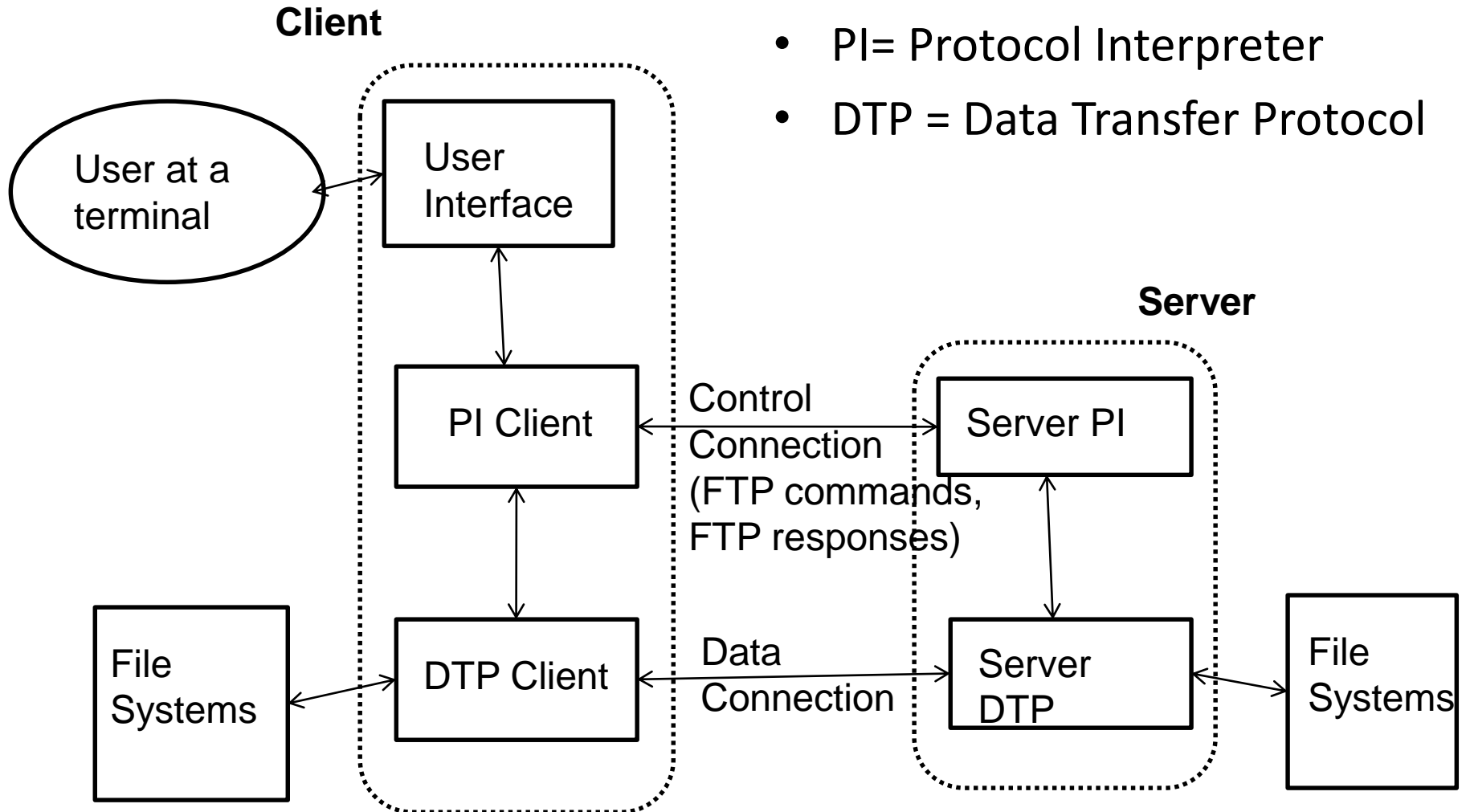
- Commands and responses are text lines
- Obs.: (FTP->) file transfer != file access (->NFS)
- See RFC 956, 1068, 2228 (FTP Security Extensions), 2428 (FTP Extensions for IPv6 and NATs)
- TELNET protocol can be used as an interactive client

Access types:

- **FTP anonymous** – RFC 1635
 - Authentication is performed using *anonymous* name and as password an e-mail address
 - Public access to a range of resources (applications, data, multimedia, etc.)
- **Authenticated**
 - Requires an existent username, accompanied by a valid password
 - For data transfer to / from the personal account

File Transfer | FTP

FTP- model



File Transfer | FTP

FTP – commands

```
adria@thor:~$ ftp
ftp> help
Commands may be abbreviated.  Commands are:

!          debug          mdir         qc            send
$          dir            mget         sendport      site
account   disconnect     mkdir        put           size
append    exit             mls          pwd           status
ascii     form             mode         quit          struct
bell      get              modtime      quote         system
binary    glob            mput        recv          sunique
bye       hash           newer        reget         tenex
case      help           nmap        rstatus       tick
cd        idle          nlist       rhelp         trace
cdup      image          ntrans      rename        type
chmod     lcd            open        reset         user
close     ls             prompt      restart       umask
cr        macdef         passive     rmdir         verbose
delete    mdelete        proxy       runique       ?
ftp>
```

File Transfer | FTP

FTP – common commands

| Command | Meaning |
|---------|--|
| Open | Create an FTP connection between the two hosts. |
| Close | Close an FTP connection between two hosts. |
| Bye | End the FTP session. |
| Get | Retrieve a remote file from the remote host. |
| Put | Store a file on the remote host. |
| Mget | Get multiple files using wildcards (for example, mget a* fetches all files that begin with the letter "a" in the current directory). |
| Mput | Put multiple files on the remote host using wildcards. |
| Glob | Enable wildcard interpretation. This is usually on by default. |
| Ascii | The file transferred is in ASCII representation (a common default). |
| Binary | The file is in image (binary) format (sometimes the default), and is useful for programs and formatted word processing files. |
| Cd | Change the directory on the remote host. |
| Dir | Get a directory listing from the remote host. |
| Ldir | Get a directory listing from the local host. |
| Hash | Display hash marks (dots) to show file transfer progress. |

→ **RETR** (*retrive*)
→ **STOR** (*store*)

File Transfer | FTP

FTP – commands

- Access control
 - **USER** *username*, **PASS** *password*, **QUIT**, **ChangeWorkingDir**,...
- Parameters transfer
 - **PORT**, **TYPE**, **MODE**
- Others
 - **RETR** *filename*, **ABOR**, **STOR** *filename*, **LIST**, **PrintWorkingDir**

The response status

Line of text containing: **XYZ status code** (used by software) + **explanatory text** (for people)

File Transfer | FTP

FTP – status code (xyz)

The first digit means:

- 1** one positive preliminary reply (“I do, but wait”)
- 2** final positive reply (“success”)
- 3** intermediate positive reply (“I need other information”)
- 4** transient negative reply (“error, try again”)
- 5** final negative reply (“fatal error”)

File Transfer | FTP

FTP – status code (xyz)

The second digit specifies groups of functions:

- 0 syntax errors
- 1 information (help, state information)
- 2 referring to connections
- 3 referring to the user authentication
- 4 not specified
- 5 referring to the file system

File Transfer | FTP

FTP – status code (xyz)

The third digit gives additional information on the error messages

Example:

125 Open connection; transfer on

200 OK

226 Transfer complete

331 Username OK, password required

452 Error on writing file

500 Syntax error (command unknown)

501 syntax error (invalid arguments)

221 Goodbye /*result of the QUIT command*/

File Transfer | FTP

FTP – Transfer Modes

- **STREAM**

- The file is sent as a stream of bytes; the end of the transmission is indicated by the normal connection closing;

- **BLOCK**

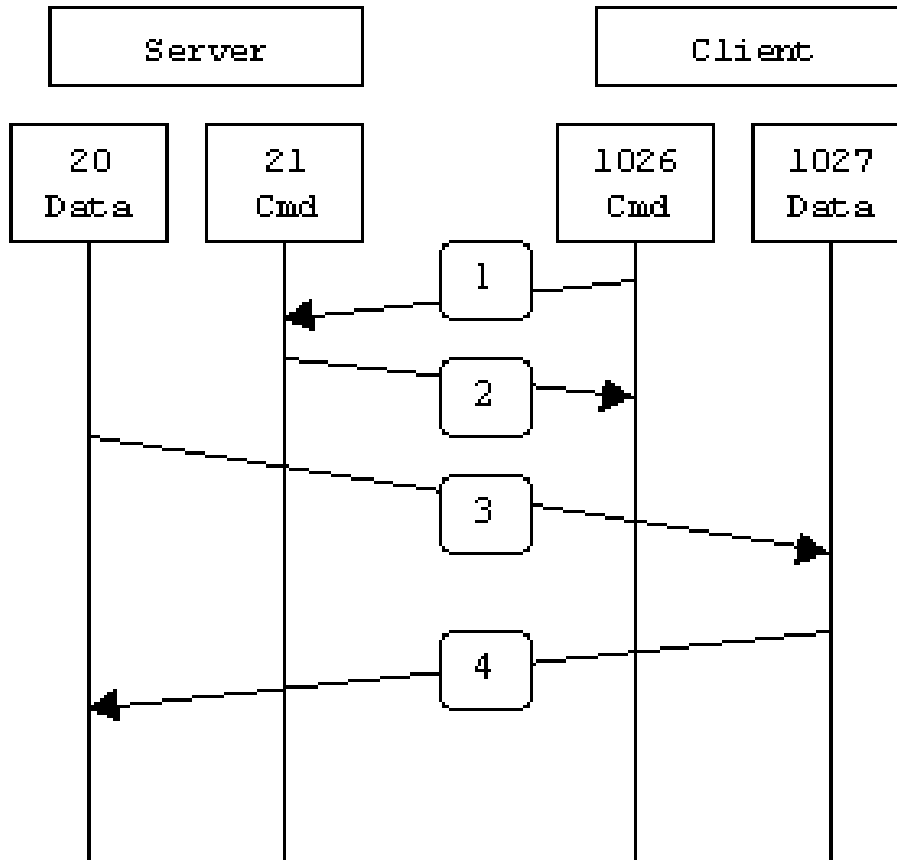
- The file is transmitted as a series of data blocks preceded by headers containing counters and block descriptors (e.g., end of data block)

- **COMPRESSED**

- The files are compressed according to a compression algorithm (e.g., gzip) and are sent as binary data

File Transfer | FTP

Active FTP – example

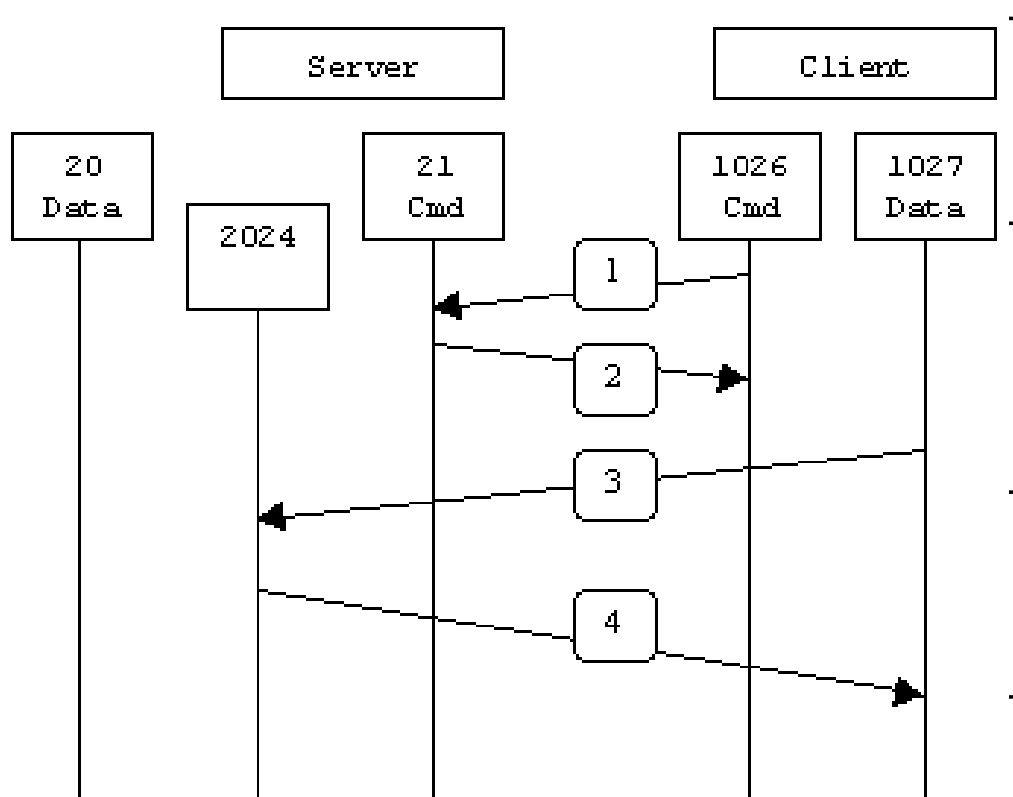


- The client connects to a server (85.122.23.145:**21**) from a port $P > 1023$
- The client sends the command PORT 85.122.23.1.**4.2** ($4 \cdot 256 + 2 = \mathbf{1026}$) which indicates to the Server to initiate a connection with the client to port $P+1$
- The client listens to $P+1$ and receives data sent through port 20

Obs.: The connection initiated by the server can be interpreted as a potential attack by the client's firewall

File Transfer | FTP

Passive FTP – example



- When initiating a FTP connection, the client uses two ports ($P > 1023$ and $P+1$)

- The client connects to a server (85.122.23.145:21) from the port P and sends PASV command

- The server opens a port $P_s > 1023$ and sends the command PORT P_s to the client

- The client will initiate a connection (from the $P+1$ port) with the server using the received port (P_s)

HTTP

- **Hyper Text Transfer Protocol**

- Protocol used in Internet based on TCP/IP
- Underpinning communication between Web servers and clients
 - Client: usually can be a browser
 - Server: Web server that sends responses to received requests
- HTTP 1.0 - RFC 1945
- HTTP 1.1 - RFC 2616
- HTTP 1.1 revised - RFC 723X (<https://www.w3.org/Protocols/>)

HTTPS Protocol – ensures “safe” HTTP communication via TLS (Transport Layer Security):

- authentication based on digital certificates + bidirectional encryption
- RFC 2818 – <https://tools.ietf.org/html/rfc2818>

HTTP

- SPDY Protocol – a Google experiment, available as an Internet Draft; Google gave up in 2016
 - Reducing latency and increasing load security
 - <https://www.chromium.org/spdy>
 - SPDY implementations exist in: Chrome, Mozilla Firefox, Opera, Amazon Silk, Internet Explorer
- **HTTP/2.0 Protocol**
 - RFC 7540
 - Extends SPDY, focused on performance
 - www.slideshare.net/mnot/what-http20-will-do-for-you

HTTP

- **Hyper Text Transfer Protocol**

General mechanism:

The client initiates a connection to the server using TCP port 80

The server accepts TCP connection

HTTP messages are exchanged between HTTP client (browser) and the Web server

TCP connection closing

HTTP

- HTTP – does not handle the routing or the check of requests
 - ? Who: TCP&IP
 - HTTP works with requests at a high level: *Fetch IndexPage* <https://www.google.com>
 - *Live HTTP Headers* (Firefox) ->

http://127.0.0.1:8080/Curs_ProgramareJS/test.html

GET /Curs_ProgramareJS/test.html HTTP/1.1

Host: 127.0.0.1:8080

User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:23.0) Gecko/20100101 Firefox/23.0

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Connection: keep-alive

HTTP/1.1 200 OK

Date: Thu, 05 Sep 2013 15:57:00 GMT

Server: Apache/2.4.3 (Win32) OpenSSL/1.0.1c PHP/5.4.7

Last-Modified: Thu, 05 Sep 2013 15:51:41 GMT

Etag: "95-4e5a4e5efb9d1"

Accept-Ranges: bytes

Content-Length: 149

Keep-Alive: timeout=5, max=100

Connection: Keep-Alive

Content-Type: text/html

Details about port 8080: https://www.grc.com/port_8080.htm

http://127.0.0.1:8080/Curs_ProgramareJS/imgsrc2.jpg

GET /Curs_ProgramareJS/imgsrc2.jpg HTTP/1.1

Host: 127.0.0.1:8080

User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64; rv:23.0) Gecko/20100101 Firefox/23.0

Accept: image/png,image/*;q=0.8,*/*;q=0.5

Accept-Language: en-US,en;q=0.5

Accept-Encoding: gzip, deflate

Referer: http://127.0.0.1:8080/Curs_ProgramareJS/test.html

Connection: keep-alive

HTTP/1.1 200 OK

Date: Thu, 05 Sep 2013 15:57:00 GMT

Server: Apache/2.4.3 (Win32) OpenSSL/1.0.1c PHP/5.4.7

Last-Modified: Wed, 28 Aug 2013 06:23:31 GMT

Etag: "3c9e-4e4fc0742da40"

Accept-Ranges: bytes

Content-Length: 15518

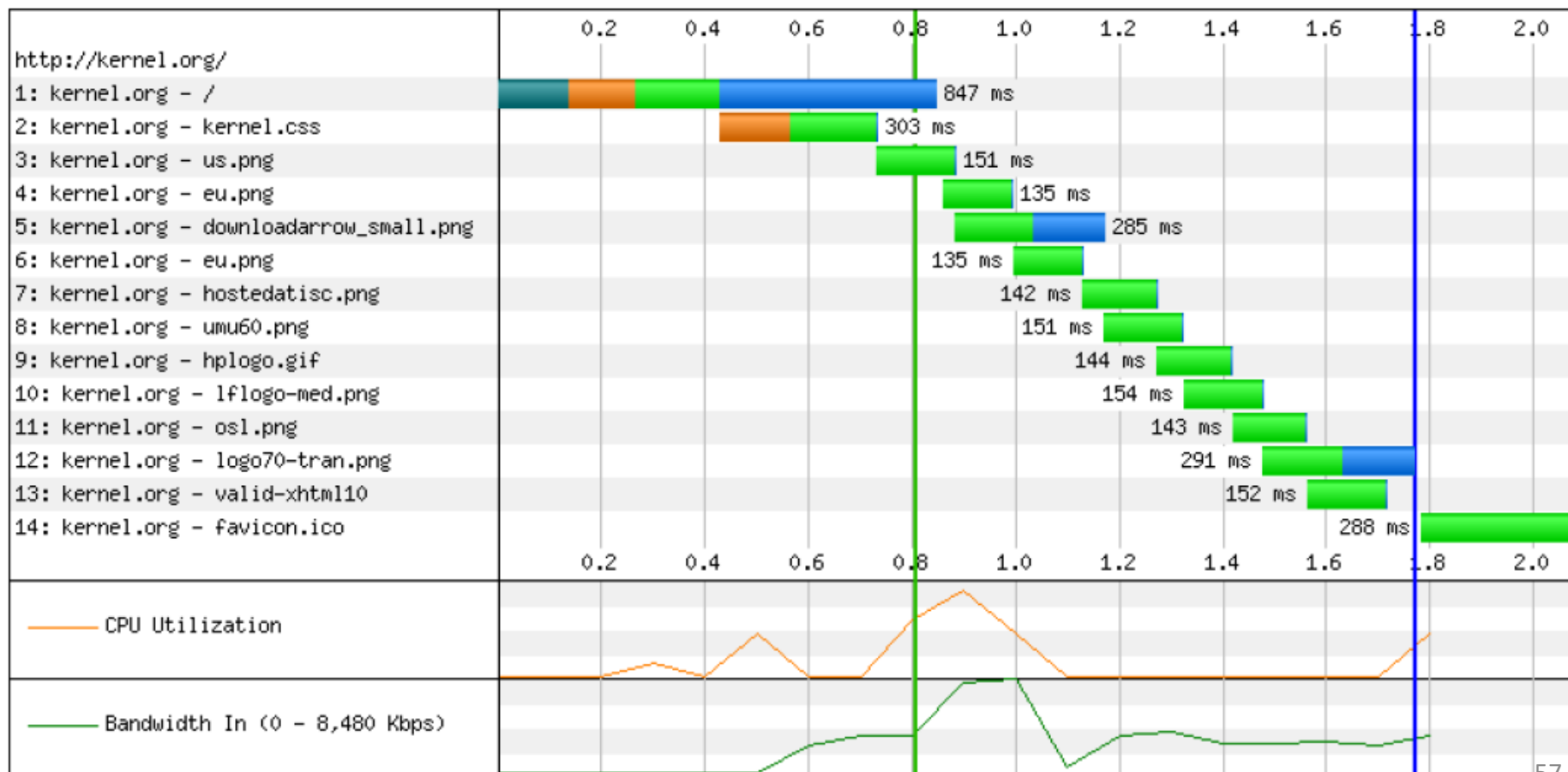
Keep-Alive: timeout=5, max=99

Connection: Keep-Alive

Content-Type: image/jpeg

Web Browser | Download

Example: access <http://kernel.org> resource



Web Browser | Download

- Steps taken by the browser:
 - Solving *kernel.org* using DNS in order to find out the IP (first segment)
 - The second segment indicates the attempt to create a new TCP connection to *kernel.org*
 - At the beginning of the third segment, the TCP connection was created and the browser can receive the answer; in our case, due to the server latency, in the four segment, the web server sends the content
 - Total: 847 milliseconds (ms) – and HTML document has been obtained
- Obs. Generally, web pages consist of links to stylesheets, images, JavaScript, etc.
 - As soon as the HTML document appears, the browser starts a fetch operation on a different resource (*kernel.css* in our case)
 - Obs. This time, there are no delays due to DNS lookup, because previous response has been placed in the browser's cache;
 - There are delays because of the initiation of the TCP connection with the server

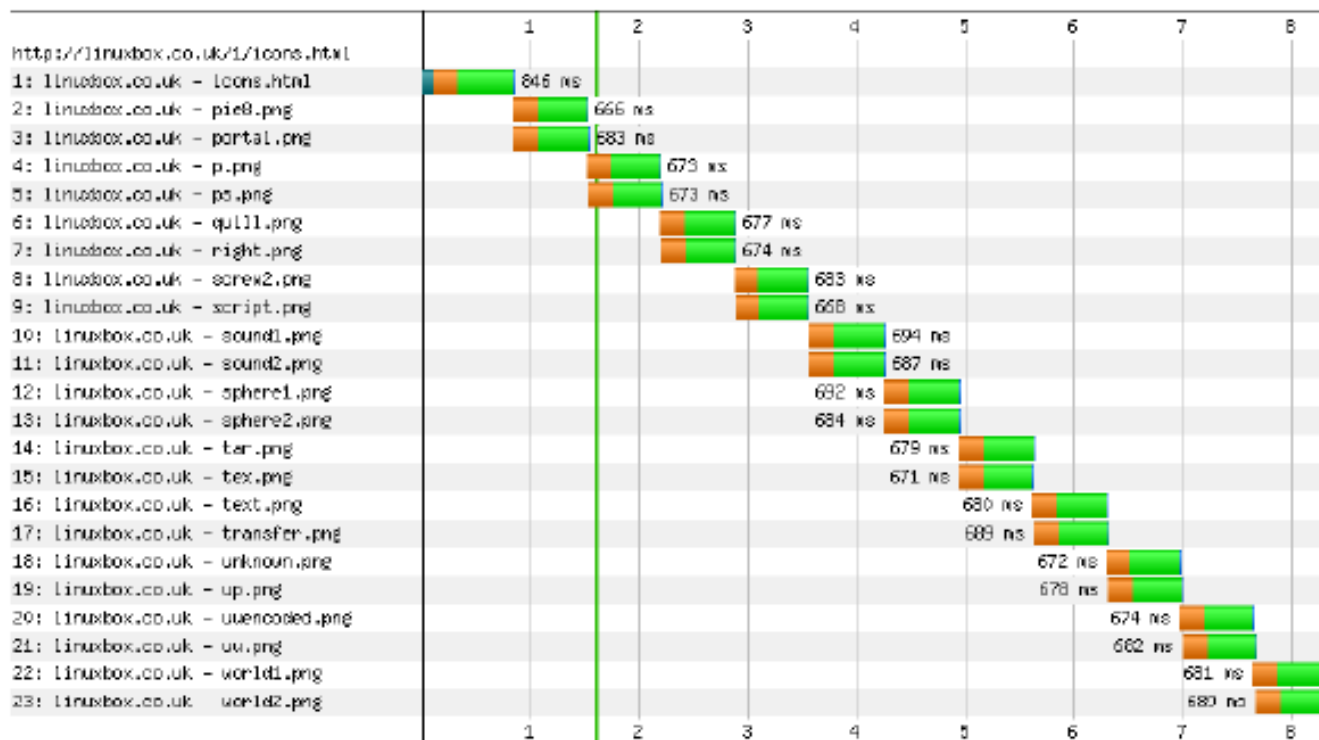
Web Browser | Download

- ? Why is there such high demand delay to *us.png*?
 - "Historical reasons": in our example, the browser downloads just two resources in parallel from one host (if *kernel.css* is finished then *us.png* starts)
 - For *us.png* resource and the subsequent, there is no TCP segment \Leftrightarrow the browser reuses the existing TCP connection => optimization (saves 0.1 seconds per request)
 - Obs. For 1, 5, and 12 resources, the time to download is half from all *fetching* process of the resource; to the remaining resources the download time is insignificant

Web Browser | Download

Persistent connections and *Keep-Alive*

- In HTTP 1.0 – the default behavior was that after obtaining each resource, the connection is closed
- Effect => latency in receiving the answer, resource utilization (CPU, RAM) on the client and server level



Web Browser | Download

Persistent connections and *Keep-Alive*

- The problem was partly solved by the introduction of *Keep-Alive*
 - The clients insert in the request the field: **Content: *Keep-Alive***
 - If the server supports this, it sends back a header with the same value
 - => Connection remains open until one of the parties decides to close them
 - ? But if a client closes the connection?
 - The server is idle and consumes memory
 - Most web servers implement a ***Keep-Alive timeout***
 - Also servers can limit the number of resources that are required per connection
- Keep-Alive: timeout = 5, max 100
- Obs.: Keep-Alive was not officially recognized and was not supported by all clients

Web Browser | Download

Persistent connections and *Keep-Alive*

- HTTP/1.1 formalized Keep-Alive => persistent connections by default
- If a client | server does not wish it, it may use a header field
Connection: close

Parallel Download

- RFC 2616: *“Clients that use persistent connections should limit the number of simultaneous connections that they maintain to a given server. A single-user client should not maintain more than 2 connections with any server or proxy.... These guidelines are intended to improve HTTP response times and avoid congestion.”*
- The aim of browser suppliers: increasing the user interaction



The load is the web servers' problem

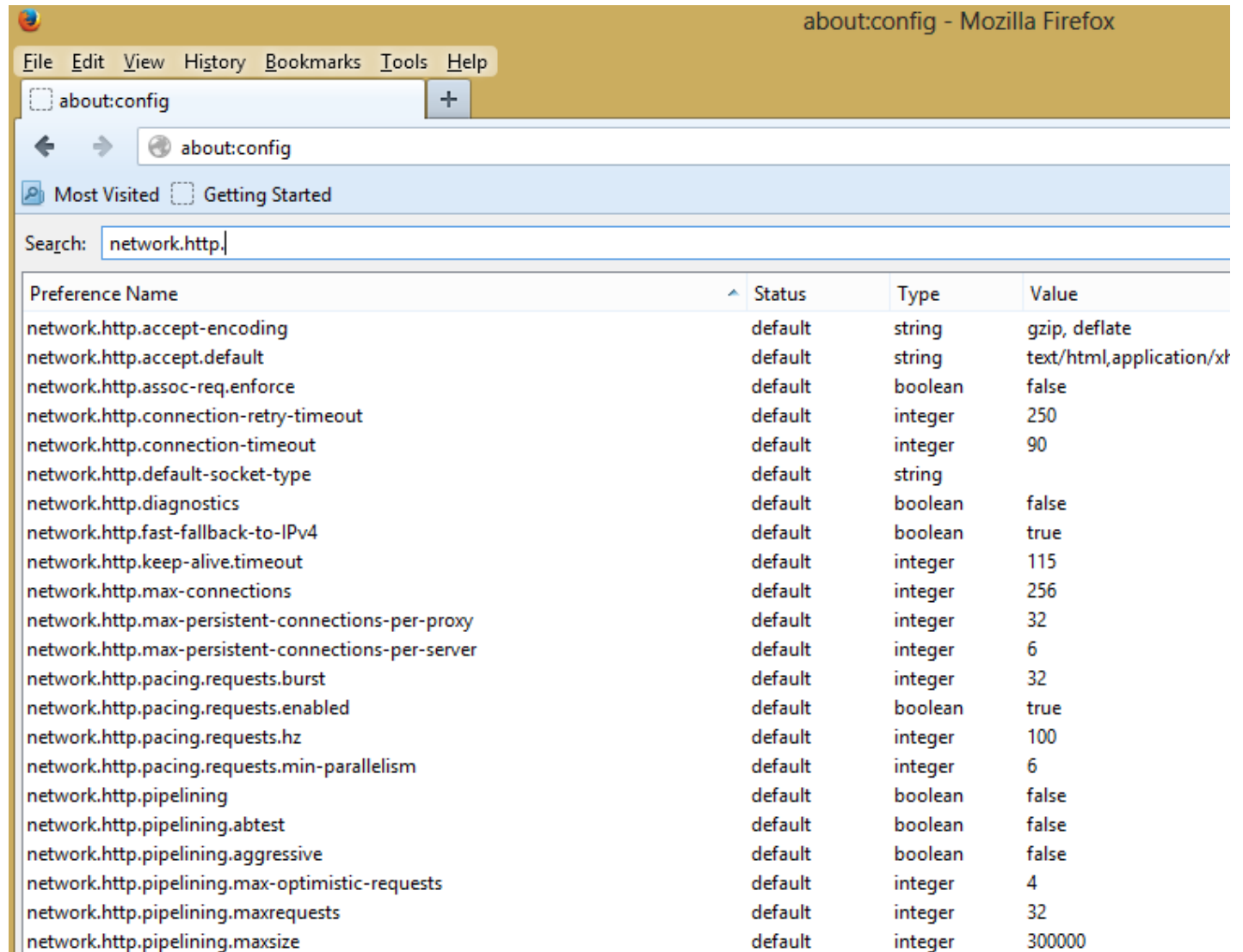
Web Browser | Download

Maximum parallel connections per host:

| BROWSER | MAX PARALLEL CONNECTIONS PER HOST |
|-----------------|-----------------------------------|
| IE 6 and 7 | 2 |
| IE 8 | 6 |
| IE 9 | 6 |
| IE 10 | 8 |
| Firefox 2 | 2 |
| Firefox 3 | 6 |
| Firefox 4 to 17 | 6 |
| Opera 9.63 | 4 |
| Opera 10 | 8 |
| Opera 11 and 12 | 6 |
| Chrome 1 and 2 | 6 |
| Chrome 3 | 4 |
| Chrome 4 to 23 | 6 |
| Safari 3 and 4 | 4 |

Web Browser | Download

Firefox:
adjusting the
parameters
related to
HTTP
connection
via URI
schema
about:config

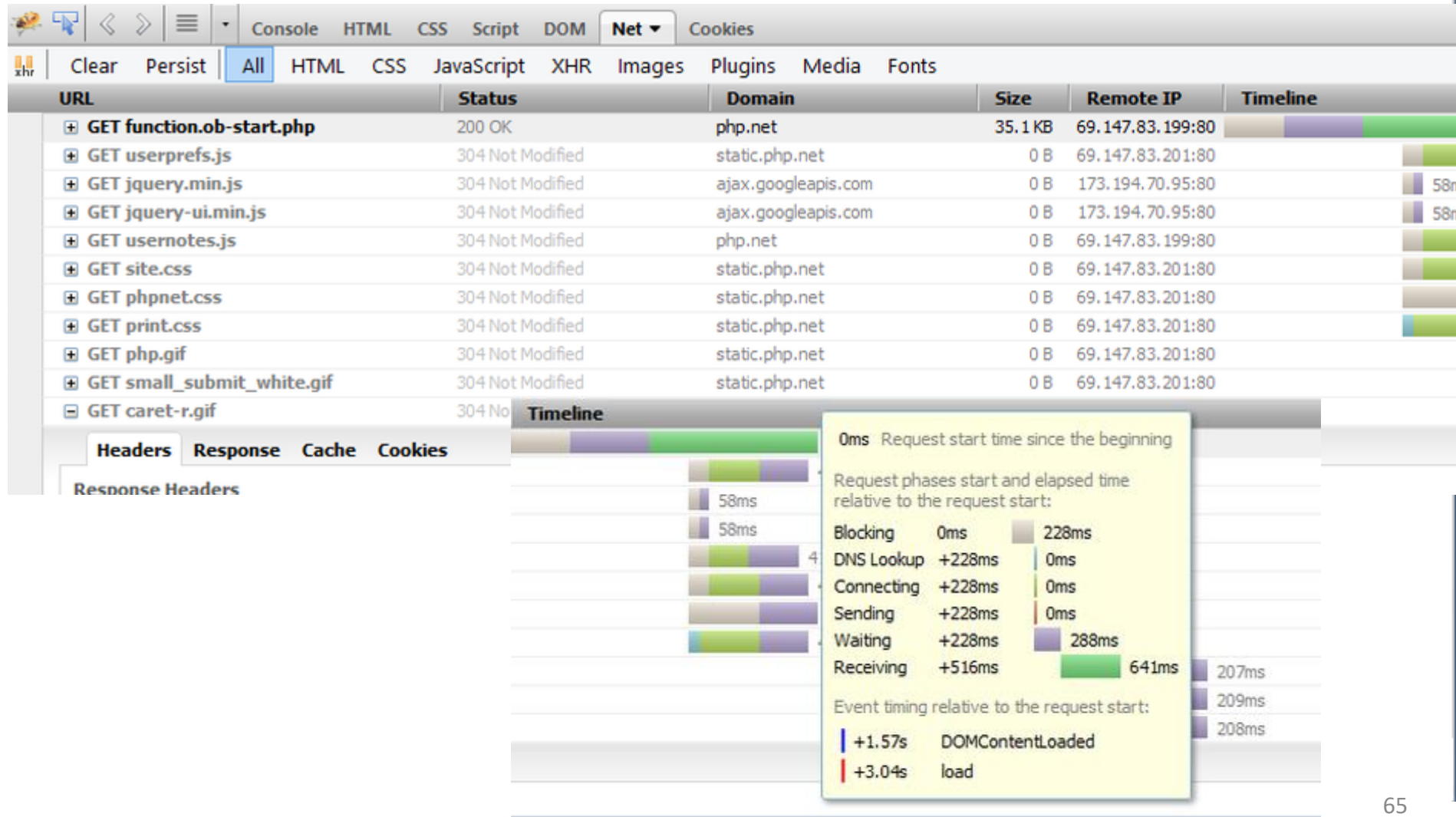


The screenshot shows the Mozilla Firefox 'about:config' page. The search bar contains 'network.http'. A table lists various network-related preferences, their status, type, and value.

| Preference Name | Status | Type | Value |
|--|---------|---------|--------------------------|
| network.http.accept-encoding | default | string | gzip, deflate |
| network.http.accept.default | default | string | text/html,application/xl |
| network.http.assoc-req.enforce | default | boolean | false |
| network.http.connection-retry-timeout | default | integer | 250 |
| network.http.connection-timeout | default | integer | 90 |
| network.http.default-socket-type | default | string | |
| network.http.diagnostics | default | boolean | false |
| network.http.fast-fallback-to-IPv4 | default | boolean | true |
| network.http.keep-alive.timeout | default | integer | 115 |
| network.http.max-connections | default | integer | 256 |
| network.http.max-persistent-connections-per-proxy | default | integer | 32 |
| network.http.max-persistent-connections-per-server | default | integer | 6 |
| network.http.pacing.requests.burst | default | integer | 32 |
| network.http.pacing.requests.enabled | default | boolean | true |
| network.http.pacing.requests.hz | default | integer | 100 |
| network.http.pacing.requests.min-parallelism | default | integer | 6 |
| network.http.pipelining | default | boolean | false |
| network.http.pipelining.abtest | default | boolean | false |
| network.http.pipelining.aggressive | default | boolean | false |
| network.http.pipelining.max-optimistic-requests | default | integer | 4 |
| network.http.pipelining.maxrequests | default | integer | 32 |
| network.http.pipelining.maxsize | default | integer | 300000 |

Web Browser | Download

Firefox → Firebug



HTTP

- **Hyper Text Transfer Protocol**

HTTP connections are persistent

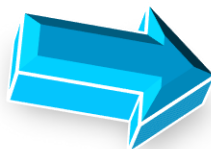
⇒ Close and open fewer TCP connections => the time of the CPU is saved in routers and hosts (clients, servers, proxies, ...) and the used memory is economized

⇒ Clients can make multiple requests within the same connections without waiting for answers for each of them

⇒ The network congestion is reduced thanks to the small numbers of packages

⇒ The requests are solved faster because there is no need of having a *handshake for each request*

**More
details?**



Web Technologies Course

Summary

- **Application Level protocols**
 - Preliminaries
 - Design Features
 - Access to the remote terminal
 - Electronic mail
 - SMTP (Simple Mail Transfer Protocol)
 - POP (Post Office Protocol)
 - File transfer
 - TFTP (Trivial File Transfer Protocol)
 - FTP (File Transfer Protocol)
 - World-Wide Web (HTTP)

Bibliography

Content Networking Fundamentals, Silvano Da Ros, Publisher: Cisco Press Pub
Date: March 30, 2006 Print ISBN-10: 1-58705-240-7 Print ISBN-13: 978-1-58705-240-8 Pages: 576

Computer and Communication Networks, Nader F. Mir, Publisher: Prentice Hall Pub
Date: November 02, 2006 Print ISBN-10: 0-13-174799-1 Print ISBN-13: 978-0-13-174799-9 Pages: 656

TCP/IP Tutorial and Technical Overview, IBM, 2006

Network + Guide to Networks, Tamara Dean, 2009



Questions?

Questions?