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, 2021/2022

TFTP

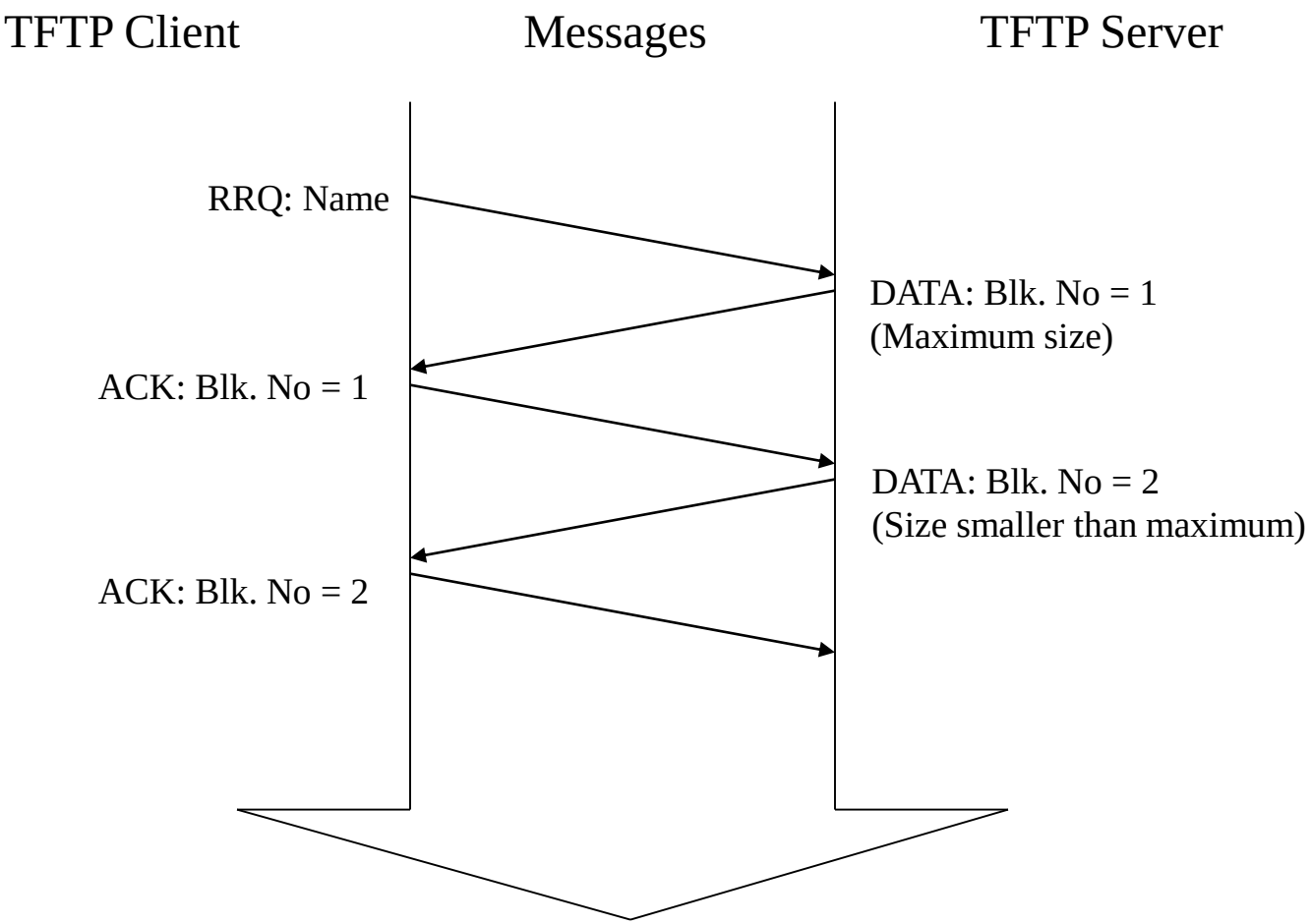
Trivial File Transfer Protocol (TFTP)

- 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.

Trivial File Transfer Protocol (TFTP)

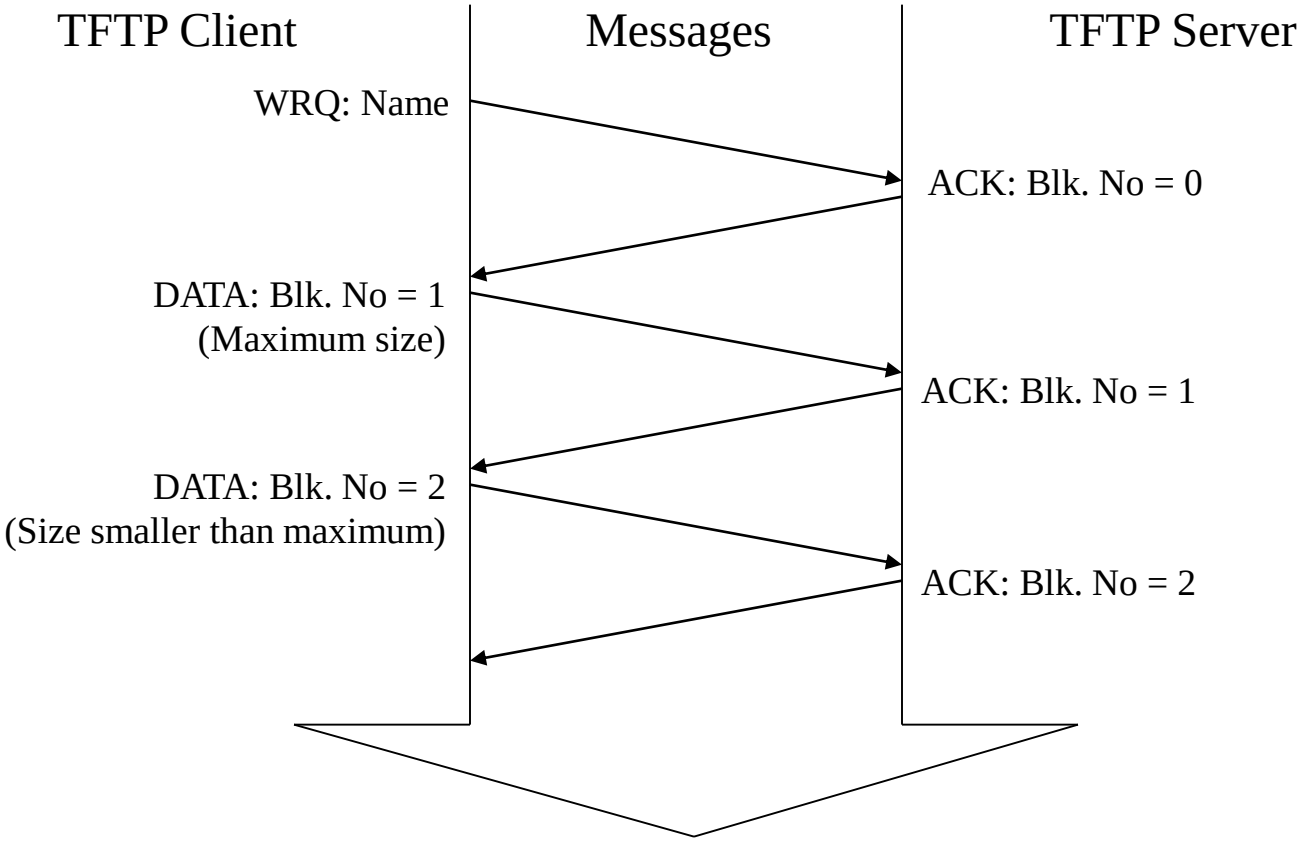
- Basic file transfer service (IETF RFC 1350)
  - It does not allow to list server files
  - It does not support authentication
- TFTP runs over UDP
  - The initial client message is sent to server port number 69
  - TFTP server answers from another locally selected port number
  - The following messages 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



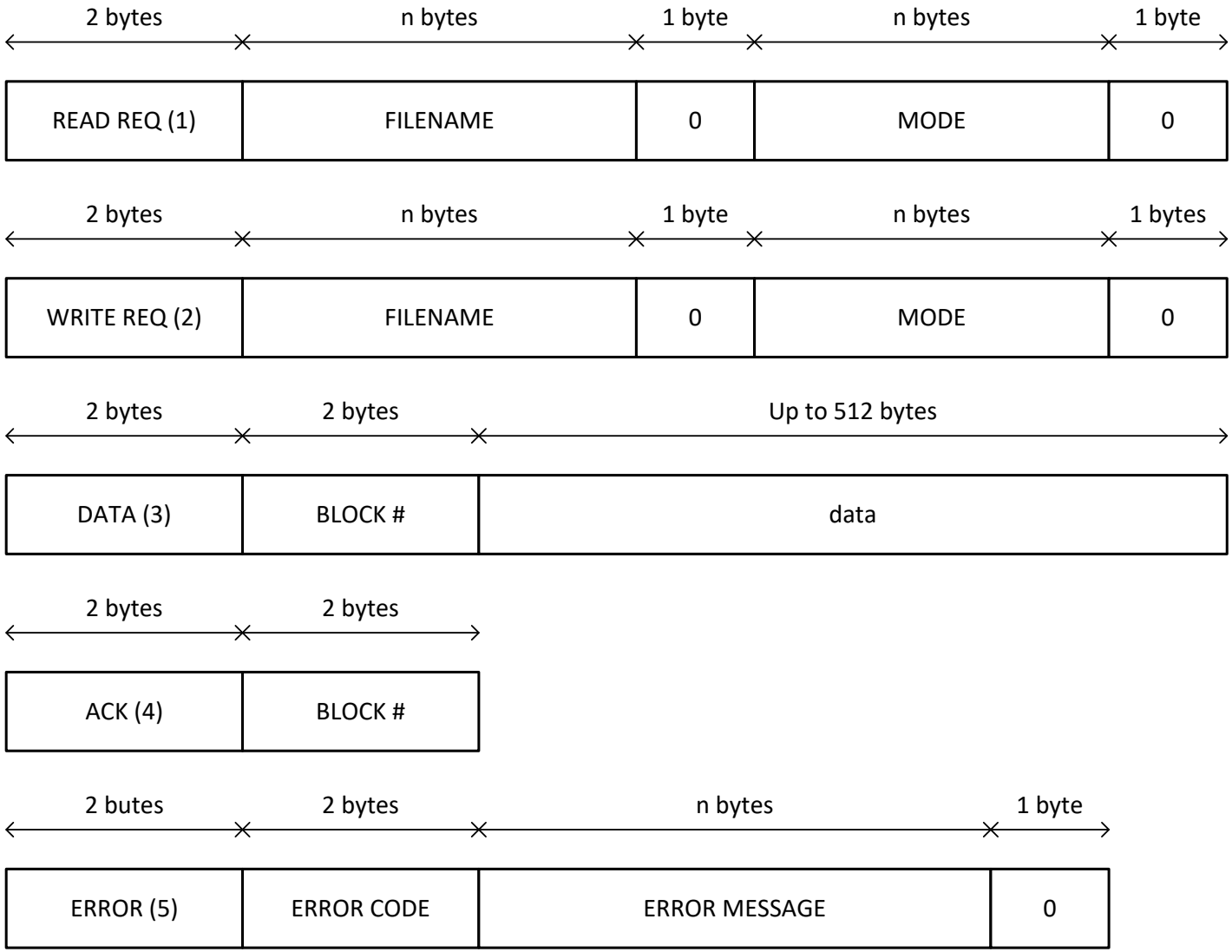
Client detects the last block of DATA when the data block size is lower than the maximum size.

Write Request session



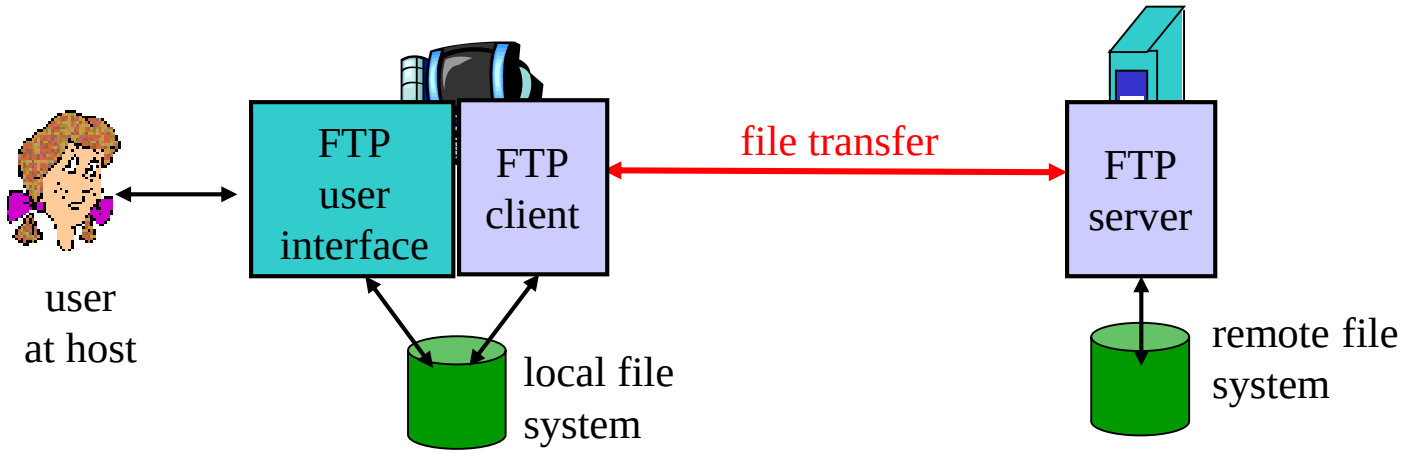
Server detects the last block of DATA when the data block size is lower than the maximum size.

Messages format



FTP

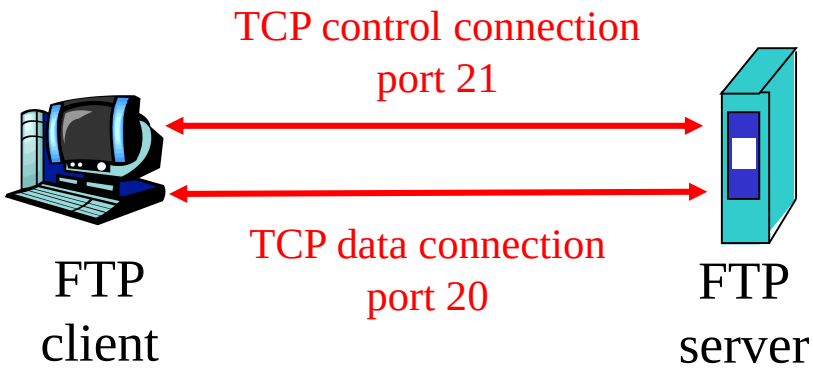
FTP: the file transfer protocol



- 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
- ftp server ports: 21 and 20

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 2^{nd} TCP connection (for file transfer) to client
- after transferring one file, server closes data connection.



- 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:

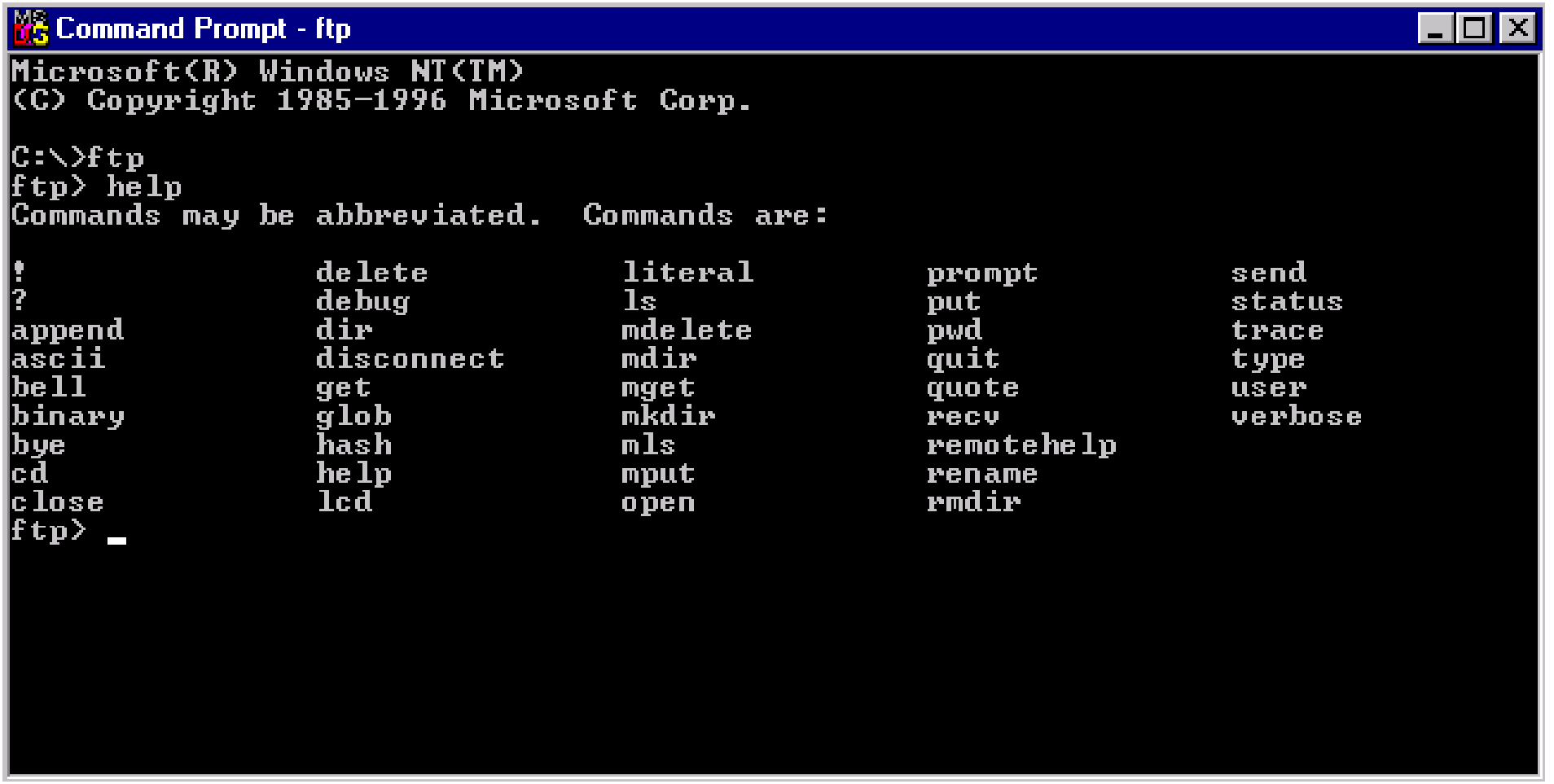
- **USER *username***
- **PASS *password***
- **LIST** (return list of files in current directory)
- **RETR filename** (retrieves file from server)
- **STOR filename** (stores file onto server)

Sample return codes

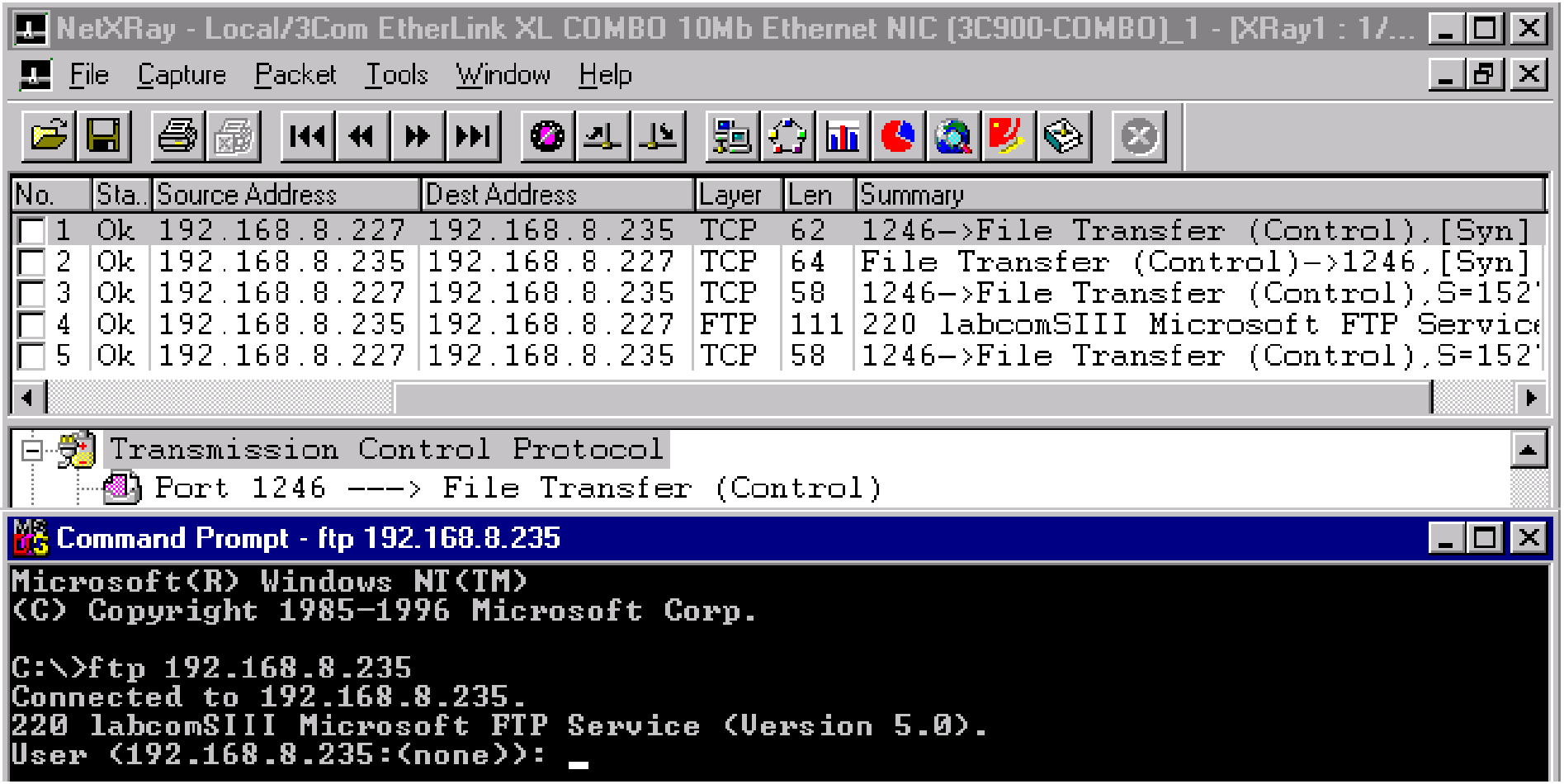
- status code and phrase (as in HTTP):

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

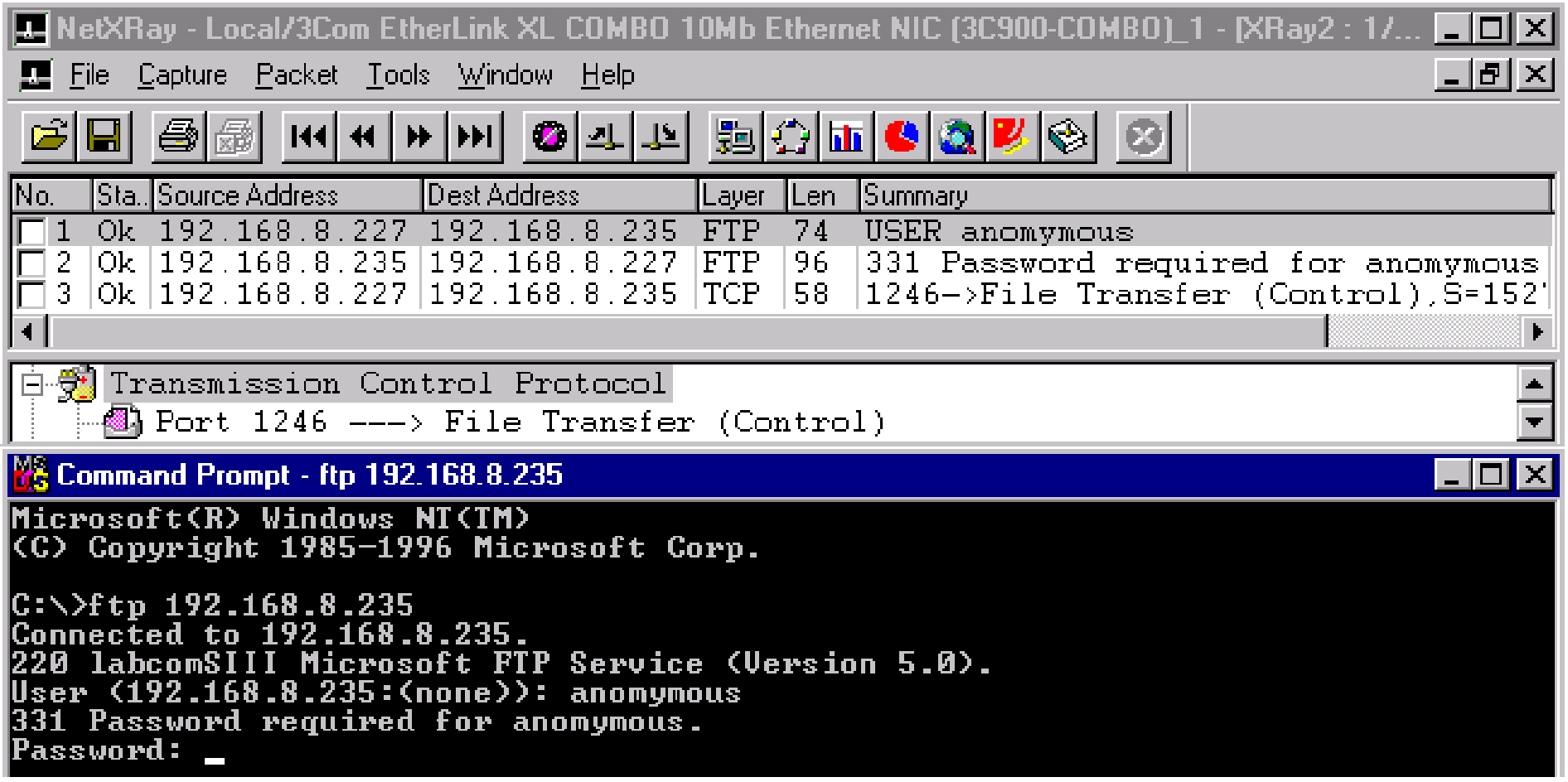
FTP client: user commands



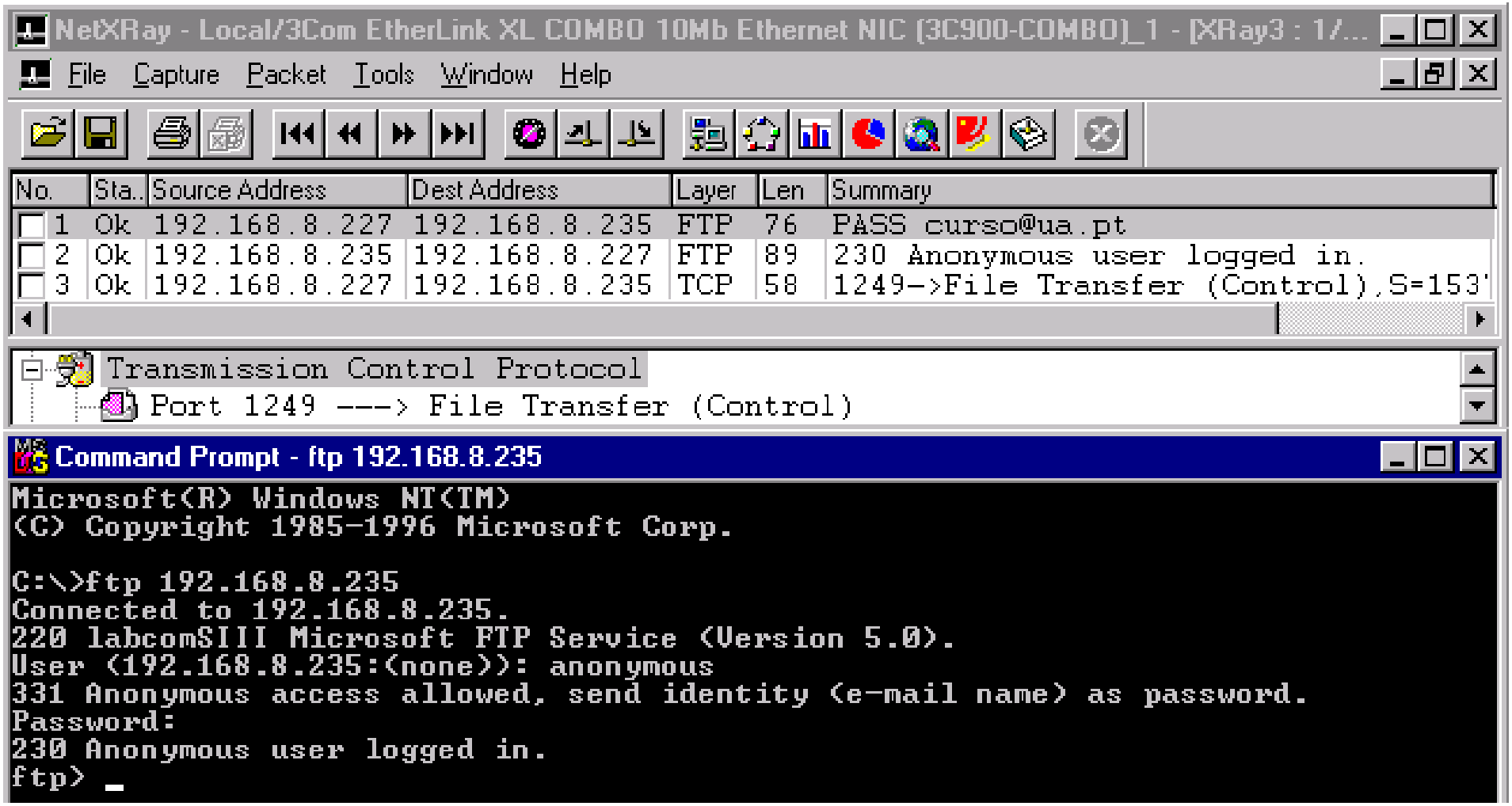
Initial connection to FTP server



Introduction of username



Introduction of password



Introduction of DIR command

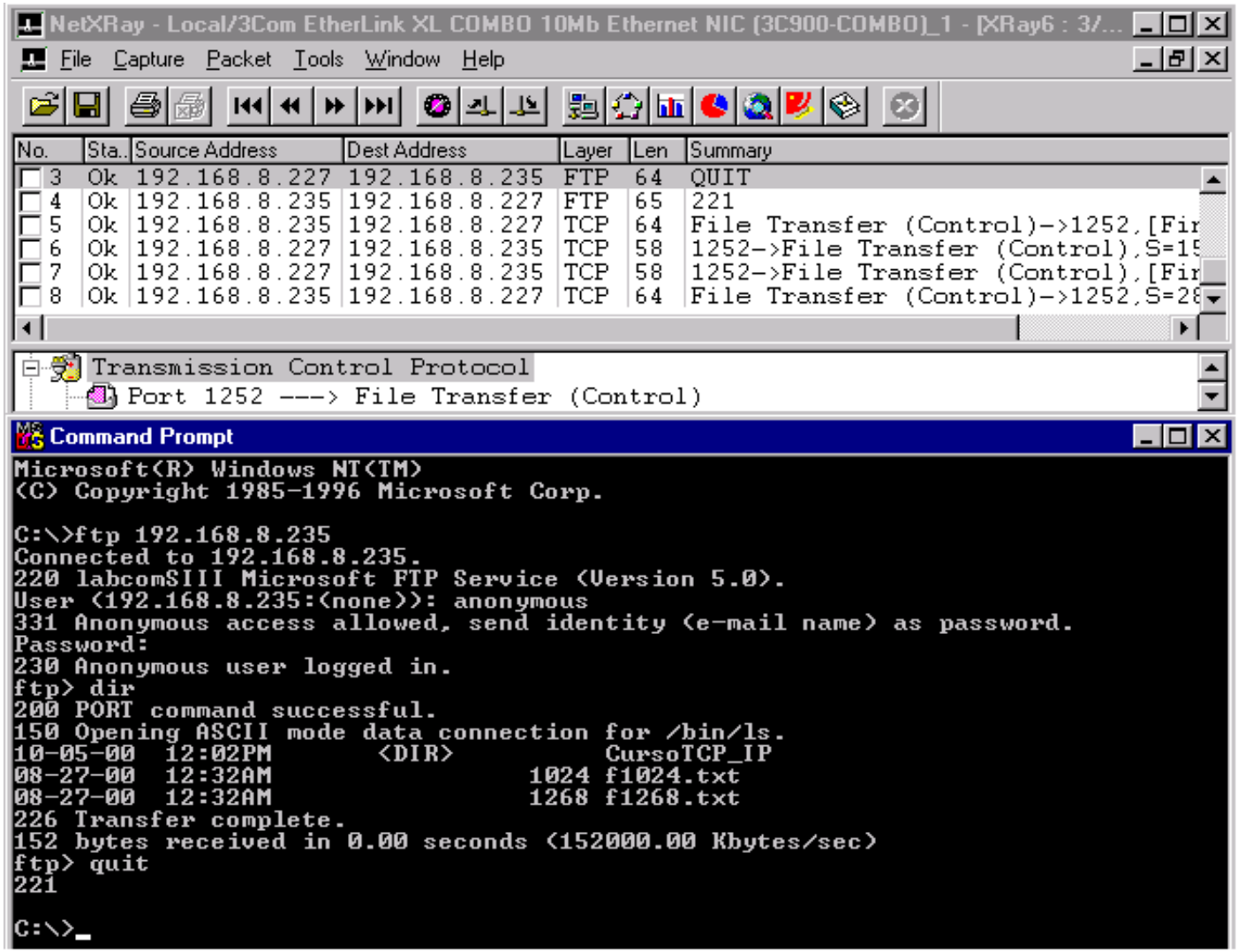
No.	Sta..	Source Address	Dest Address	Layer	Len	Summary
1	Ok	192.168.8.227	192.168.8.235	FTP	84	PORT 192,168,8,227,4,229
2	Ok	192.168.8.235	192.168.8.227	FTP	88	200 PORT command successful.
3	Ok	192.168.8.227	192.168.8.235	FTP	64	LIST
4	Ok	192.168.8.235	192.168.8.227	FTP	111	150 Opening ASCII mode data 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,S
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),S
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,S
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

```
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)



DNS

DNS: Domain Name System

People: many identifiers:

- SSN, name, passport #

Internet hosts, routers:

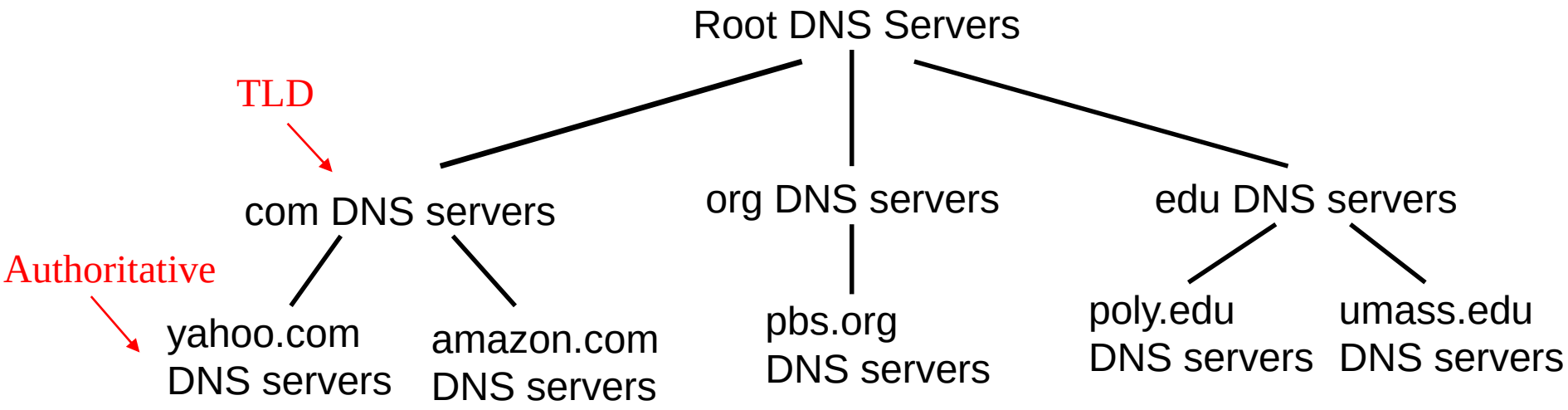
- IP address (32 bit) - used for addressing datagrams
- "name" (for example, www.yahoo.com) - used by humans

DNS: provides a mapping between IP addresses and names

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:

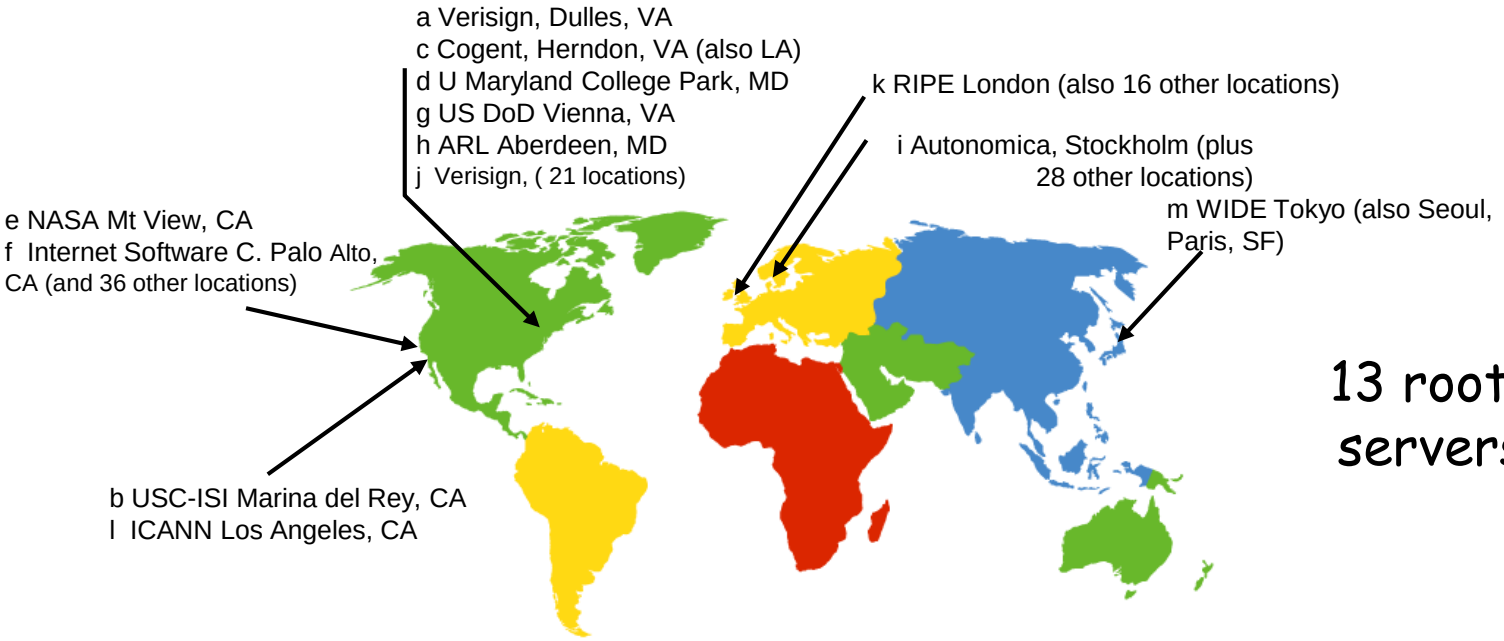
- client queries a root server to find 'com' DNS server
- client queries 'com' DNS server to get 'amazon.com' DNS server
- client queries 'amazon.com' DNS server to get IP address for www.amazon.com

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



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

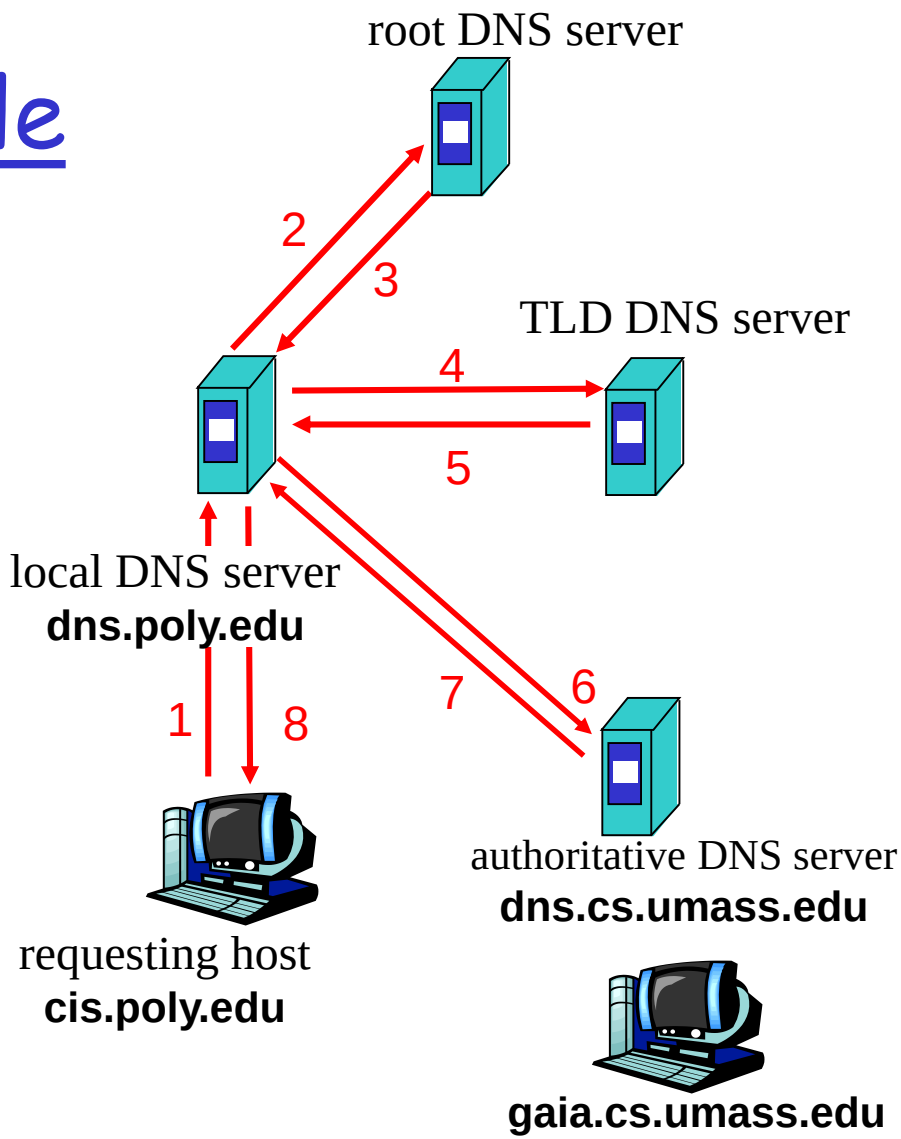
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one.
  - also called "default Name Server"
- when host makes DNS query, query is sent to its local DNS server
  - acts as proxy, forwards query into hierarchy

DNS name resolution example

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

iterated query:

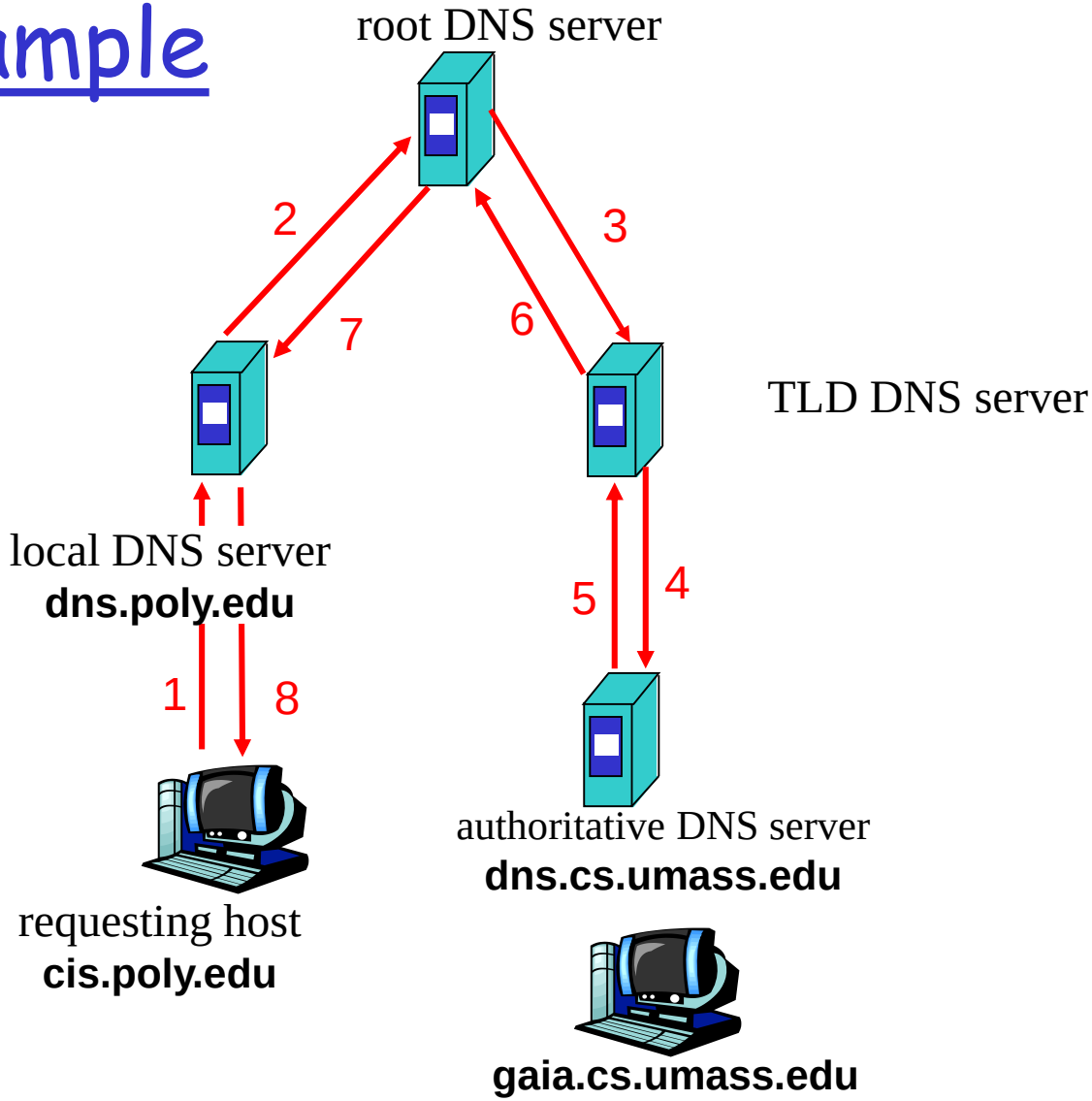
- contacted server replies with name of server to contact
- "I don't know this name, but ask this server"



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)`

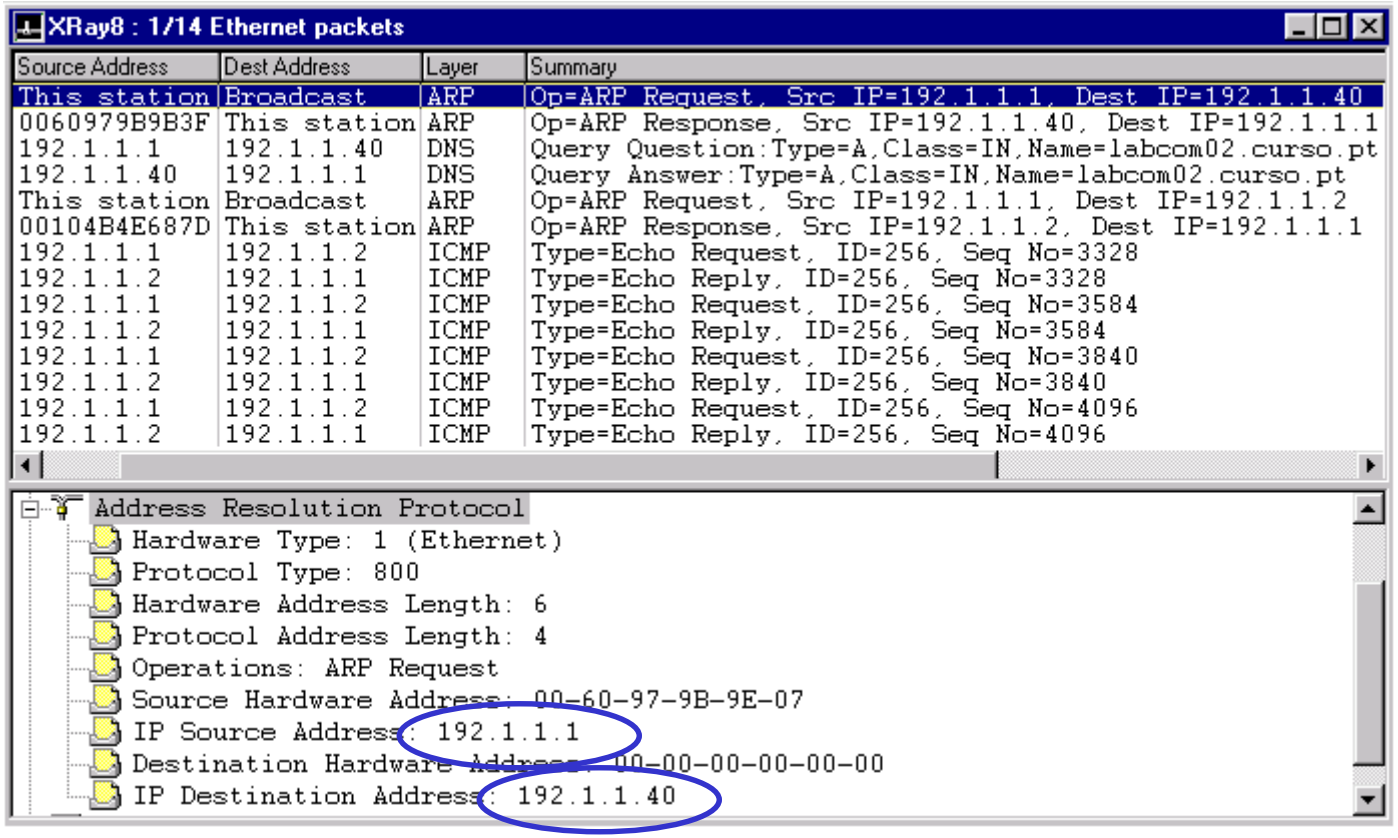
- Type=A
  - **name** is hostname
  - **value** is IP address
  - e.g. (relay1.bar.foo.com, 145.37.93.126, A)
- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain
  - e.g. (foo.com, dns.foo.com, NS)
- Type=CNAME
  - **name** is alias name 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

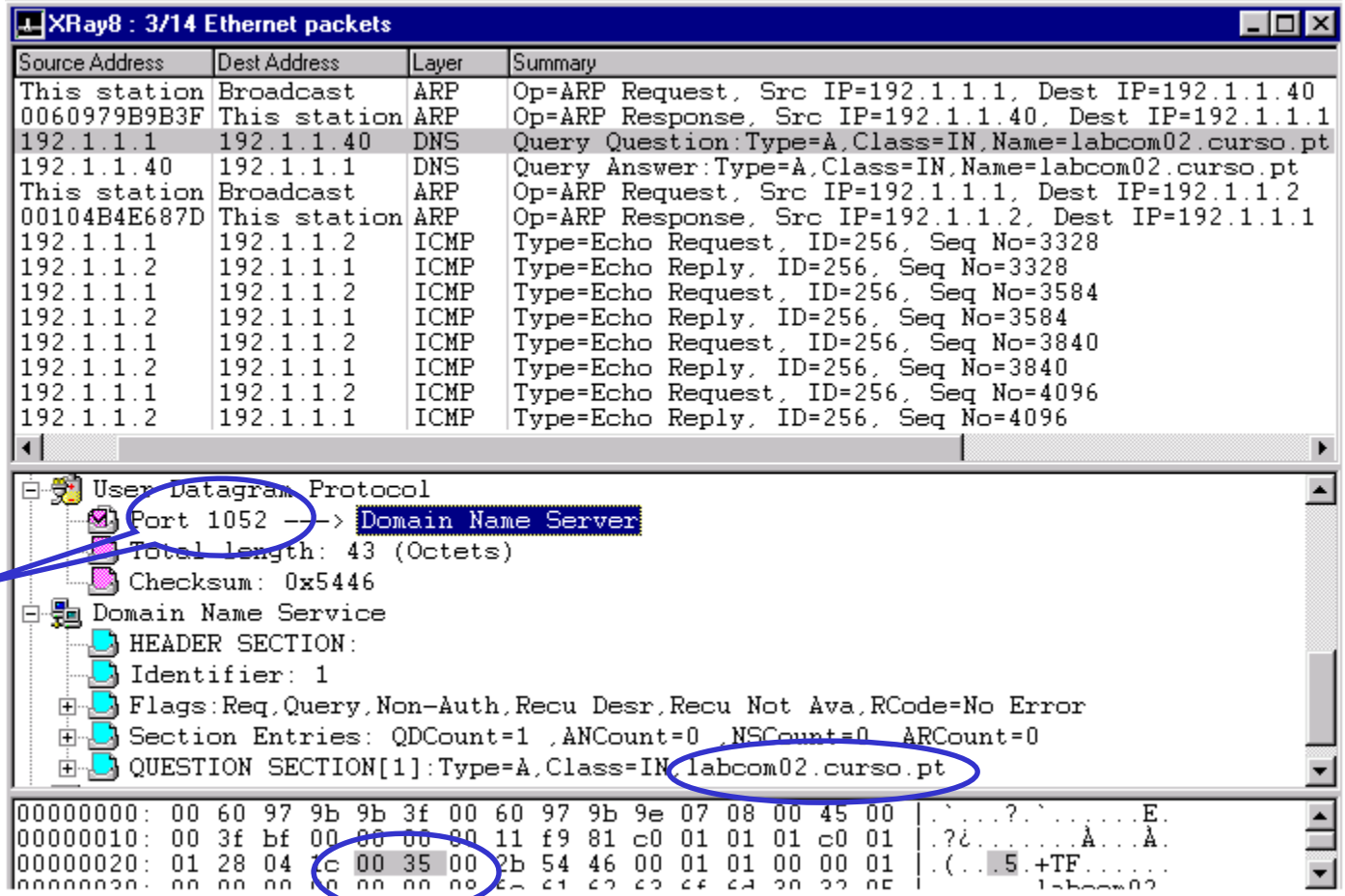
host: 192.1.1.1

DNS: 192.1.1.40

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



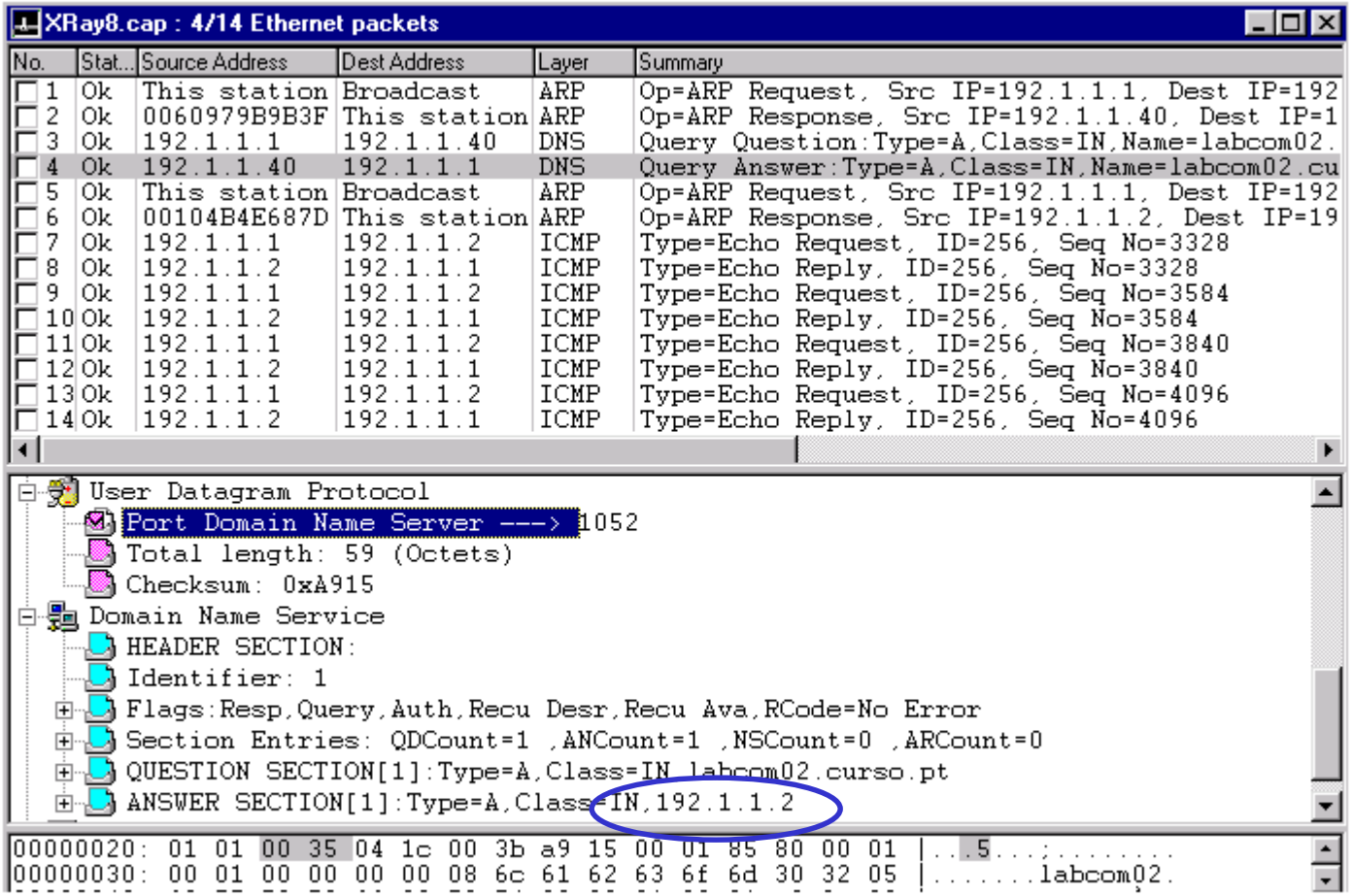
DNS messages



Source port: 1052

Destination port: 53

DNS messages



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:

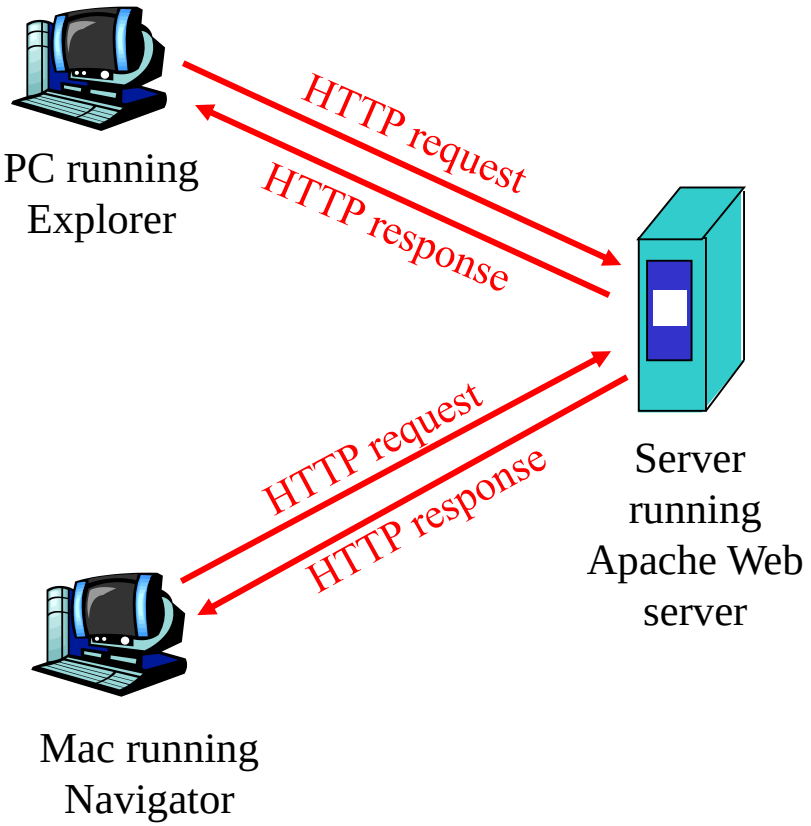
`www.someschool.edu/someDept/pic.gif`

host name | path name

HTTP overview

HTTP: HyperText Transfer Protocol

- Web's application layer protocol
- client/server model
  - *client:* browser that requests, receives, "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

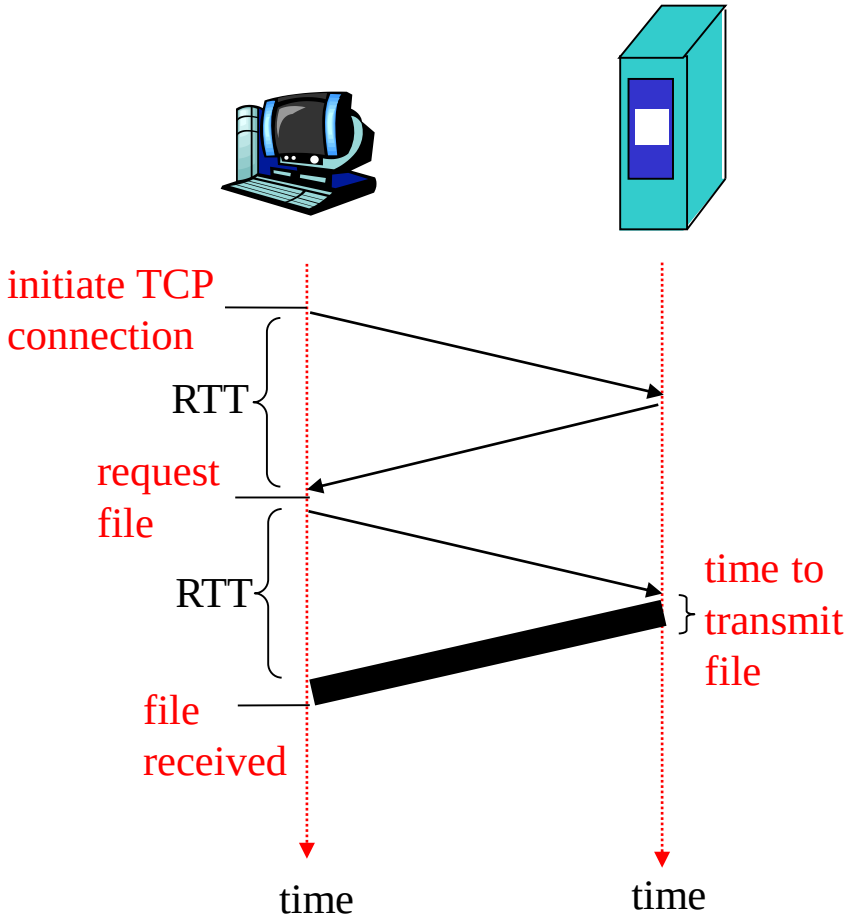
Nonpersistent HTTP: Response time

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:

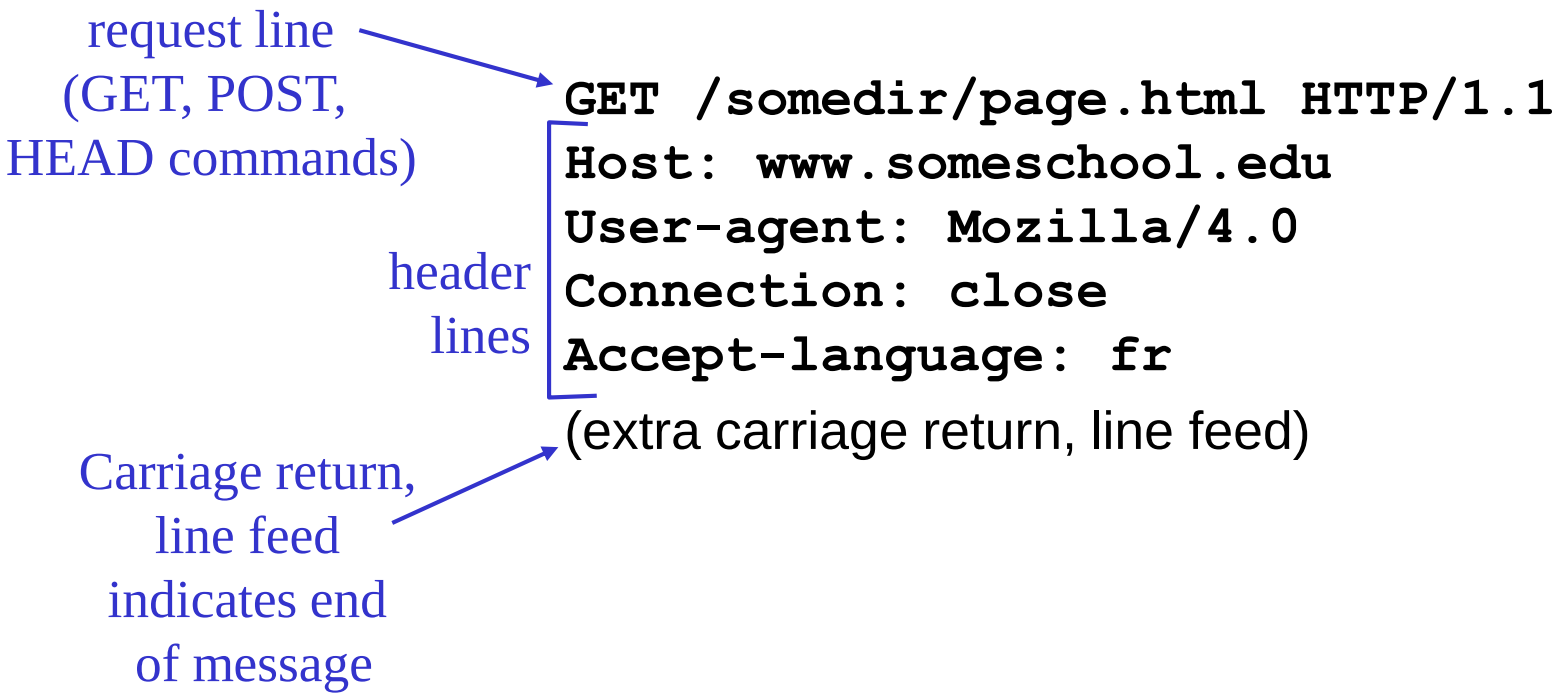
- client issues new request only when previous response has been received
- one RTT for each referenced object

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)



HTTP response message