universidade de aveiro
theoria poiesis praxis

Camada Protocolar de Aplicação (Application Layer)

Redes de Comunicações 1

Licenciatura em Engenharia de Computadores e Informática

DETI-UA, 2023/2024

TFTP - TRIVIAL FILE TRANSFER PROTOCOL

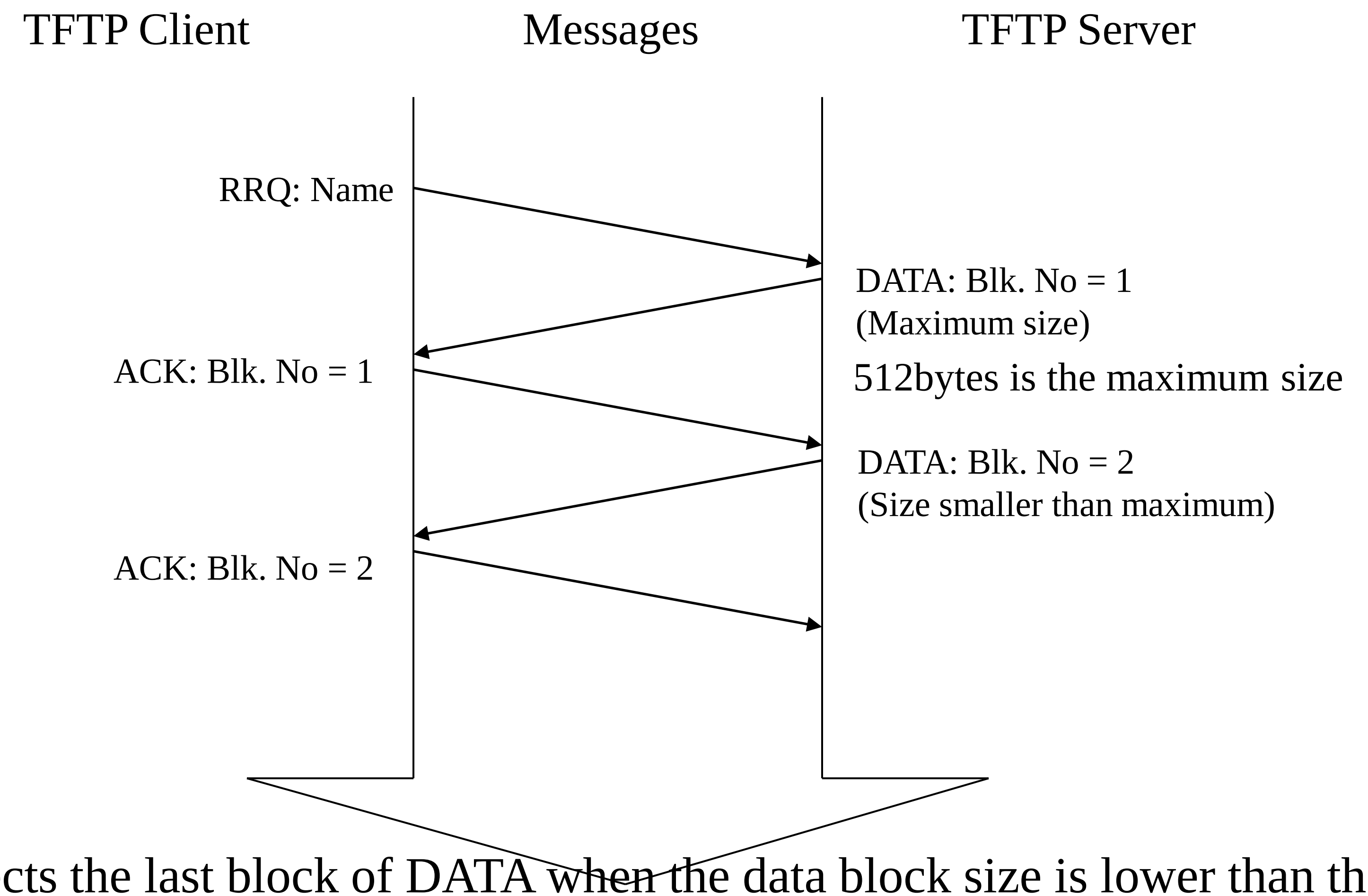
Trivial File Transfer Protocol (TFTP)

- Transfer file service with very simple client/server interactions.
- A TFTP client, to receive a file, needs to know the name and directory where the file is stored.
- This service can be used to configure network elements.
- Most *Routers* and *Switches* allow its configuration to be performed through a TFTP server to receive the configuration file.
  - They need the IP address of the server and the name of the file.
  - Very small files.

Trivial File Transfer Protocol (TFTP)

- Basic file transfer service (IETF RFC 1350)
  - It does not allow to list server files
  - It does not support authentication
- TFTP runs over **UDP**
  - The initial client message is sent to server port number **69**
  - **TFTP is the one that has to address packet retransmission**
  - TFTP server answers from another locally selected port number
  - The following messages are exchanged with the selected port number
- TFTP uses *Stop and Wait* ARQ mechanism
- TFTP has 5 messages:
  - *Read Request* (RRQ)
  - *Write Request* (WRQ)
  - *Data*
  - *Acknowledgement* (ACK)
  - *Error* (ERR)

Read Request session



TFTP Client

Messages

TFTP Server

RRQ: Name

DATA: Blk. No = 1
(Maximum size)

512bytes is the maximum size

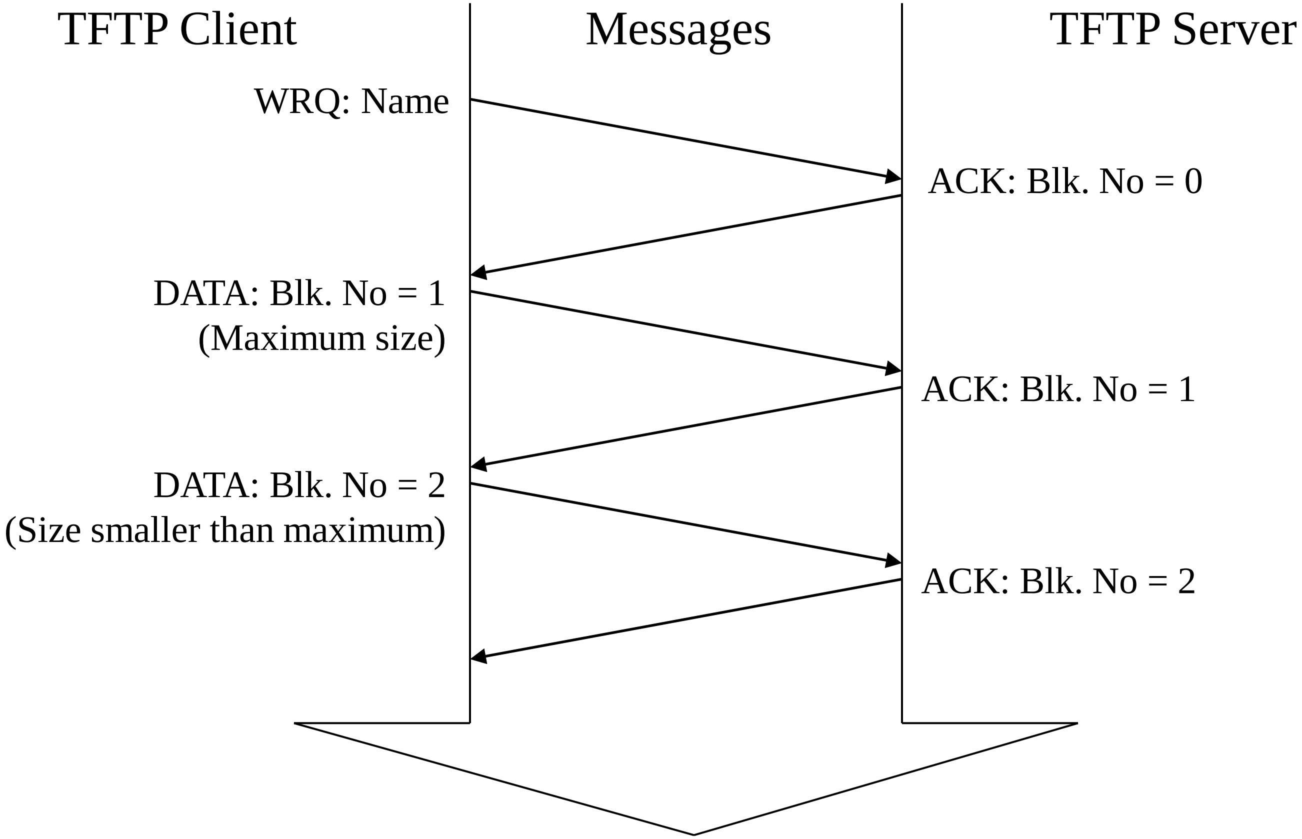
ACK: Blk. No = 1

DATA: Blk. No = 2
(Size smaller than maximum)

ACK: Blk. No = 2

Client detects the last block of DATA when the data block size is lower than the maximum size. If the size of the file is **multiple** of the maximum size, an aditional data block is sent with **0** bytes of data. RRQ always with odd number of packets (RRQ+pairs of data and ACK).

Write Request session



Server detects the last block of DATA when the data block size is lower than the maximum size. ThIfe the size of the file is **multiple** of the maximum size, an aditional data block is sent with **0** bytes of data. WRQ always with even number of packets (WRQ+ACK0+pairs of data and ACK).

Messages format

2 bytes	n bytes	1 byte	n bytes	1 byte
READ REQ (1)	FILENAME	0	MODE	0

2 bytes	n bytes	1 byte	n bytes	1 bytes
WRITE REQ (2)	FILENAME	0	MODE	0

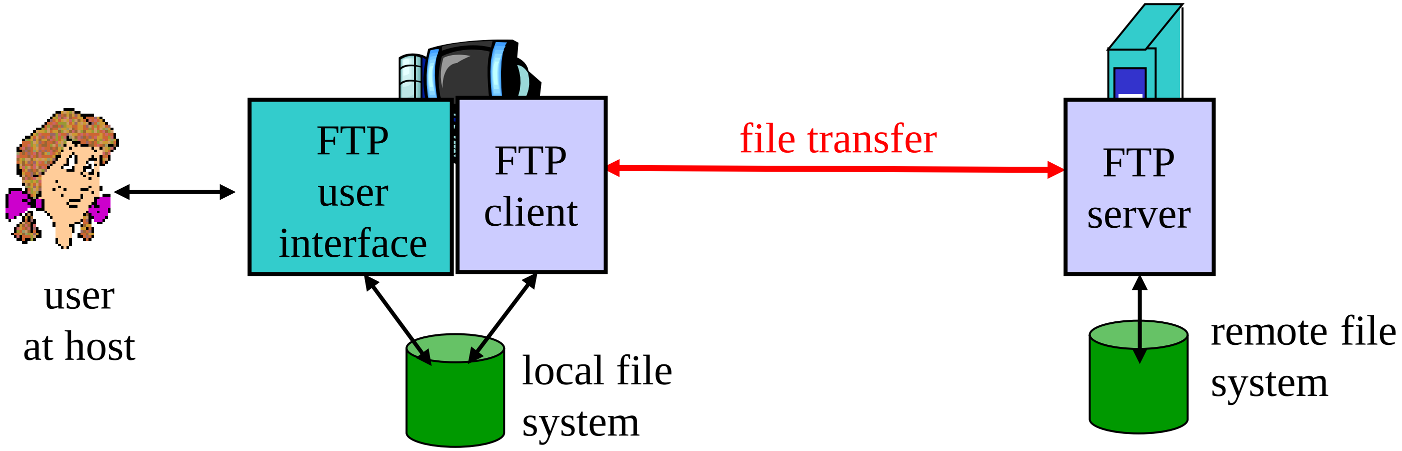
2 bytes	2 bytes	Up to 512 bytes
DATA (3)	BLOCK #	data

2 bytes	2 bytes
ACK (4)	BLOCK #

2 butes	2 bytes	n bytes	1 byte
ERROR (5)	ERROR CODE	ERROR MESSAGE	0

FTP – FILE TRANSFER PROTOCOL

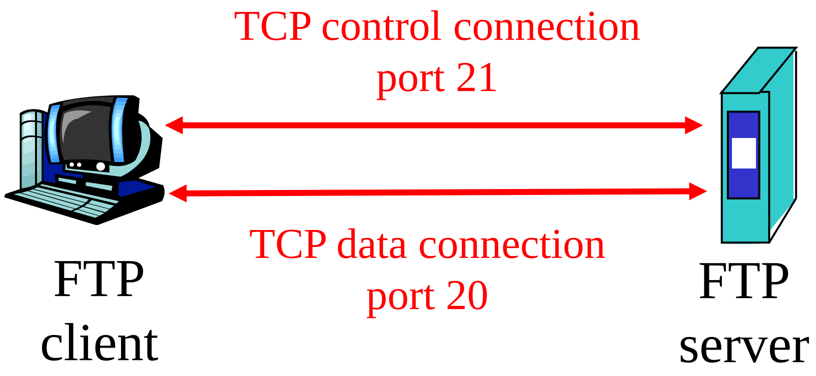
FTP: the file transfer protocol



- transfer file to/from remote host (IETF RFC 959)
- client/server model
  - *client:* side that initiates file transfer (either to/from remote host)
  - *server:* remote host
- ftp: runs over **TCP (TCP takes care of packet retransmission)**
- ftp server ports: **21** and **20**
  - **21** for the control connection
  - **20** for each data connection

FTP: separate control, data connections

- FTP client contacts FTP server at port 21, TCP is transport protocol
- client authorized over control connection
- client browses remote directory by sending commands over control connection
- when server receives file transfer command, server opens 2nd TCP connection (for file transfer) to client
- after transferring one file, server closes data connection.



TCP control connection port 21

TCP data connection port 20

FTP client

FTP server

- server opens another TCP data connection to transfer another file.
- control connection: "out of band"
- FTP server maintains the "state" of client interactions: current directory, earlier authentication, etc…

FTP commands, responses

Sample commands:

- sent as ASCII text over control channel:

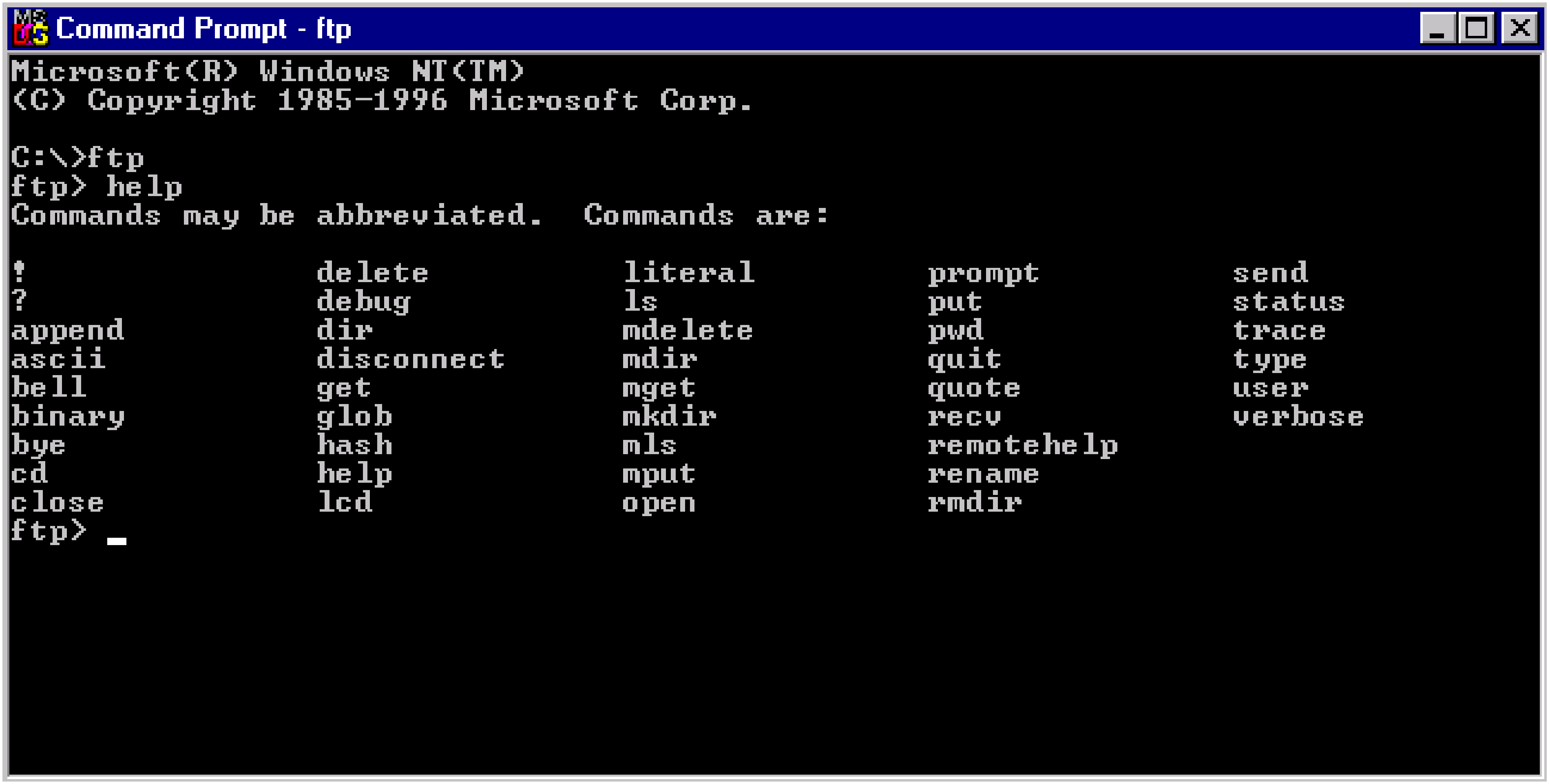
- **USER *username***
- **PASS *password***
- **LIST** (return list of files in current directory)
- **RETR filename** (retrieves file from server)
- **STOR filename** (stores file onto server)

Sample return codes

- status code and phrase (as in HTTP):

- **331 Username OK, password required**
- **125 Data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

FTP client: user commands

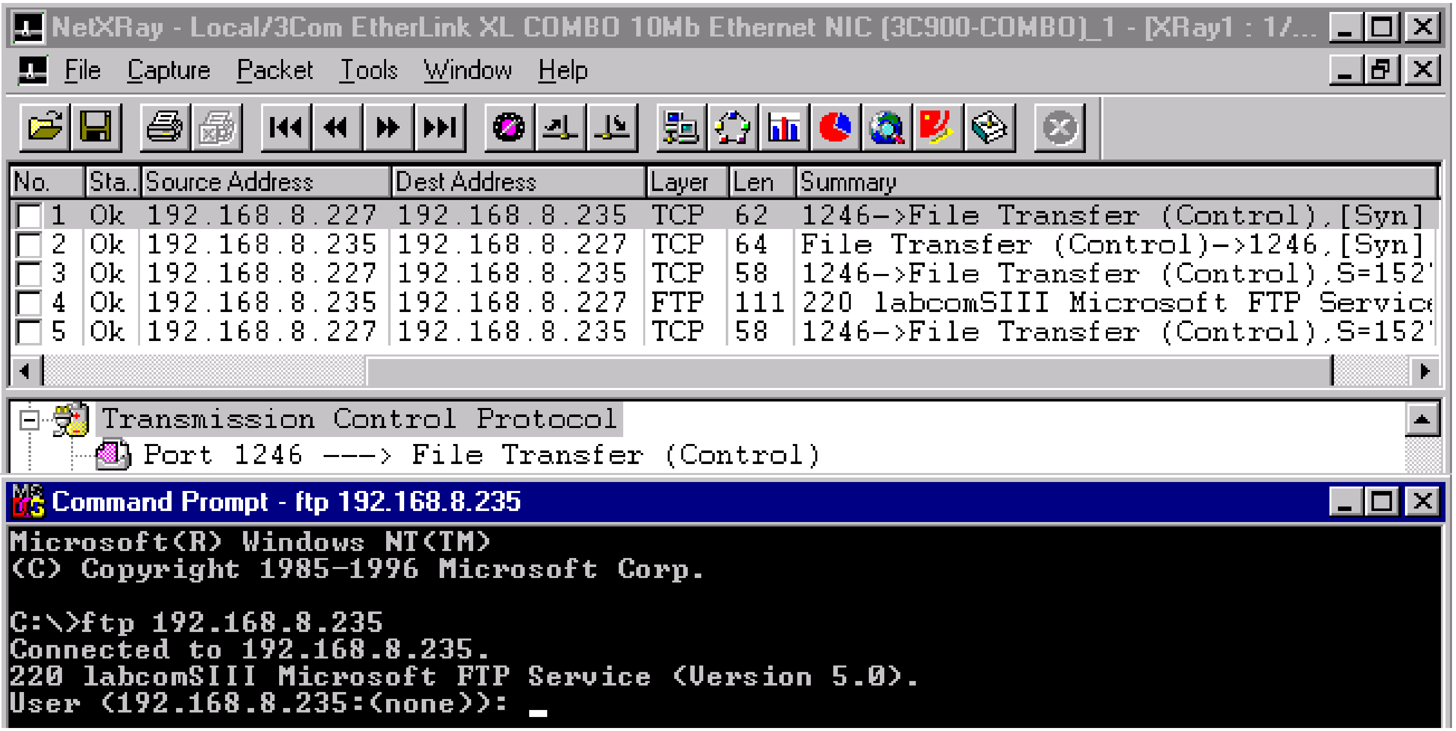


```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>ftp
ftp> help
Commands may be abbreviated.  Commands are:

!               delete          literal         prompt          send
?               debug           ls              put             status
append          dir             mdelete         pwd             trace
ascii           disconnect      mdir            quit            type
bell            get             mget            quote           user
binary          glob            mkdir           recv            verbose
bye             hash            mls             remotehelp
cd              help            mput            rename
close           lcd             open            rmdir
ftp> _
```

Initial connection to FTP server


NetXRay - Local/3Com EtherLink XL COMBO 10Mb Ethernet NIC (3C900-COMBO)_1 - [XRay1 : 1/...
File Capture Packet Tools Window Help
Command Prompt - ftp 192.168.8.235

No.	Sta..	Source Address	Dest Address	Layer	Len	Summary
□ 1	Ok	192.168.8.227	192.168.8.235	TCP	62	1246->File Transfer (Control),[Syn]
□ 2	Ok	192.168.8.235	192.168.8.227	TCP	64	File Transfer (Control)->1246,[Syn]
□ 3	Ok	192.168.8.227	192.168.8.235	TCP	58	1246->File Transfer (Control),S=152
□ 4	Ok	192.168.8.235	192.168.8.227	FTP	111	220 labcomSIII Microsoft FTP Servic
□ 5	Ok	192.168.8.227	192.168.8.235	TCP	58	1246->File Transfer (Control),S=152

Transmission Control Protocol
Port 1246 ---> File Transfer (Control)

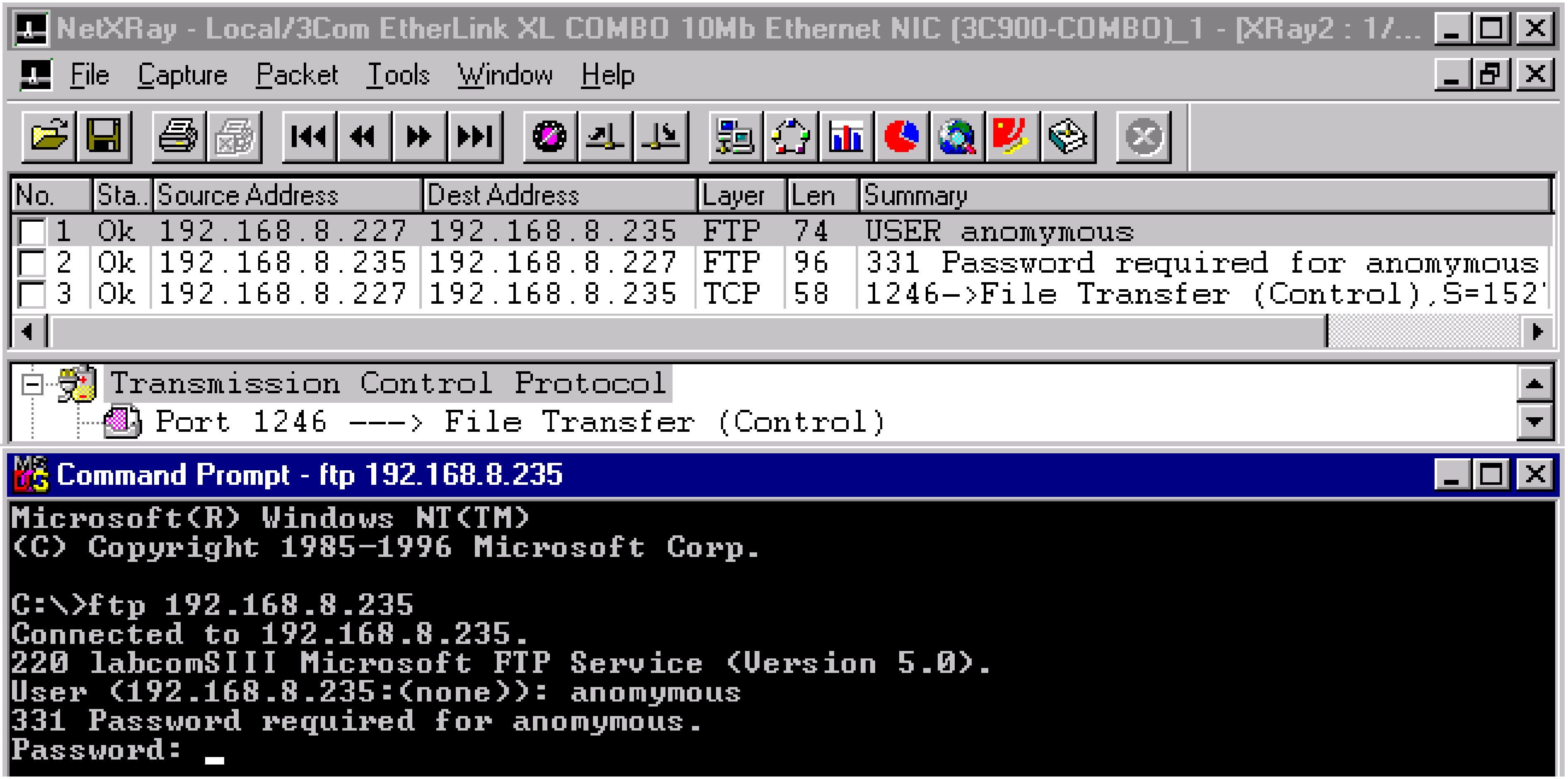
```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>ftp 192.168.8.235
Connected to 192.168.8.235.
220 labcomSIII Microsoft FTP Service (Version 5.0).
User (192.168.8.235:(none)): _
```

TCP 3-way handshake to initiate the control connection in port 21

For each packet sent, a TCP ACK is needed to confirm the previous packet

Introduction of username



NetXRay - Local/3Com EtherLink XL COMBO 10Mb Ethernet NIC (3C900-COMBO)_1 - [XRay2 : 1/...
File Capture Packet Tools Window Help

No.	Sta..	Source Address	Dest Address	Layer	Len	Summary
□ 1	Ok	192.168.8.227	192.168.8.235	FTP	74	USER anomymous
□ 2	Ok	192.168.8.235	192.168.8.227	FTP	96	331 Password required for anomymous
□ 3	Ok	192.168.8.227	192.168.8.235	TCP	58	1246->File Transfer (Control),S=152'

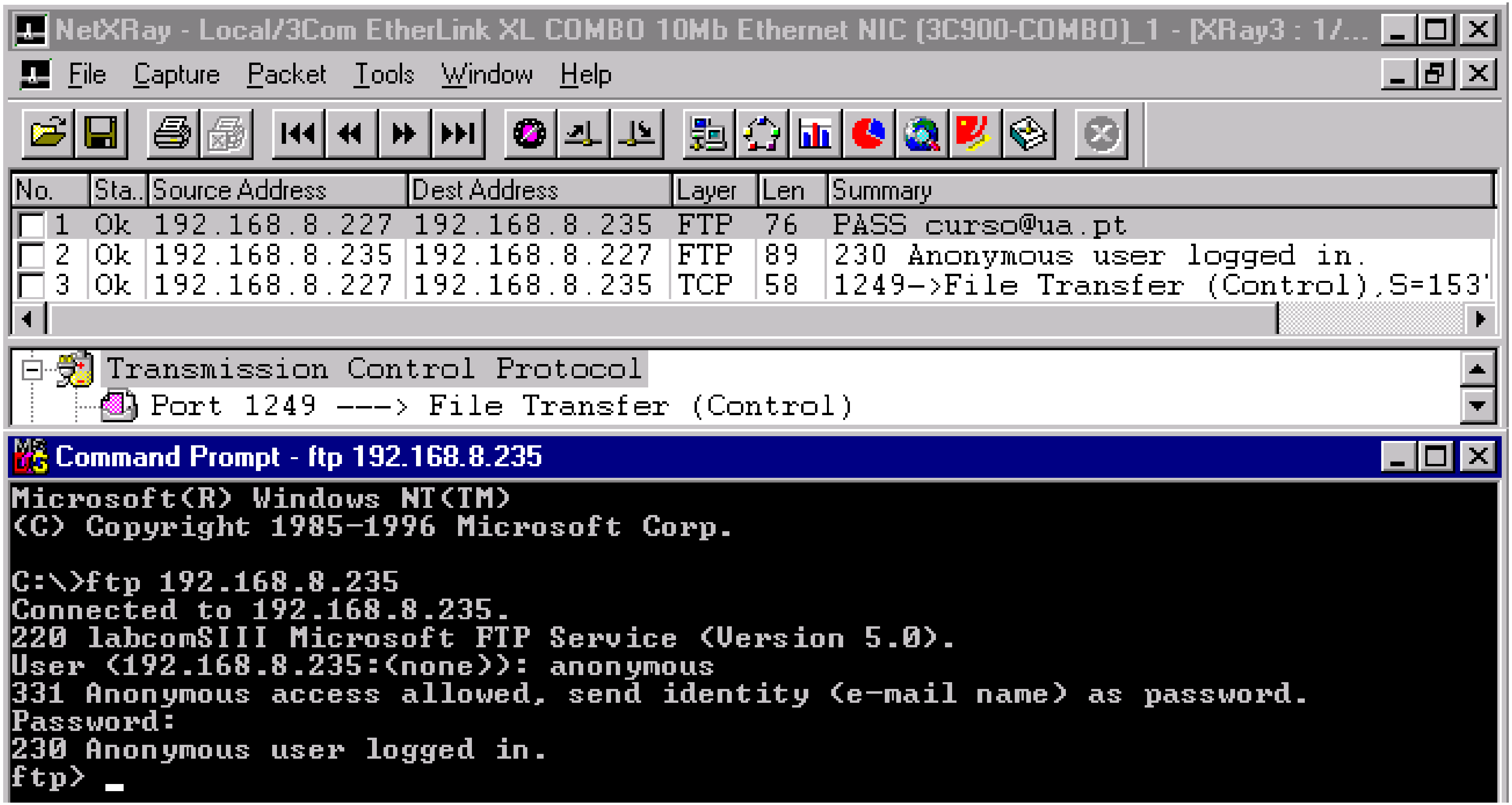
```
Transmission Control Protocol
    Port 1246 ---> File Transfer (Control)
```

Command Prompt - ftp 192.168.8.235

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>ftp 192.168.8.235
Connected to 192.168.8.235.
220 labcomSIII Microsoft FTP Service (Version 5.0).
User (192.168.8.235:(none)): anomymous
331 Password required for anomymous.
Password: _
```

Introduction of password


NetXRay - Local/3Com EtherLink XL COMBO 10Mb Ethernet NIC (3C900-COMBO)_1 - [XRay3 : 1/...
File Capture Packet Tools Window Help
Command Prompt - ftp 192.168.8.235

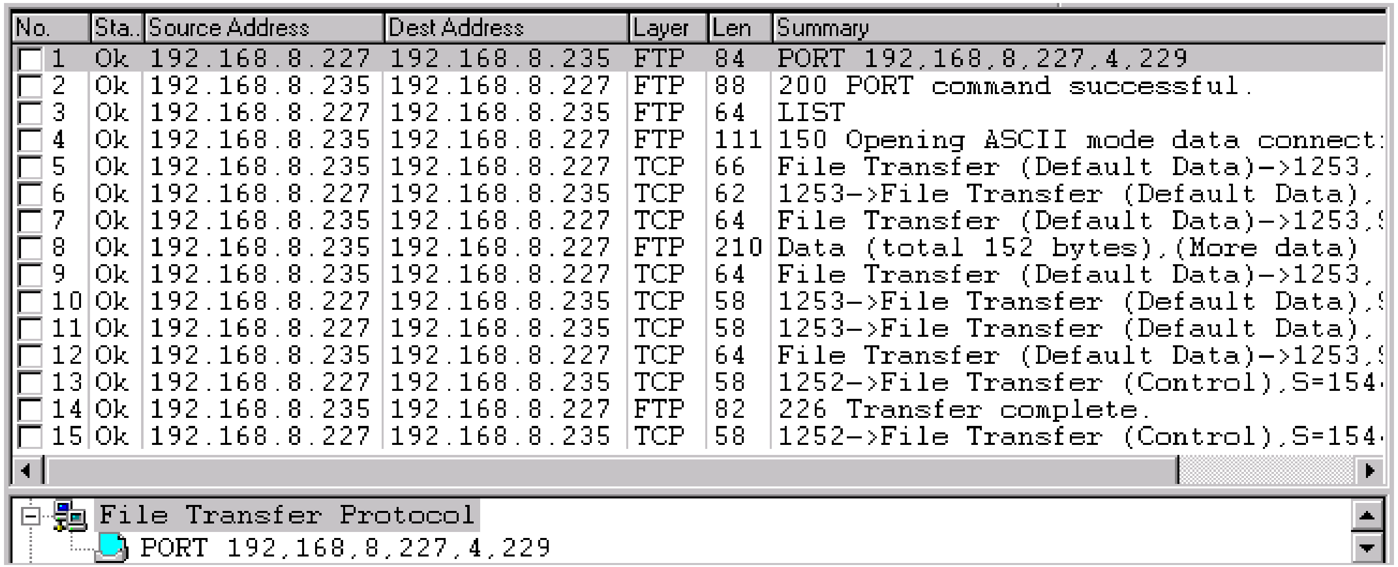
No.	Sta..	Source Address	Dest Address	Layer	Len	Summary
□ 1	Ok	192.168.8.227	192.168.8.235	FTP	76	PASS curso@ua.pt
□ 2	Ok	192.168.8.235	192.168.8.227	FTP	89	230 Anonymous user logged in.
□ 3	Ok	192.168.8.227	192.168.8.235	TCP	58	1249-->File Transfer (Control),S=153'

Transmission Control Protocol
Port 1249 ---> File Transfer (Control)

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>ftp 192.168.8.235
Connected to 192.168.8.235.
220 labcomSIII Microsoft FTP Service (Version 5.0).
User (192.168.8.235:(none)): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 Anonymous user logged in.
ftp> _
```

Introduction of DIR command



No.	Sta.	Source Address	Dest Address	Layer	Len	Summary
□ 1	Ok	192.168.8.227	192.168.8.235	FTP	84	PORT 192,168,8,227,4,229
□ 2	Ok	192.168.8.235	192.168.8.227	FTP	88	200 PORT command successful.
□ 3	Ok	192.168.8.227	192.168.8.235	FTP	64	LIST
□ 4	Ok	192.168.8.235	192.168.8.227	FTP	111	150 Opening ASCII mode data connect
□ 5	Ok	192.168.8.235	192.168.8.227	TCP	66	File Transfer (Default Data)->1253,
□ 6	Ok	192.168.8.227	192.168.8.235	TCP	62	1253->File Transfer (Default Data),
□ 7	Ok	192.168.8.235	192.168.8.227	TCP	64	File Transfer (Default Data)->1253,
□ 8	Ok	192.168.8.235	192.168.8.227	FTP	210	Data (total 152 bytes),(More data)
□ 9	Ok	192.168.8.235	192.168.8.227	TCP	64	File Transfer (Default Data)->1253,
□ 10	Ok	192.168.8.227	192.168.8.235	TCP	58	1253->File Transfer (Default Data),
□ 11	Ok	192.168.8.227	192.168.8.235	TCP	58	1253->File Transfer (Default Data),
□ 12	Ok	192.168.8.235	192.168.8.227	TCP	64	File Transfer (Default Data)->1253,
□ 13	Ok	192.168.8.227	192.168.8.235	TCP	58	1252->File Transfer (Control),S=154
□ 14	Ok	192.168.8.235	192.168.8.227	FTP	82	226 Transfer complete.
□ 15	Ok	192.168.8.227	192.168.8.235	TCP	58	1252->File Transfer (Control),S=154

```
File Transfer Protocol
    PORT 192,168,8,227,4,229
```

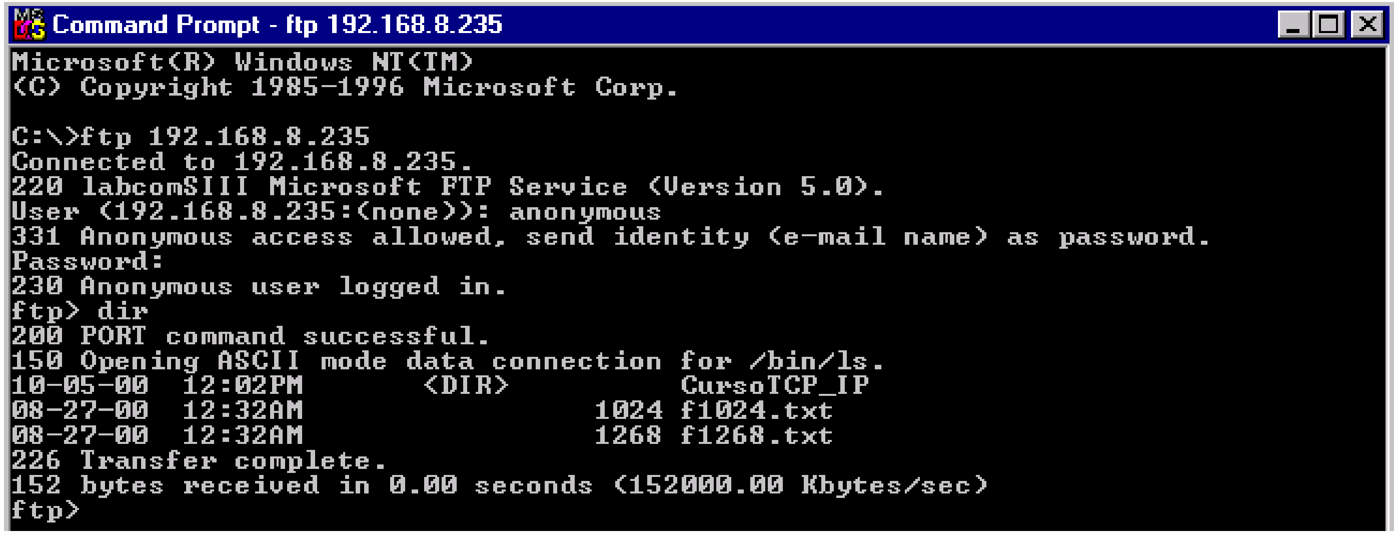
Information to start the data connection

FTP command for DIR

TCP 3-way handshake to initiate the data connection in port 20

Data is sent

FIN and ACK packets (4 in total) to finish the data connection in port 20

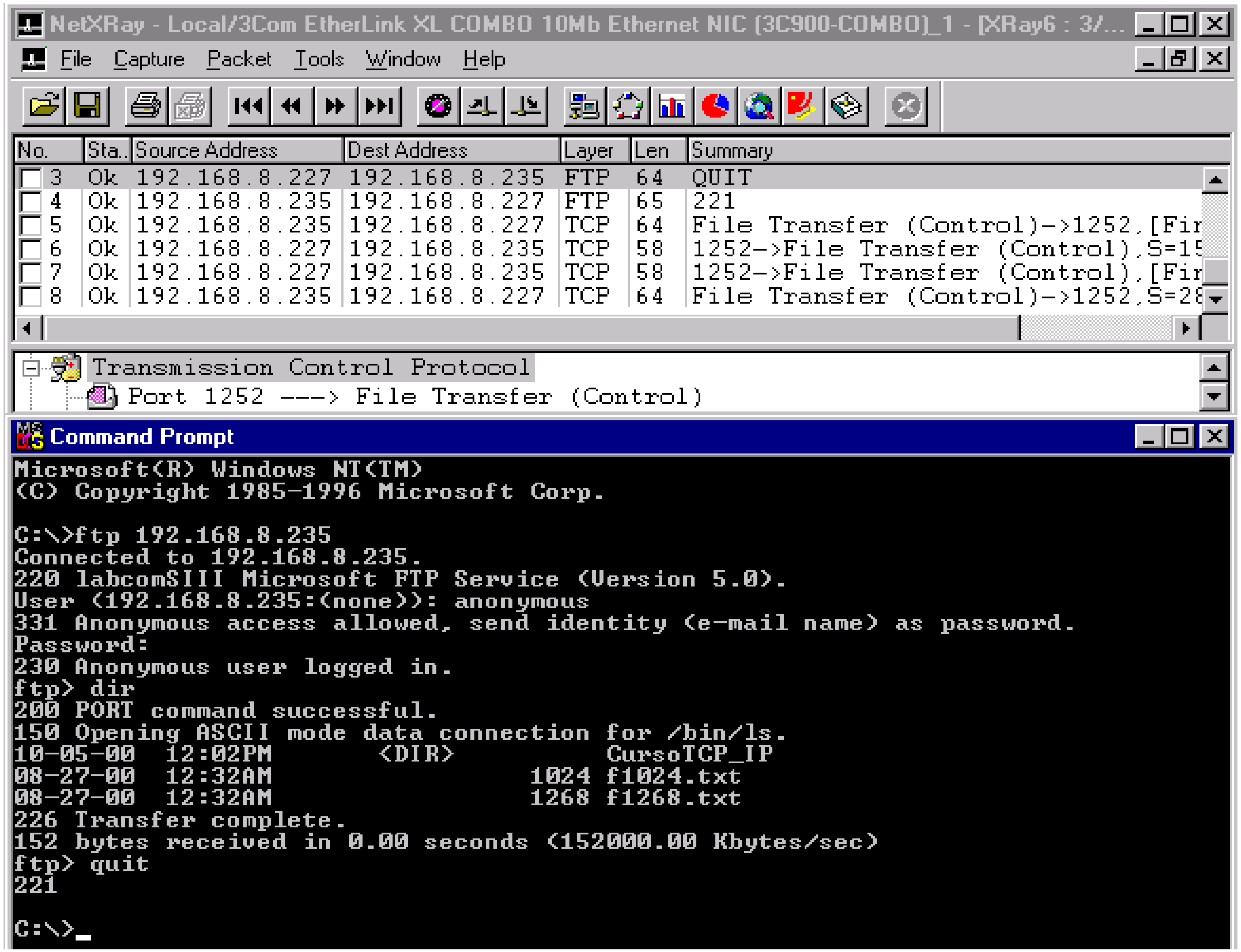


Command Prompt - ftp 192.168.8.235

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>ftp 192.168.8.235
Connected to 192.168.8.235.
220 labcomSIII Microsoft FTP Service (Version 5.0).
User (192.168.8.235:(none)): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 Anonymous user logged in.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
10-05-00  12:02PM       <DIR>          CursoTCP_IP
08-27-00  12:32AM                 1024 f1024.txt
08-27-00  12:32AM                 1268 f1268.txt
226 Transfer complete.
152 bytes received in 0.00 seconds (152000.00 Kbytes/sec)
ftp>
```

Termination of connection (quit)



NetXRay - Local/3Com EtherLink XL COMBO 10Mb Ethernet NIC (3C900-COMBO)_1 - [XRay6 : 3/...
File
Capture
Packet
Tools
Window
Help

No.	Sta.	Source Address	Dest Address	Layer	Len	Summary
□ 3	Ok	192.168.8.227	192.168.8.235	FTP	64	QUIT
□ 4	Ok	192.168.8.235	192.168.8.227	FTP	65	221
□ 5	Ok	192.168.8.235	192.168.8.227	TCP	64	File Transfer (Control)->1252,[Fir
□ 6	Ok	192.168.8.227	192.168.8.235	TCP	58	1252->File Transfer (Control),S=15
□ 7	Ok	192.168.8.227	192.168.8.235	TCP	58	1252->File Transfer (Control),[Fir
□ 8	Ok	192.168.8.235	192.168.8.227	TCP	64	File Transfer (Control)->1252,S=28

```
⊟ Transmission Control Protocol
   Port 1252 ---> File Transfer (Control)
```

```
Microsoft(R) Windows NT(TM)
(C) Copyright 1985-1996 Microsoft Corp.

C:\>ftp 192.168.8.235
Connected to 192.168.8.235.
220 labcomSIII Microsoft FTP Service (Version 5.0).
User (192.168.8.235:(none)): anonymous
331 Anonymous access allowed, send identity (e-mail name) as password.
Password:
230 Anonymous user logged in.
ftp> dir
200 PORT command successful.
150 Opening ASCII mode data connection for /bin/ls.
10-05-00  12:02PM       <DIR>          CursoTCP_IP
08-27-00  12:32AM                 1024 f1024.txt
08-27-00  12:32AM                 1268 f1268.txt
226 Transfer complete.
152 bytes received in 0.00 seconds (152000.00 Kbytes/sec)
ftp> quit
221

C:\>_
```

FIN and ACK packets (4 in total) to finish the control connection in port 21

DNS – DOMAIN NAME SYSTEM

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

- IP address (32 bit) - used for addressing datagrams
- "name" (for example, www.yahoo.com) - used by humans

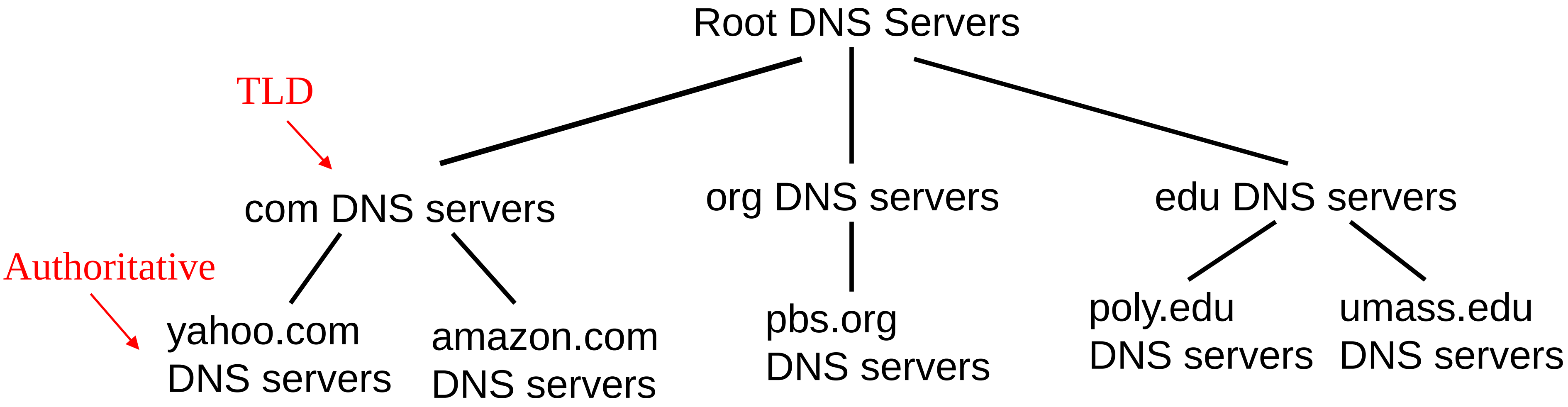
DNS: provides a mapping between IP addresses and names

Works in **UDP port 53**

Domain Name System:

- *distributed database* implemented in a hierarchy of many *Name Servers*
- *application-layer protocol* hosts communicate with Name Servers to *resolve* names (name - IP address translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

Distributed, Hierarchical Database



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

- client queries a 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

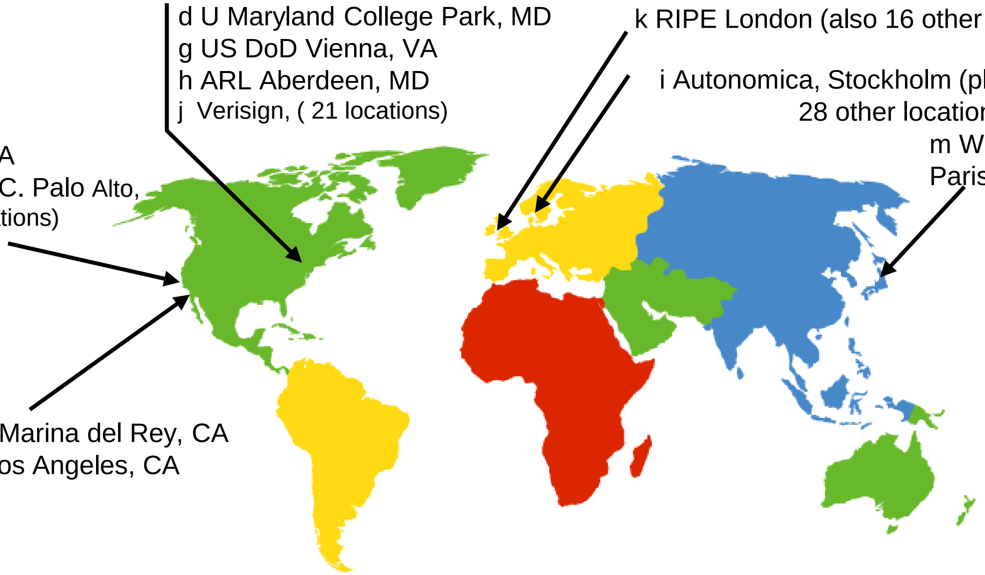
Server	Operator	Cities	IP Addr	Home AS	Answers ICMP?
A	VeriSign Global Registry Services	Herndon VA, US	198.41.0.4	19836	*yes*
B	Information Sciences Institute	Marina Del Rey CA, US	128.9.0.107	*tba*	*yes*
C	Cogent Communications	Herndon VA, US	192.33.4.12	2149	*yes*
D	University of Maryland	College Park MD, US	128.8.10.90	27	*yes*
E	NASA Ames Research Center	Mountain View CA, US	192.203.230.10	297	*yes*

F	Internet Software Consortium	Palo Alto CA, US;
		San Francisco CA, US;
		Madrid, ES;
		San Jose, CA, US;
		New York, NY, US;
		Hong Kong, HK
			IPv4: 192.5.5.241
			IPv6: 2001:500::1035
				3557	*yes*

G	U.S. DOD Network Information Center	Vienna VA, US	192.112.36.4	568	*no*
H	U.S. Army Research Lab	Aberdeen MD, US	IPv4: 128.63.2.53 IPv6: 2001:500:1::803f:235	13	*yes*
I	Autonomica	Stockholm, SE	192.36.148.17	8674	*yes*
J	VeriSign Global Registry Services	Herndon VA, US	192.58.128.30	26415	*yes*
K	Reseaux IP Europeens - Network Coordination Centre	London, UK	193.0.14.129	5459	*yes*
L	Internet Corporation for Assigned Names and Numbers	Los Angeles CA, US	198.32.64.12	20144	*no*
M	WIDE Project	Tokyo, JP	202.12.27.33	7500	*yes*

DNS: Root Name Servers

- contacted by local Name Server that cannot resolve name



a Verisign, Dulles, VA
c Cogent, Herndon, VA (also LA)
d U Maryland College Park, MD
g US DoD Vienna, VA
h ARL Aberdeen, MD
j Verisign, (21 locations)

k RIPE London (also 16 other locations)

i Autonomica, Stockholm (plus
28 other locations)

m WIDE Tokyo (also Seoul,
Paris, SF)

e NASA Mt View, CA
f Internet Software C. Palo Alto,
CA (and 36 other locations)

b USC-ISI Marina del Rey, CA
l ICANN Los Angeles, CA

13 root name
servers worldwide

Top Level Domains (TLD)

- Organizational domains:

com	- commercial organizations
edu	- educational institutions
gov	- govern institutions
mil	- military institutions
net	- network operators
int	- international organizations
org	- other organizations

- Country domains:

pt	- Portugal
es	- Spain

Local Name Server

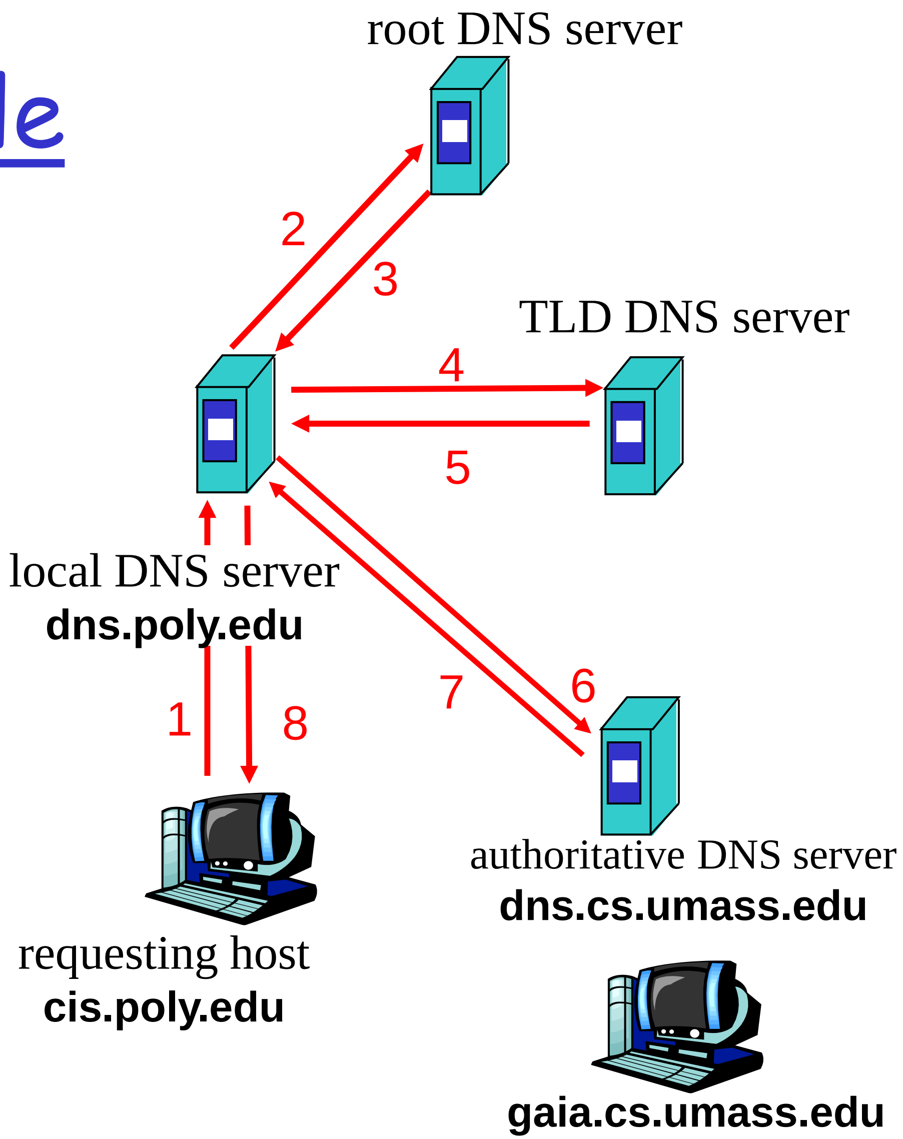
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one.
  - also called "default Name Server"
- when host makes DNS query, query is sent to its local DNS server
  - acts as proxy, forwards query into hierarchy

DNS name resolution example

- Host at cis.poly.edu wants IP address for gaia.cs.umass.edu

iterated query:

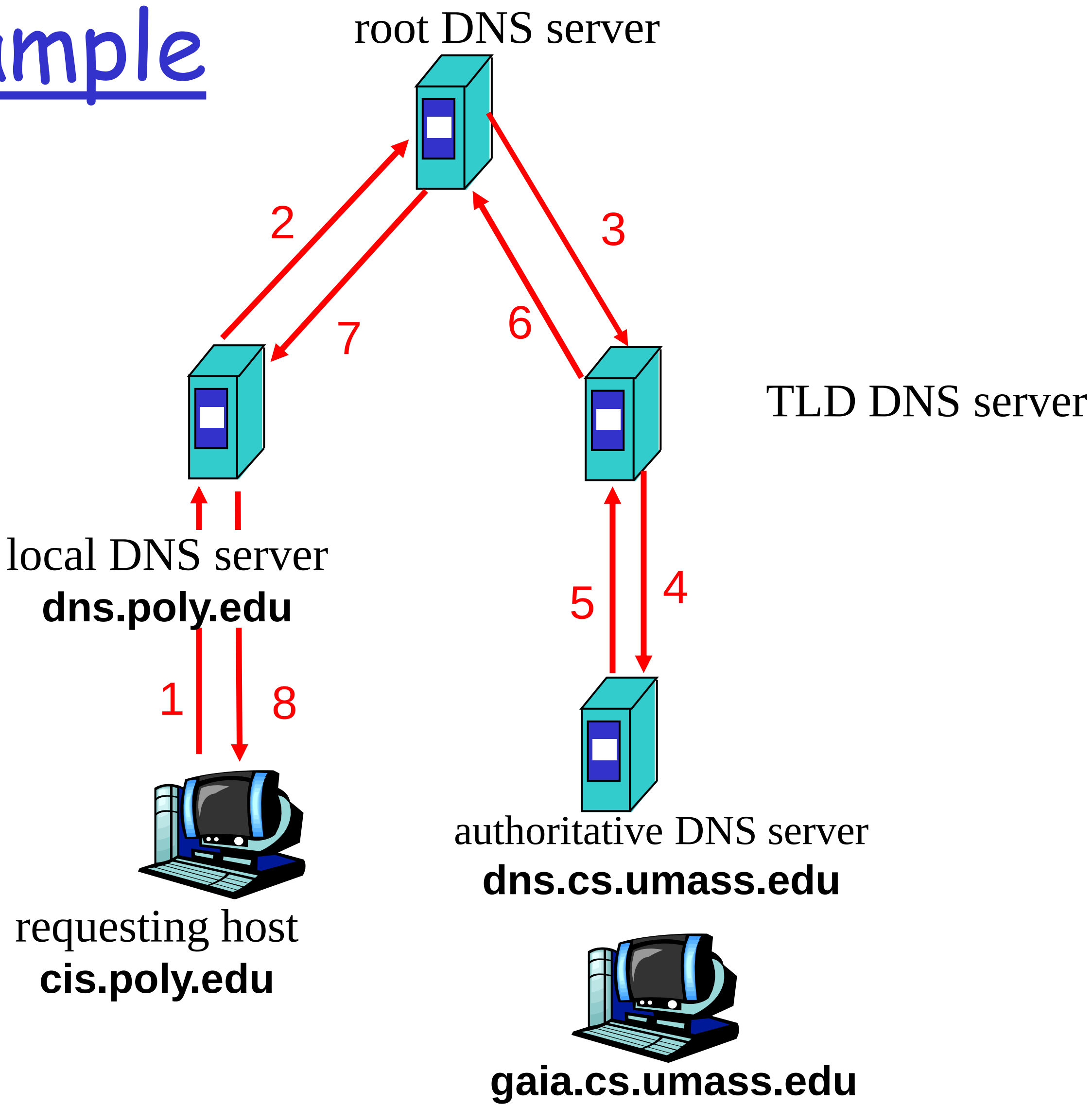
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



DNS name resolution example

recursive query:

- ❒ puts burden of name resolution on contacted name server



Recursive vs. iterated resolution

- Recursive resolution:
  - More efficient: minimizes the time between the query and the answer
  - Requires more processing power in DNS servers: each server has more simultaneous ongoing requests, on average
- Iterated resolution:
  - Less efficient: the time between the query and the answer is larger, on average
  - Minimizes the processing power required on DNS servers: each server replies immediately to each received query

DNS: caching and updating records

- once (any) Name Server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time
  - TLD server addresses are typically cached in local Name Servers
    - Thus, Root Name Servers are not often visited

DNS records

DNS: distributed database storing Resource Records (RR)

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

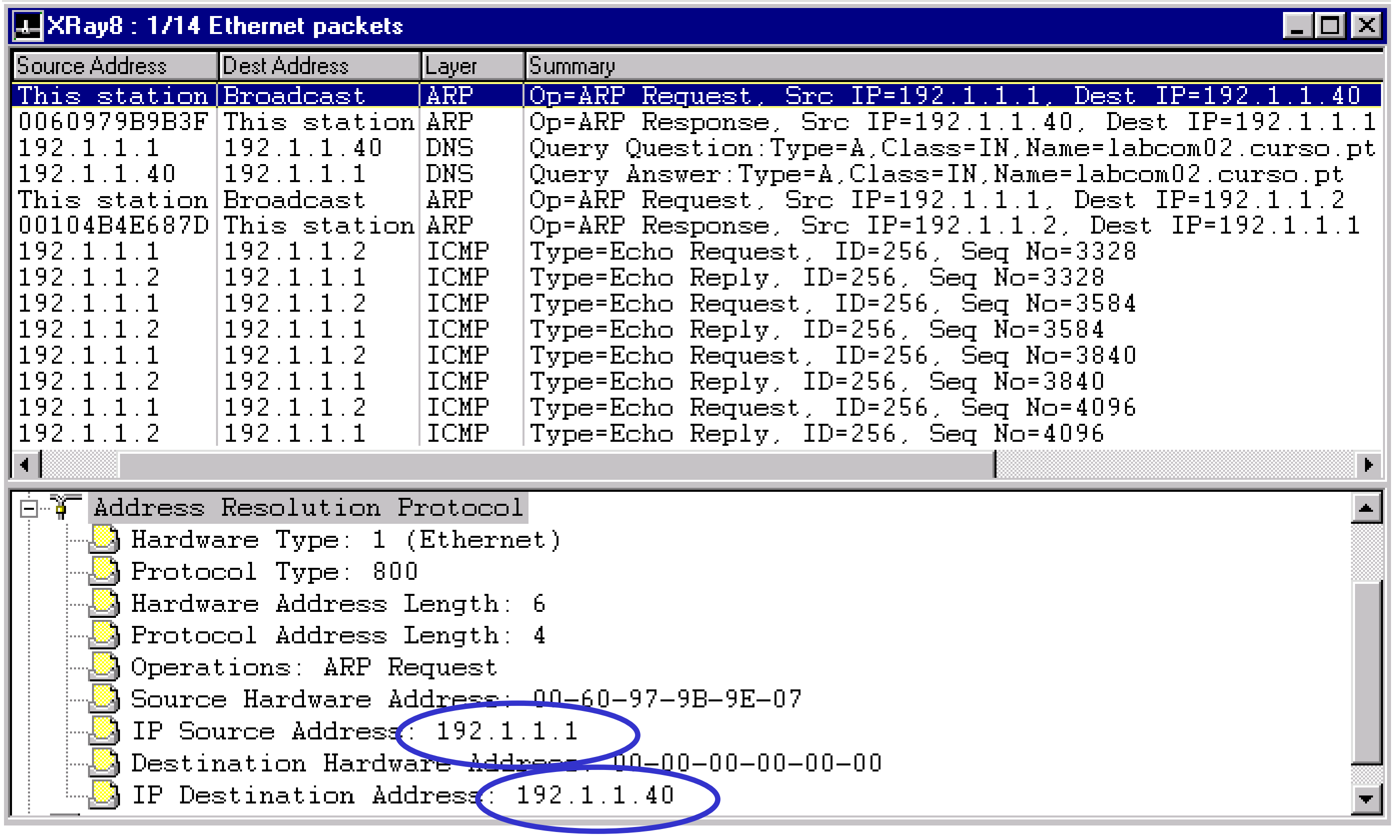
- Type=A
  - **name** is hostname
  - **value** is IP address
  - e.g. (relay1.bar.foo.com, 145.37.93.126, A)
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain
  - e.g. (foo.com, dns.foo.com, NS)
- Type=CNAME
  - **name** is alias name for some "canonical" (the real) name
  - **value** is canonical name
  - e.g. (foo.com, relay1.bar.foo.com, CNAME)
- Type=MX
  - **value** is name of mailserver associated with **name**
  - e.g. (foo.com, mail.bar.foo.com, MX)
- Type=AAAA
  - **name** is hostname
  - **value** is IPv6 address

DNS messages

host: 192.1.1.1

DNS: 192.1.1.40

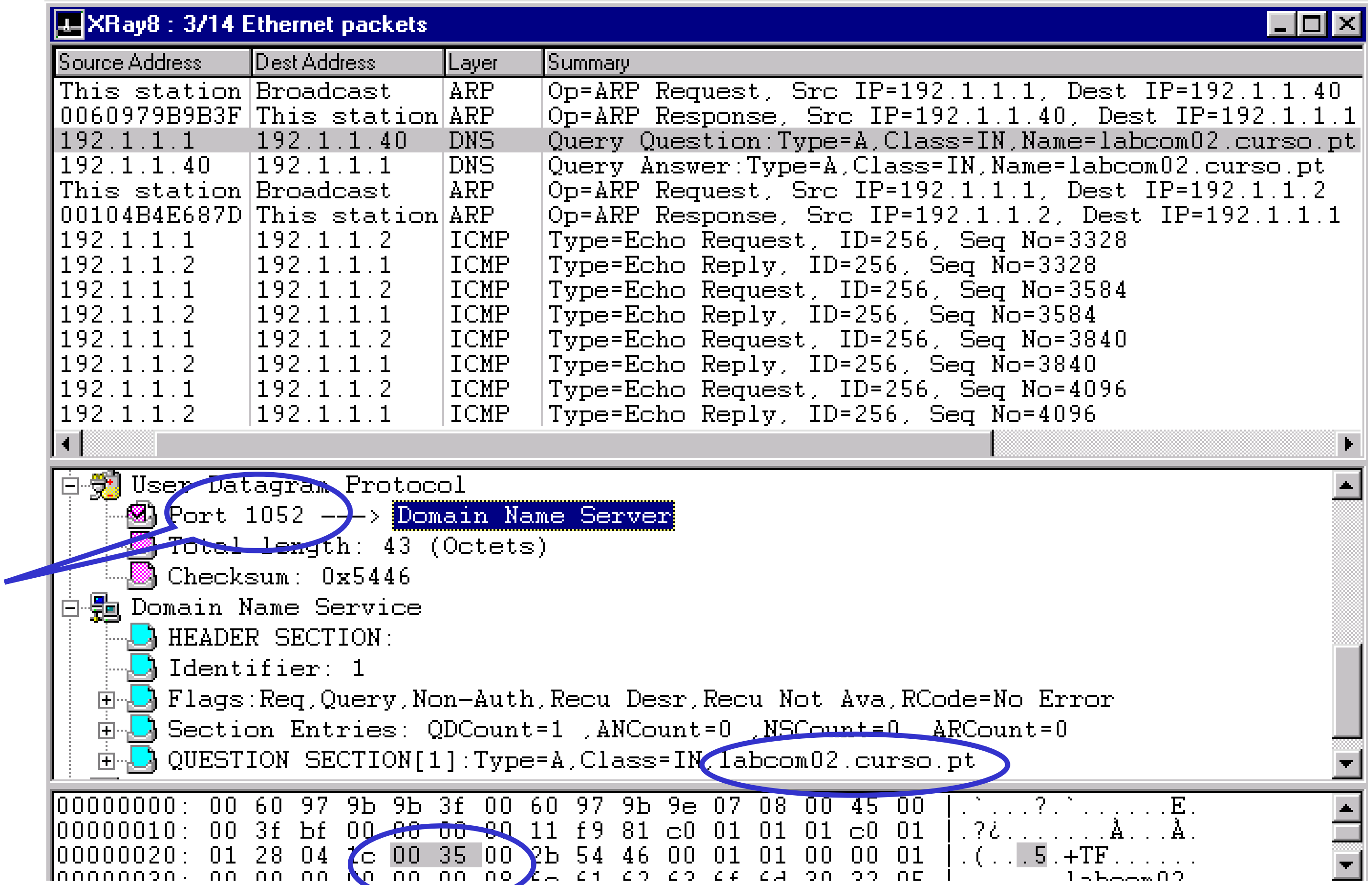
```
C:\>ping labcom02.curso.pt
```



```
Source Address Dest Address   Layer  Summary
This station   Broadcast      ARP    Op=ARP Request, Src IP=192.1.1.1, Dest IP=192.1.1.40
0060979B9B3F   This station   ARP    Op=ARP Response, Src IP=192.1.1.40, Dest IP=192.1.1.1
192.1.1.1      192.1.1.40     DNS    Query Question:Type=A,Class=IN,Name=labcom02.curso.pt
192.1.1.40     192.1.1.1      DNS    Query Answer:Type=A,Class=IN,Name=labcom02.curso.pt
This station   Broadcast      ARP    Op=ARP Request, Src IP=192.1.1.1, Dest IP=192.1.1.2
00104B4E687D   This station   ARP    Op=ARP Response, Src IP=192.1.1.2, Dest IP=192.1.1.1
192.1.1.1      192.1.1.2      ICMP   Type=Echo Request, ID=256, Seq No=3328
192.1.1.2      192.1.1.1      ICMP   Type=Echo Reply, ID=256, Seq No=3328
192.1.1.1      192.1.1.2      ICMP   Type=Echo Request, ID=256, Seq No=3584
192.1.1.2      192.1.1.1      ICMP   Type=Echo Reply, ID=256, Seq No=3584
192.1.1.1      192.1.1.2      ICMP   Type=Echo Request, ID=256, Seq No=3840
192.1.1.2      192.1.1.1      ICMP   Type=Echo Reply, ID=256, Seq No=3840
192.1.1.1      192.1.1.2      ICMP   Type=Echo Request, ID=256, Seq No=4096
192.1.1.2      192.1.1.1      ICMP   Type=Echo Reply, ID=256, Seq No=4096
```

```
Address Resolution Protocol
    Hardware Type: 1 (Ethernet)
    Protocol Type: 800
    Hardware Address Length: 6
    Protocol Address Length: 4
    Operations: ARP Request
    Source Hardware Address: 00-60-97-9B-9E-07
    IP Source Address: 192.1.1.1
    Destination Hardware Address: 00-00-00-00-00-00
    IP Destination Address: 192.1.1.40
```

DNS messages



```
Source Address Dest Address   Layer Summary
This station   Broadcast      ARP   Op=ARP Request, Src IP=192.1.1.1, Dest IP=192.1.1.40
0060979B9B3F   This station   ARP   Op=ARP Response, Src IP=192.1.1.40, Dest IP=192.1.1.1
192.1.1.1      192.1.1.40     DNS   Query Question:Type=A,Class=IN,Name=labcom02.curso.pt
192.1.1.40     192.1.1.1      DNS   Query Answer:Type=A,Class=IN,Name=labcom02.curso.pt
This station   Broadcast      ARP   Op=ARP Request, Src IP=192.1.1.1, Dest IP=192.1.1.2
00104B4E687D   This station   ARP   Op=ARP Response, Src IP=192.1.1.2, Dest IP=192.1.1.1
192.1.1.1      192.1.1.2      ICMP  Type=Echo Request, ID=256, Seq No=3328
192.1.1.2      192.1.1.1      ICMP  Type=Echo Reply, ID=256, Seq No=3328
192.1.1.1      192.1.1.2      ICMP  Type=Echo Request, ID=256, Seq No=3584
192.1.1.2      192.1.1.1      ICMP  Type=Echo Reply, ID=256, Seq No=3584
192.1.1.1      192.1.1.2      ICMP  Type=Echo Request, ID=256, Seq No=3840
192.1.1.2      192.1.1.1      ICMP  Type=Echo Reply, ID=256, Seq No=3840
192.1.1.1      192.1.1.2      ICMP  Type=Echo Request, ID=256, Seq No=4096
192.1.1.2      192.1.1.1      ICMP  Type=Echo Reply, ID=256, Seq No=4096
```

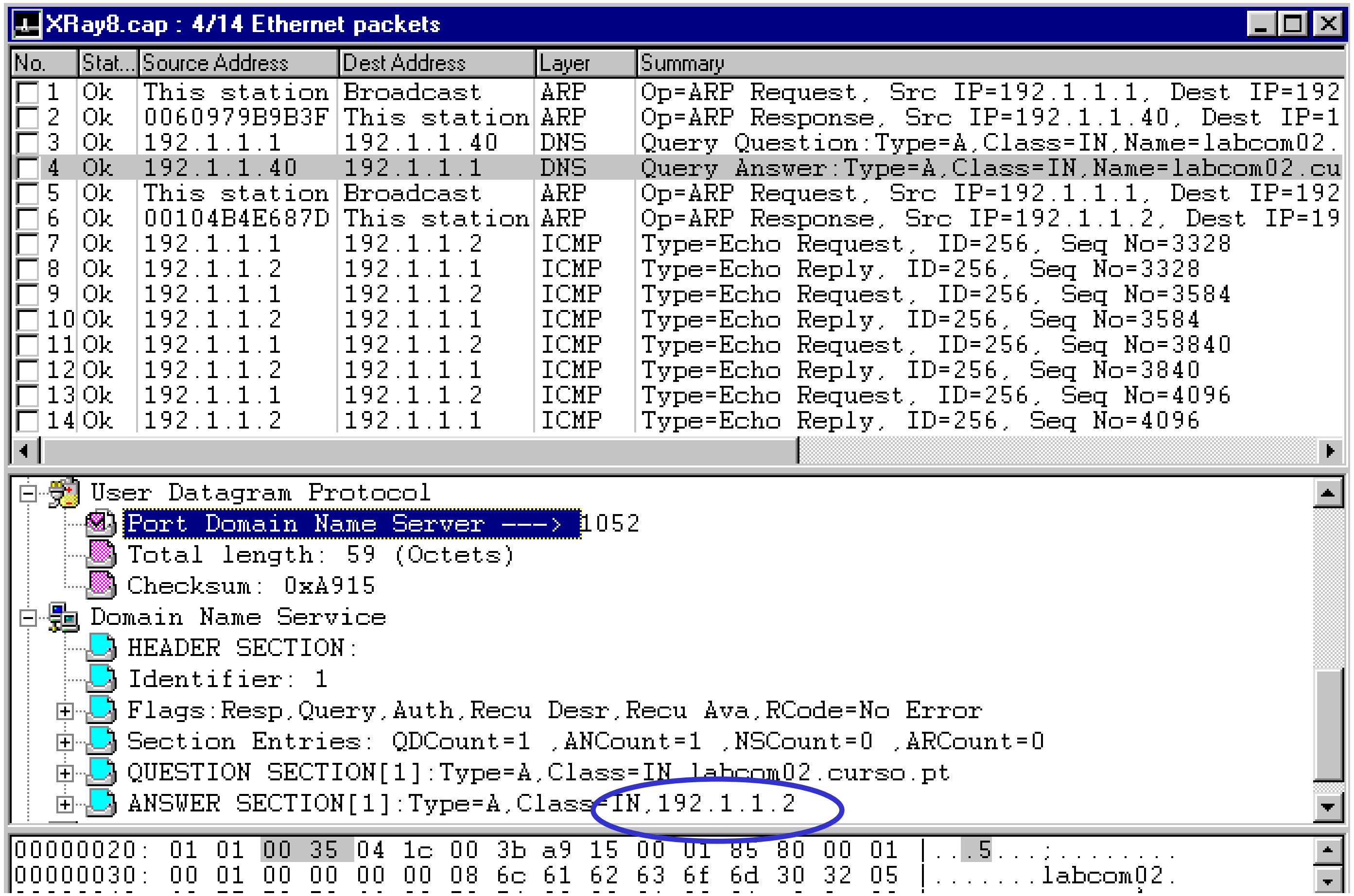
```
⊟ User Datagram Protocol
    Port 1052 ---> Domain Name Server
    Total length: 43 (Octets)
    Checksum: 0x5446
⊟ Domain Name Service
    HEADER SECTION:
    Identifier: 1
  ⊞ Flags:Req,Query,Non-Auth,Recu Desr,Recu Not Ava,RCode=No Error
  ⊞ Section Entries: QDCount=1 ,ANCount=0 ,NSCount=0 ,ARCount=0
  ⊞ QUESTION SECTION[1]:Type=A,Class=IN,labcom02.curso.pt
```

```
00000000: 00 60 97 9b 9b 3f 00 60 97 9b 9e 07 08 00 45 00 |.`...?.`......E.
00000010: 00 3f bf 00 00 00 00 11 f9 81 c0 01 01 01 c0 01 |.?¿.......À...À.
00000020: 01 28 04 1c 00 35 00 2b 54 46 00 01 01 00 00 01 |.(...5.+TF......
00000030: 00 00 00 00 00 00 08 6c 61 62 63 6f 6d 30 32 05 |.......labcom02.
```

Source port: 1052

Destination port: 53

DNS messages



XRay8.cap : 4/14 Ethernet packets

No.	Stat...	Source Address	Dest Address	Layer	Summary
□ 1	Ok	This station	Broadcast	ARP	Op=ARP Request, Src IP=192.1.1.1, Dest IP=192
□ 2	Ok	0060979B9B3F	This station	ARP	Op=ARP Response, Src IP=192.1.1.40, Dest IP=1
□ 3	Ok	192.1.1.1	192.1.1.40	DNS	Query Question:Type=A,Class=IN,Name=labcom02.
□ 4	Ok	192.1.1.40	192.1.1.1	DNS	Query Answer:Type=A,Class=IN,Name=labcom02.cu
□ 5	Ok	This station	Broadcast	ARP	Op=ARP Request, Src IP=192.1.1.1, Dest IP=192
□ 6	Ok	00104B4E687D	This station	ARP	Op=ARP Response, Src IP=192.1.1.2, Dest IP=19
□ 7	Ok	192.1.1.1	192.1.1.2	ICMP	Type=Echo Request, ID=256, Seq No=3328
□ 8	Ok	192.1.1.2	192.1.1.1	ICMP	Type=Echo Reply, ID=256, Seq No=3328
□ 9	Ok	192.1.1.1	192.1.1.2	ICMP	Type=Echo Request, ID=256, Seq No=3584
□ 10	Ok	192.1.1.2	192.1.1.1	ICMP	Type=Echo Reply, ID=256, Seq No=3584
□ 11	Ok	192.1.1.1	192.1.1.2	ICMP	Type=Echo Request, ID=256, Seq No=3840
□ 12	Ok	192.1.1.2	192.1.1.1	ICMP	Type=Echo Reply, ID=256, Seq No=3840
□ 13	Ok	192.1.1.1	192.1.1.2	ICMP	Type=Echo Request, ID=256, Seq No=4096
□ 14	Ok	192.1.1.2	192.1.1.1	ICMP	Type=Echo Reply, ID=256, Seq No=4096

```
⊟ User Datagram Protocol
     Port Domain Name Server ---> 1052
     Total length: 59 (Octets)
     Checksum: 0xA915
⊟ Domain Name Service
     HEADER SECTION:
     Identifier: 1
  ⊞  Flags:Resp,Query,Auth,Recu Desr,Recu Ava,RCode=No Error
  ⊞  Section Entries: QDCount=1 ,ANCount=1 ,NSCount=0 ,ARCount=0
  ⊞  QUESTION SECTION[1]:Type=A,Class=IN labcom02.curso.pt
  ⊞  ANSWER SECTION[1]:Type=A,Class=IN,192.1.1.2
```

```
00000020: 01 01 00 35 04 1c 00 3b a9 15 00 01 85 80 00 01 |..5...;.........
00000030: 00 01 00 00 00 00 08 6c 61 62 63 6f 6d 30 32 05 |.......labcom02.
```

HTTP

Web and HTTP

First some basic concepts

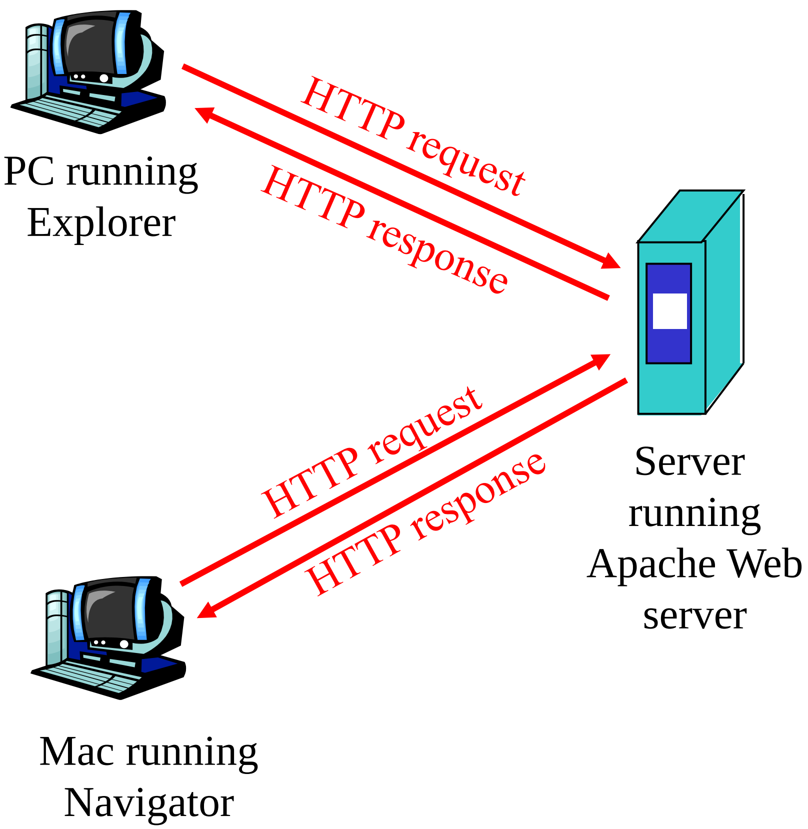
- Web page consists of objects
- Object can be HTML file, JPEG image, Java applet, audio file,…
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL
- Example URL:

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

HTTP overview

HTTP: HyperText Transfer Protocol

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, "displays" Web objects
  - *server:* Web server sends objects in response to requests
- HTTP 1.0: RFC 1945
- HTTP 1.1: RFC 2068



HTTP overview (continued)

Uses TCP:

- client initiates TCP connection (creates socket) to server, port 80
- server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed

HTTP is "stateless"

- server maintains no information about past client requests

aside

Protocols that maintain "state" are complex!

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled

HTTP connections

Nonpersistent HTTP

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

Persistent HTTP

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

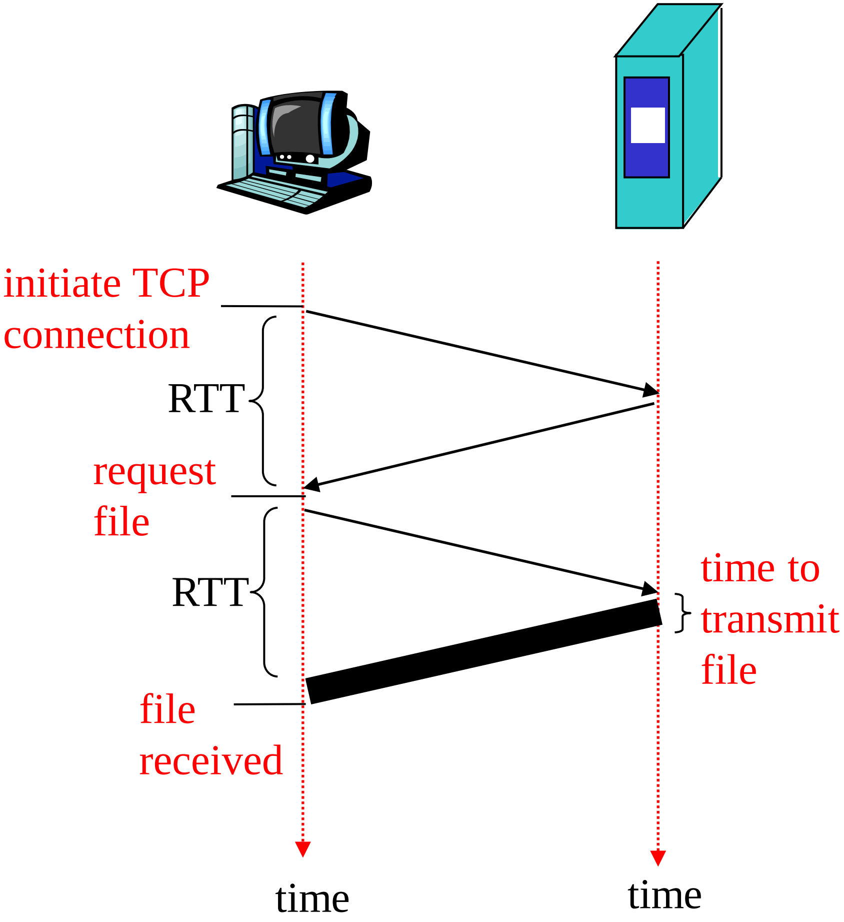
Nonpersistent HTTP: Response time

Round Trip Time (RTT:
time to send a small packet to travel from client to server and back.

Response time:

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

total = 2RTT+transmit time



Persistent HTTP

Nonpersistent HTTP issues:

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

Persistent HTTP

- server leaves connection open after sending response
- subsequent HTTP messages between same client/server sent over open connection

Persistent ***without*** pipelining:

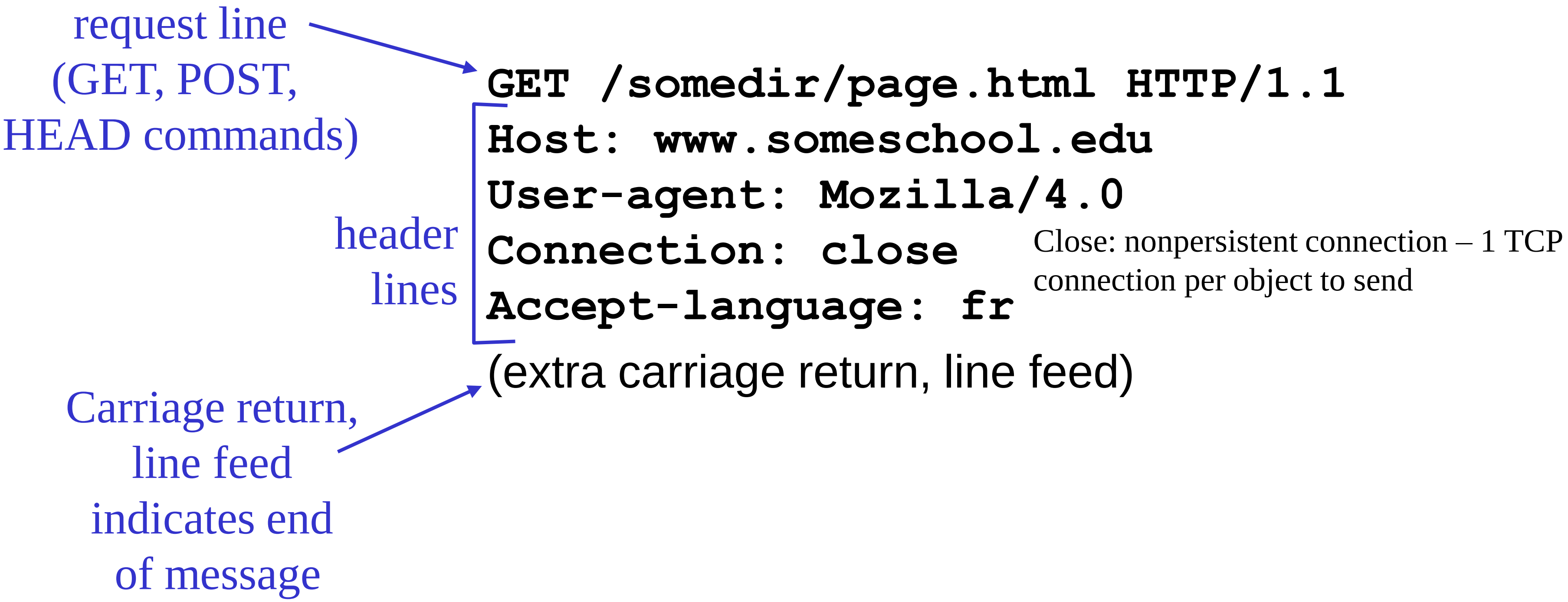
- client issues new request only when previous response has been received
- one RTT for each referenced object

Persistent ***with*** pipelining:

- default in HTTP/1.1
- client sends requests as soon as it encounters a referenced object
- as little as one RTT for all the referenced objects

HTTP request message

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



request line (GET, POST, HEAD commands)

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

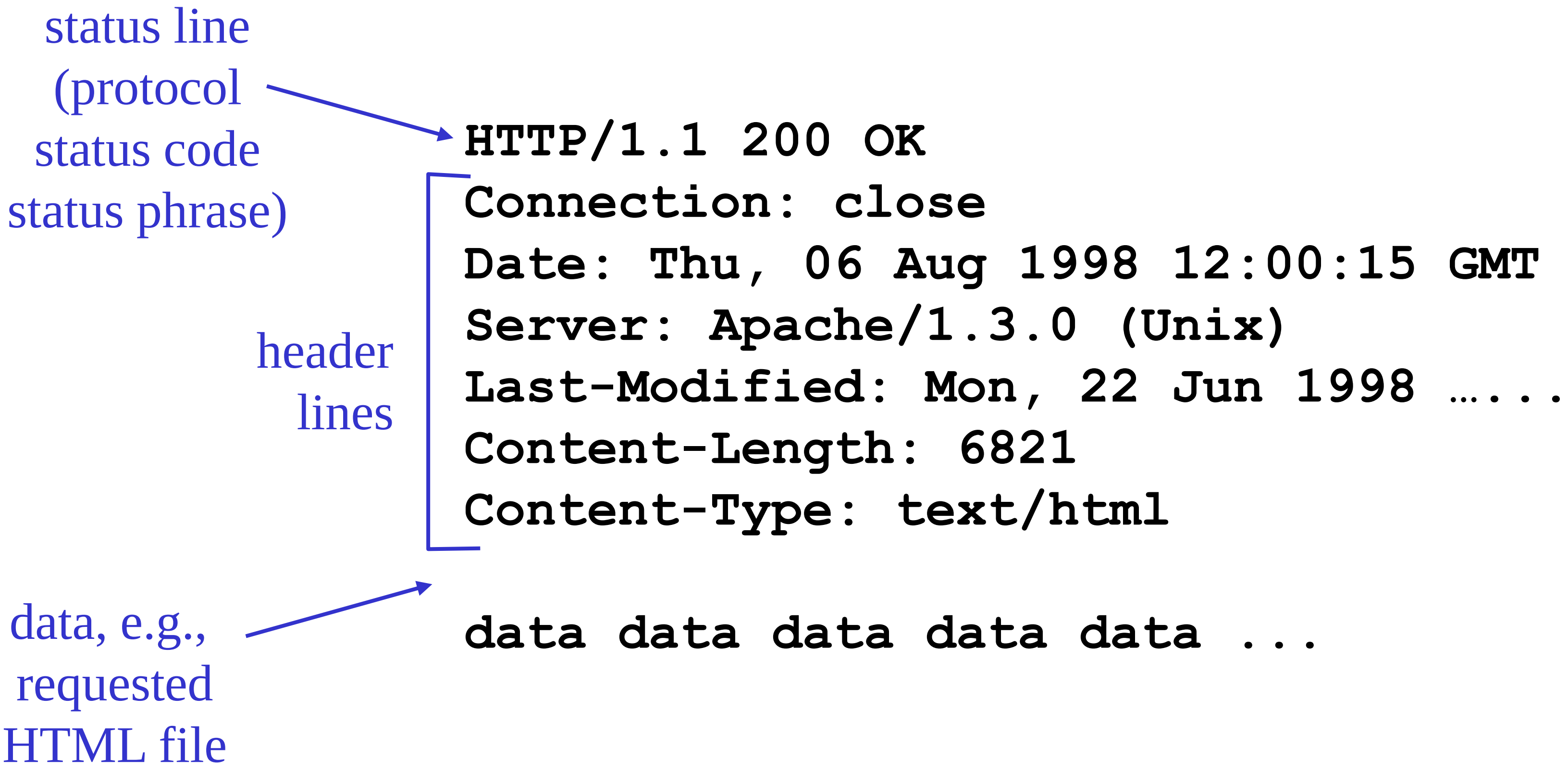
(extra carriage return, line feed)

header lines

Close: nonpersistent connection – 1 TCP connection per object to send

Carriage return, line feed indicates end of message

HTTP response message



status line (protocol status code status phrase)

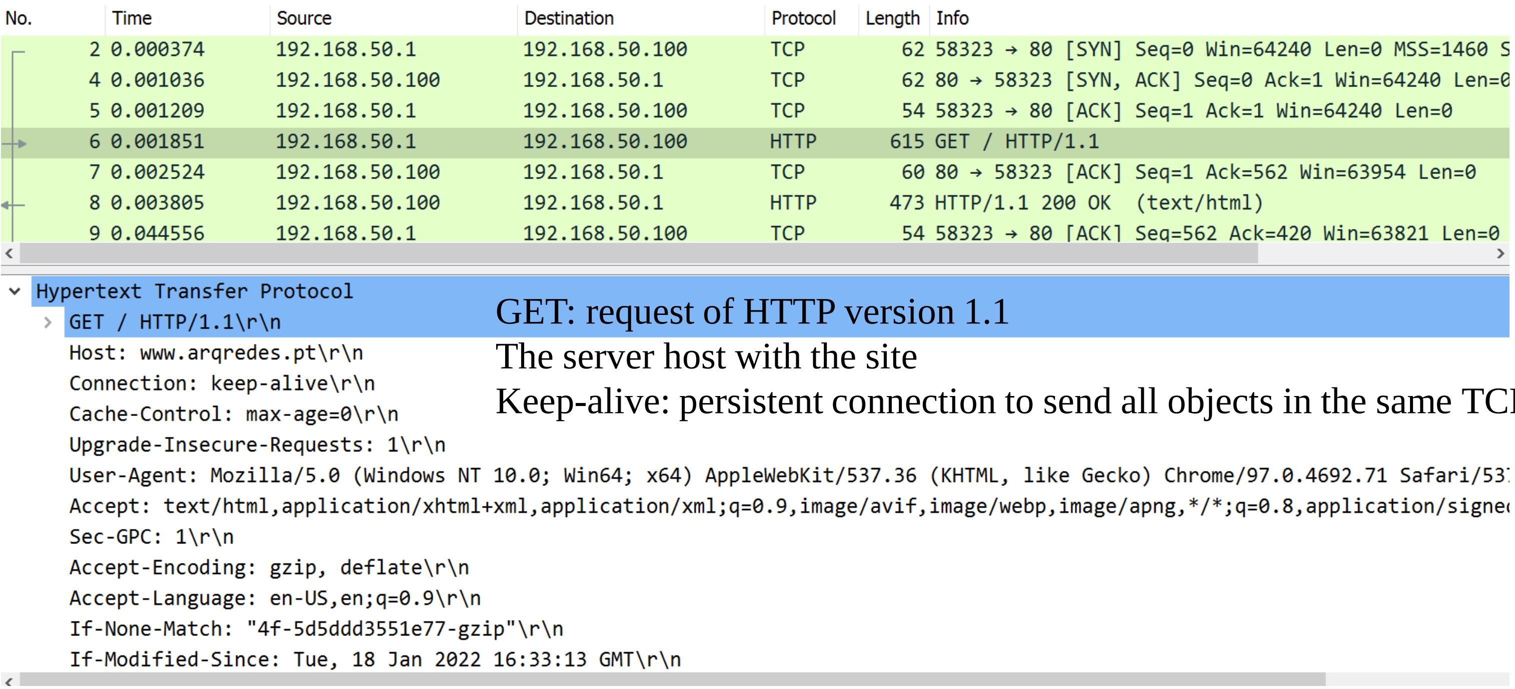
header lines

data, e.g., requested HTML file

```
HTTP/1.1 200 OK
Connection: close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```

HTTP Request - starts with TCP connection



```
No.   Time       Source          Destination     Protocol  Length  Info
  2 0.000374   192.168.50.1    192.168.50.100  TCP         62 58323 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S
  4 0.001036   192.168.50.100  192.168.50.1    TCP         62 80 → 58323 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
  5 0.001209   192.168.50.1    192.168.50.100  TCP         54 58323 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
  6 0.001851   192.168.50.1    192.168.50.100  HTTP       615 GET / HTTP/1.1
  7 0.002524   192.168.50.100  192.168.50.1    TCP         60 80 → 58323 [ACK] Seq=1 Ack=562 Win=63954 Len=0
  8 0.003805   192.168.50.100  192.168.50.1    HTTP       473 HTTP/1.1 200 OK  (text/html)
  9 0.044556   192.168.50.1    192.168.50.100  TCP         54 58323 → 80 [ACK] Seq=562 Ack=420 Win=63821 Len=0
```

```
Hypertext Transfer Protocol
  > GET / HTTP/1.1\r\n
    Host: www.arqredes.pt\r\n
    Connection: keep-alive\r\n
    Cache-Control: max-age=0\r\n
    Upgrade-Insecure-Requests: 1\r\n
    User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/97.0.4692.71 Safari/53
    Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signe
    Sec-GPC: 1\r\n
    Accept-Encoding: gzip, deflate\r\n
    Accept-Language: en-US,en;q=0.9\r\n
    If-None-Match: "4f-5d5ddd3551e77-gzip"\r\n
    If-Modified-Since: Tue, 18 Jan 2022 16:33:13 GMT\r\n
```

GET: request of HTTP version 1.1

The server host with the site

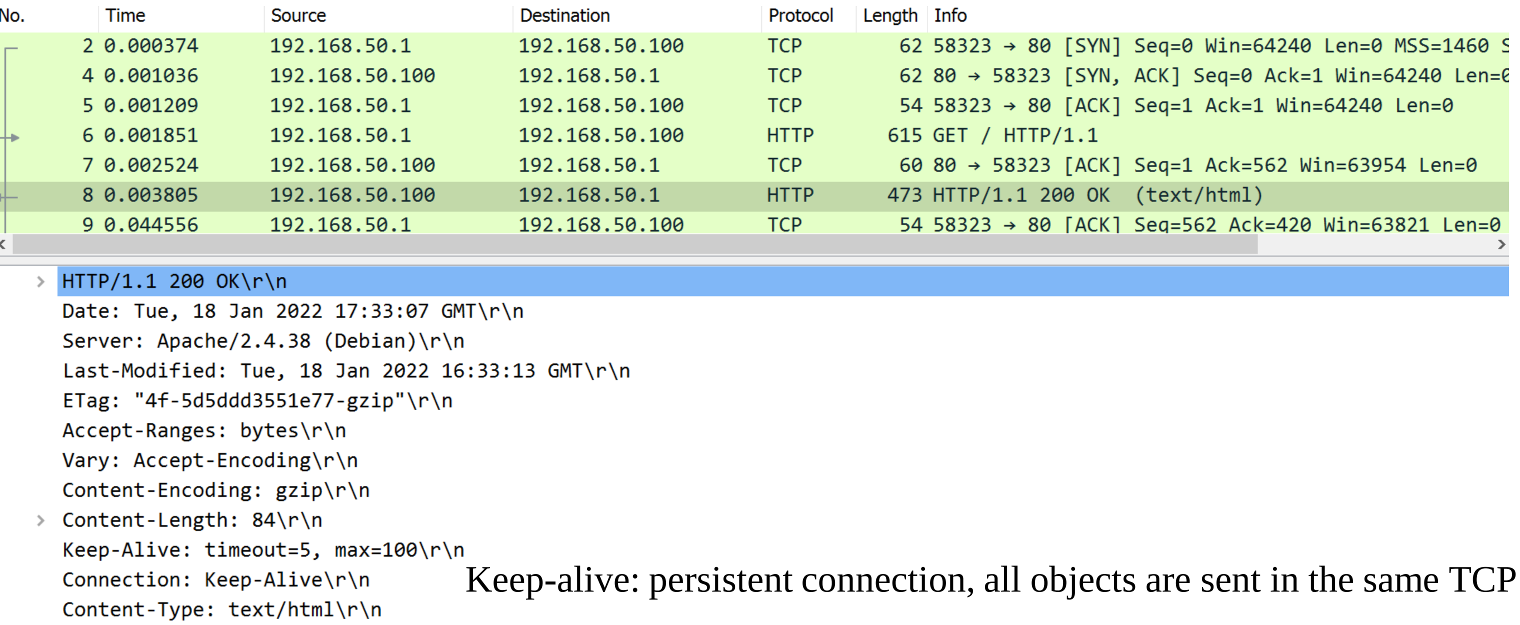
Keep-alive: persistent connection to send all objects in the same TCP connection

TCP 3-way handshake to initiate the HTTP connect in port 80

GET: request of HTTP version 1.1

200 OK: answer with data transmission

HTTP Request - answers with data, ACKed with TCP



```
No.   Time      Source          Destination     Protocol  Length Info
    2 0.000374  192.168.50.1    192.168.50.100  TCP           62 58323 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S
    4 0.001036  192.168.50.100  192.168.50.1    TCP           62 80 → 58323 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
    5 0.001209  192.168.50.1    192.168.50.100  TCP           54 58323 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
    6 0.001851  192.168.50.1    192.168.50.100  HTTP         615 GET / HTTP/1.1
    7 0.002524  192.168.50.100  192.168.50.1    TCP           60 80 → 58323 [ACK] Seq=1 Ack=562 Win=63954 Len=0
    8 0.003805  192.168.50.100  192.168.50.1    HTTP         473 HTTP/1.1 200 OK  (text/html)
    9 0.044556  192.168.50.1    192.168.50.100  TCP           54 58323 → 80 [ACK] Seq=562 Ack=420 Win=63821 Len=0
```

```
> HTTP/1.1 200 OK\r\n
  Date: Tue, 18 Jan 2022 17:33:07 GMT\r\n
  Server: Apache/2.4.38 (Debian)\r\n
  Last-Modified: Tue, 18 Jan 2022 16:33:13 GMT\r\n
  ETag: "4f-5d5ddd3551e77-gzip"\r\n
  Accept-Ranges: bytes\r\n
  Vary: Accept-Encoding\r\n
  Content-Encoding: gzip\r\n
> Content-Length: 84\r\n
  Keep-Alive: timeout=5, max=100\r\n
  Connection: Keep-Alive\r\n
  Content-Type: text/html\r\n
```

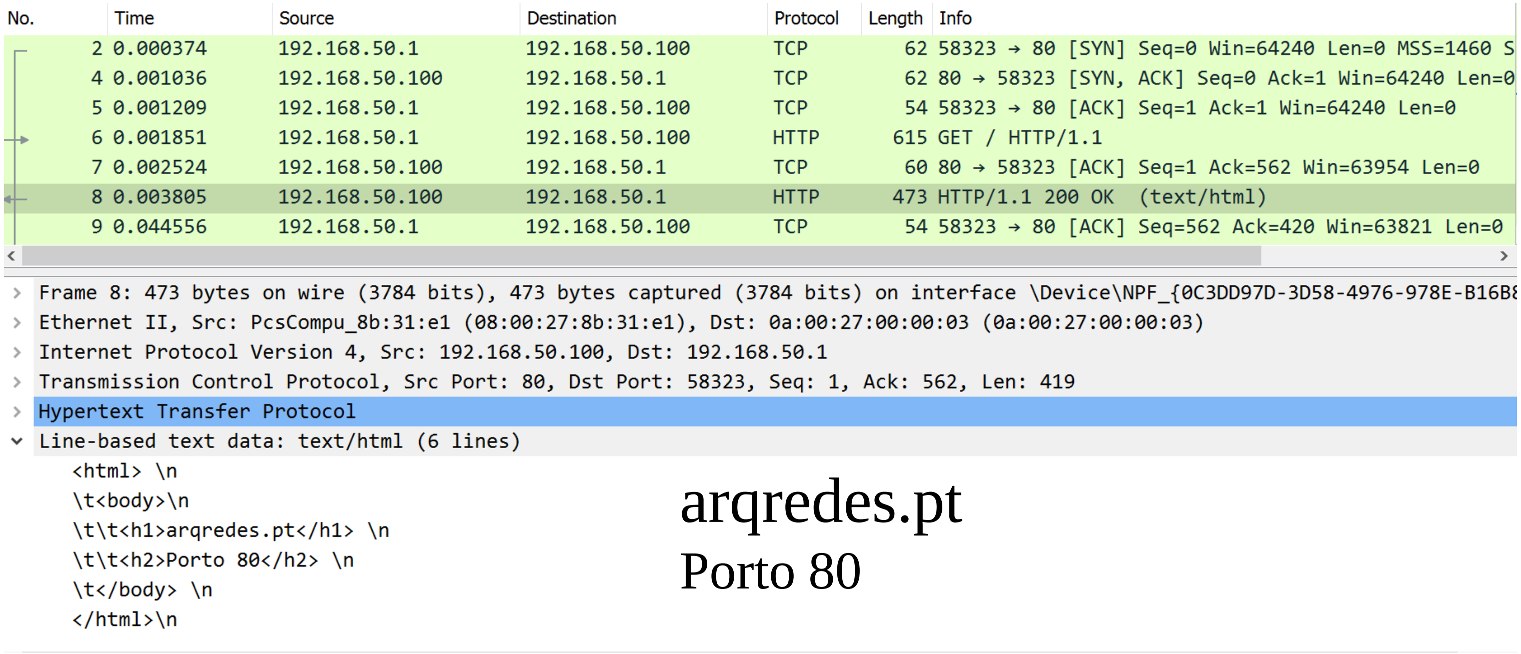
TCP 3-way handshake to initiate the HTTP connect in port 80

GET: request of HTTP version 1.1

200 OK: answer sending the data

Keep-alive: persistent connection, all objects are sent in the same TCP connection

HTTP: The content of the page



```
No.      Time       Source           Destination      Protocol  Length  Info
       2 0.000374   192.168.50.1     192.168.50.100   TCP          62 58323 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 S
       4 0.001036   192.168.50.100   192.168.50.1     TCP          62 80 → 58323 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0
       5 0.001209   192.168.50.1     192.168.50.100   TCP          54 58323 → 80 [ACK] Seq=1 Ack=1 Win=64240 Len=0
       6 0.001851   192.168.50.1     192.168.50.100   HTTP        615 GET / HTTP/1.1
       7 0.002524   192.168.50.100   192.168.50.1     TCP          60 80 → 58323 [ACK] Seq=1 Ack=562 Win=63954 Len=0
       8 0.003805   192.168.50.100   192.168.50.1     HTTP        473 HTTP/1.1 200 OK  (text/html)
       9 0.044556   192.168.50.1     192.168.50.100   TCP          54 58323 → 80 [ACK] Seq=562 Ack=420 Win=63821 Len=0
```

```
> Frame 8: 473 bytes on wire (3784 bits), 473 bytes captured (3784 bits) on interface \Device\NPF_{0C3DD97D-3D58-4976-978E-B16B8
> Ethernet II, Src: PcsCompu_8b:31:e1 (08:00:27:8b:31:e1), Dst: 0a:00:27:00:00:03 (0a:00:27:00:00:03)
> Internet Protocol Version 4, Src: 192.168.50.100, Dst: 192.168.50.1
> Transmission Control Protocol, Src Port: 80, Dst Port: 58323, Seq: 1, Ack: 562, Len: 419
> Hypertext Transfer Protocol
v Line-based text data: text/html (6 lines)
     <html> \n
     \t<body>\n
     \t\t<h1>arqredes.pt</h1> \n
     \t\t<h2>Porto 80</h2> \n
     \t</body> \n
     </html>\n
```

arqredes.pt

Porto 80