Chapter 2 Application Layer

The **application layer** enables the user, whether human or software, to access the network. It provides **user interfaces** and support for services such as **electronic mail**, **remote file access** and **transfer**, **shared database management**, and other types of distributed information services.

- **Mail Services**
- **Network Virtual Terminal**
- **Directory Services**
- **File Transfer, Access and Management (FTAM)**

FTP, HTTP, DNS, SMPT, Telnet

Application architectures

- Client-server
- Peer-to-peer (P2P)
- Hybrid of client-server and P2P

Client-server architectures

CLIENT-SERVER PARADIGM

- **Server:-** A *server* is a program running on the remote machine providing service to the clients. (Provider , infinite program)
    - always-on host
    - permanent IP address
    - server farms for scaling

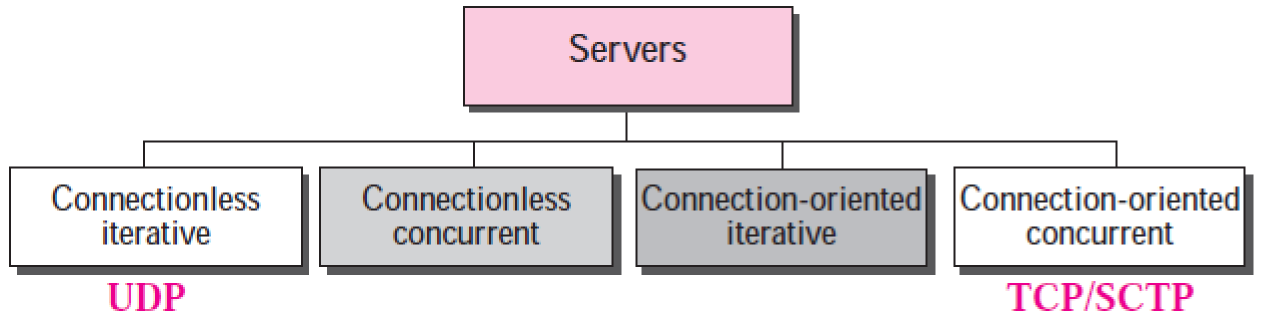
- Client :- A *client* is a program running on the local machine requesting service from a server. (requestor , finite program)
    - communicate with server
    - may be intermittently connected
    - may have dynamic IP addresses
    - do not communicate directly with each other

CLIENT-SERVER PARADIGM

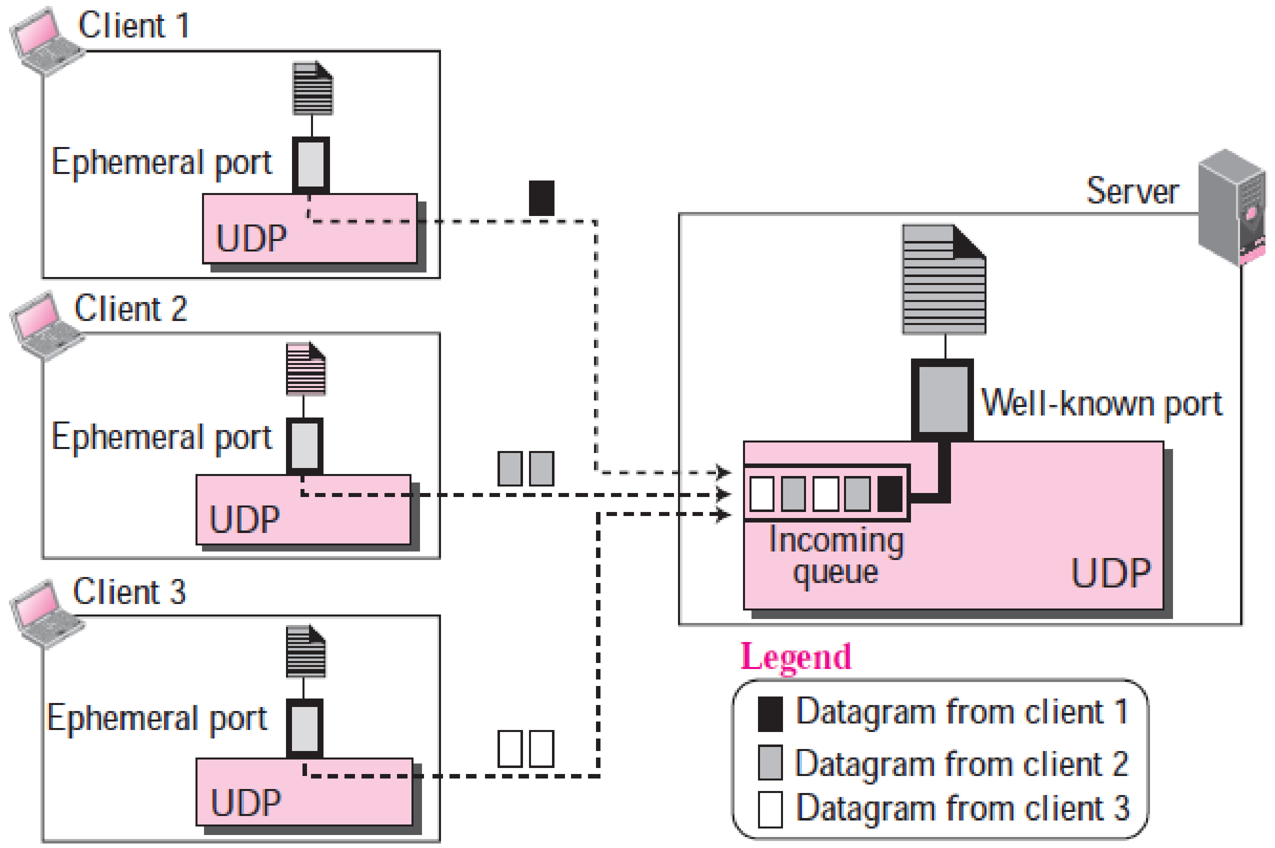
Concurrency

- Both clients and servers can run in concurrent mode.

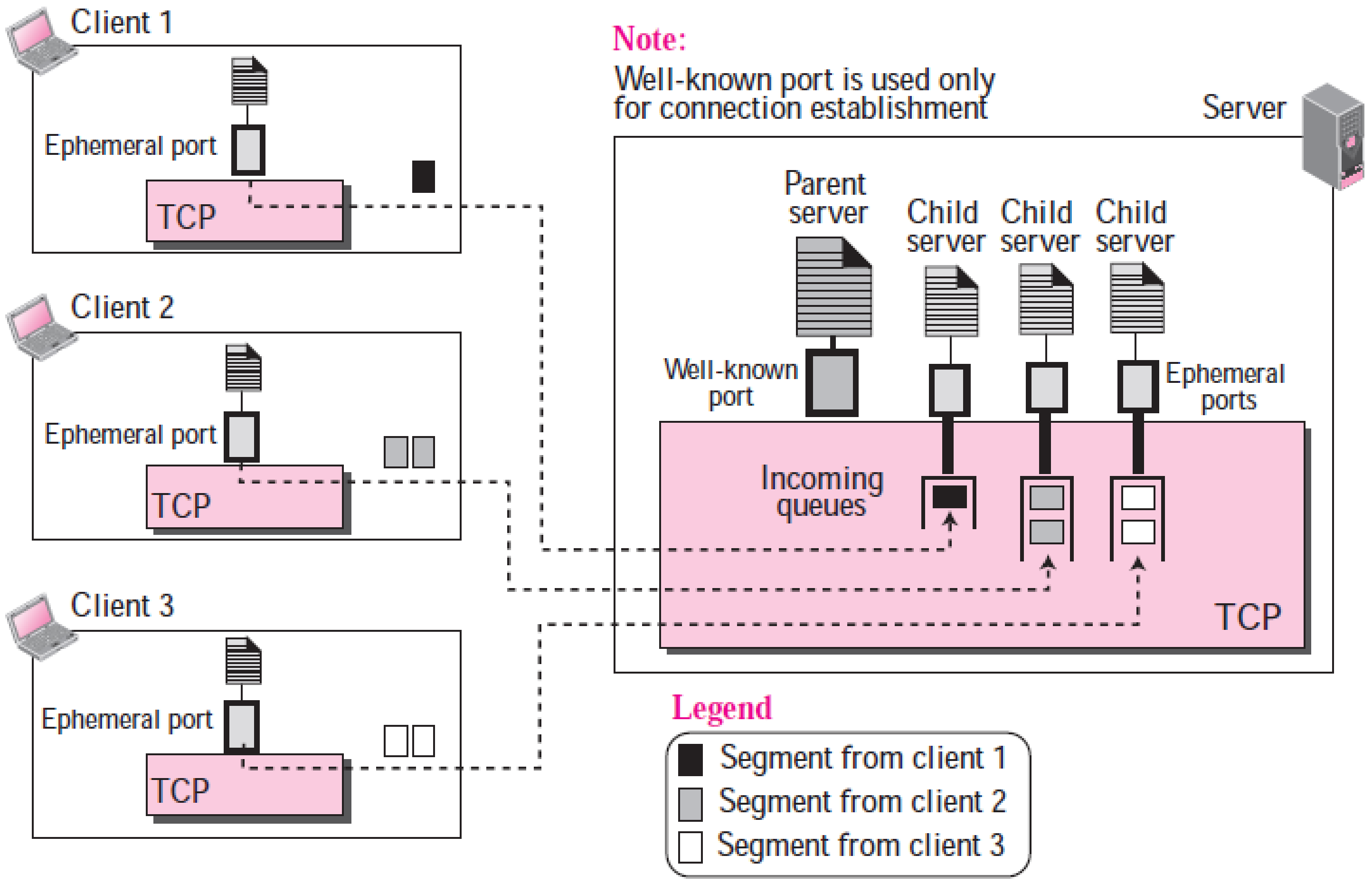
Server Types


Servers
Connectionless iterative
Connectionless concurrent
Connection-oriented iterative
Connection-oriented concurrent
UDP
TCP/SCTP

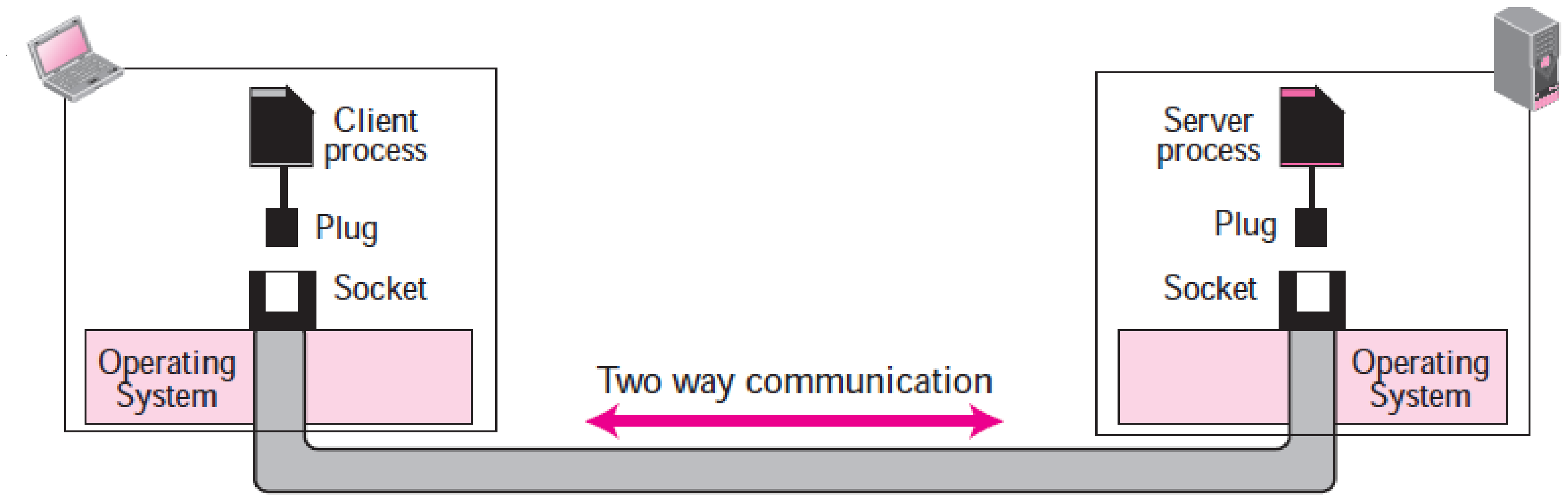
Connectionless iterative server

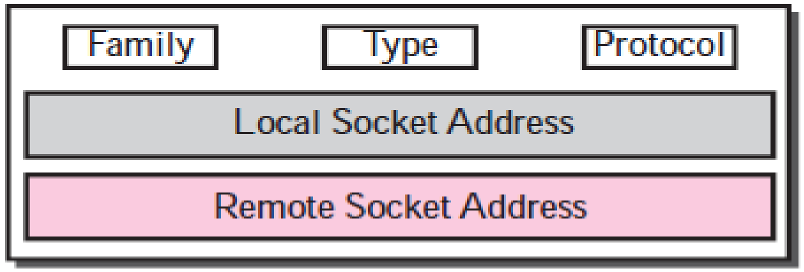


Connection-oriented concurrent server



Socket



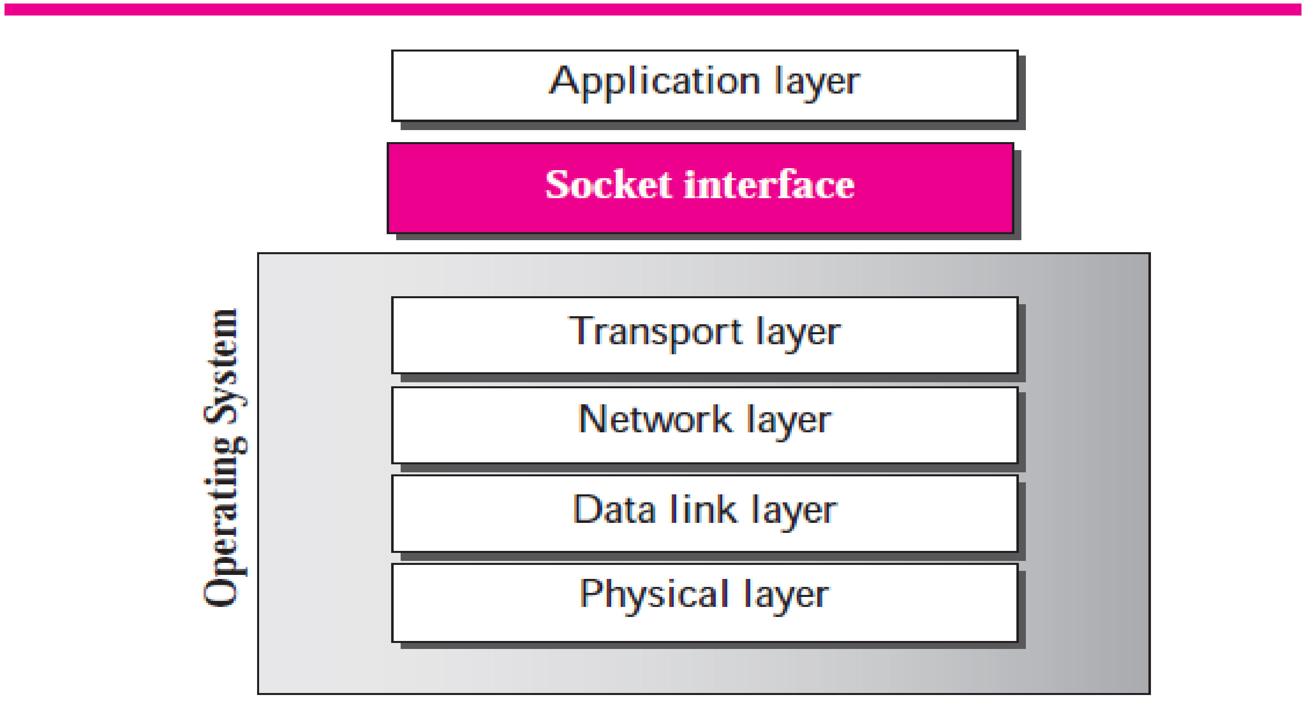


Fields

```
struct  socket
{
      int family;
      int type;
      int protocol;
      socketaddr local;
      socketaddr remote;
 };
```
Generic definition

Relation between the operating system and the TCP/IP suite

Relation between the operating system and the TCP/IP suite

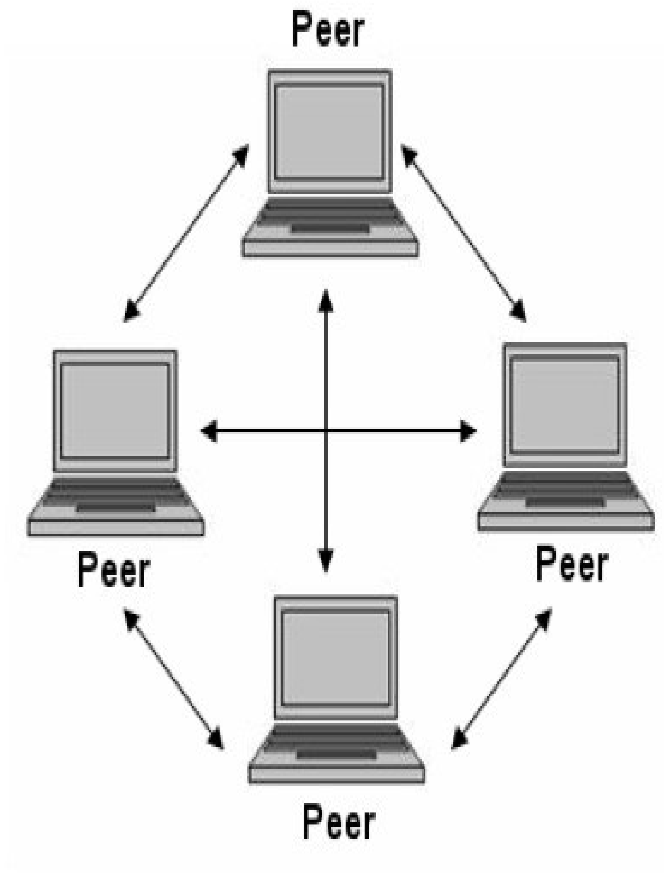


Pure P2P architecture

- P2P architecture is a commonly used computer networking architecture in which each workstation, or node, has the **same capabilities and responsibilities.**

- ***no*** always-on server

- P2P may also be used to refer to a single software program designed so that each instance of the program **may act as both client and server,** with the same responsibilities and status.



Hybrid of client-server and P2P

- Combine the **advantages** of both **client-server** and **peer-to-peer** architectures.

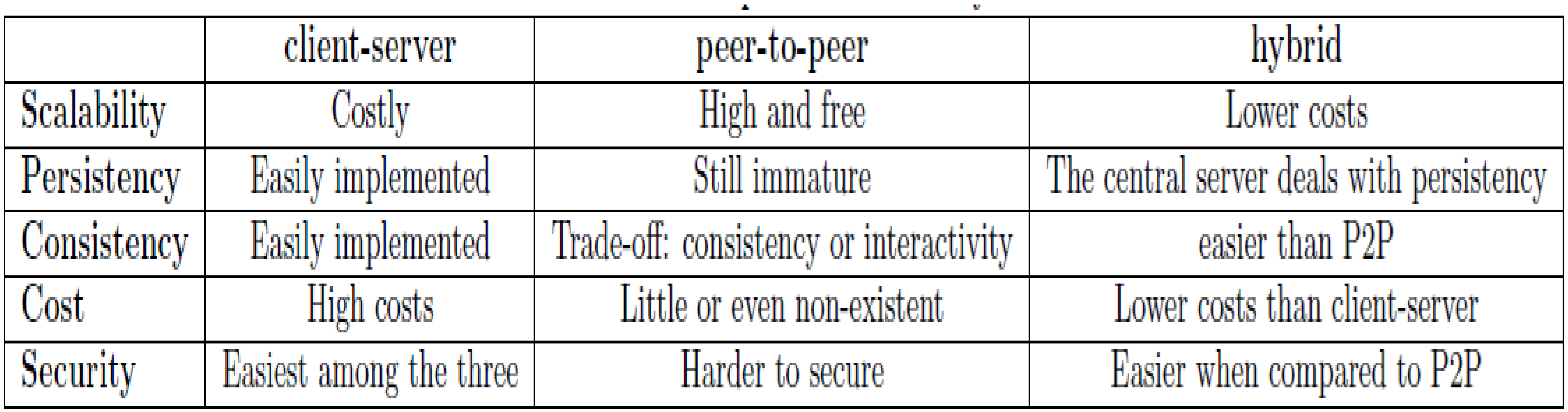
Skype

- voice-over-IP P2P application
- centralized server: finding address of remote party:
- client-client connection: direct (not through server)

Instant messaging

- chatting between two users is P2P
- centralized service: client presence detection/location
  - user registers its IP address with central server when it comes online
  - user contacts central server to find IP addresses of buddies

Comparison



	client-server	peer-to-peer	hybrid
Scalability	Costly	High and free	Lower costs
Persistency	Easily implemented	Still immature	The central server deals with persistency
Consistency	Easily implemented	Trade-off: consistency or interactivity	easier than P2P
Cost	High costs	Little or even non-existent	Lower costs than client-server
Security	Easiest among the three	Harder to secure	Easier when compared to P2P

File Transfer: FTP and TFTP

Objectives

Upon completion you will be able to:

- *Understand the connections needed for FTP file transfer*
- *Be familiar with FTP commands and responses*
- *Know the differences between FTP and TFTP*
- *Be familiar with TFTP message types*
- *Understand TFTP flow and error control*

FILE TRANSFER PROTOCOL (FTP)

File Transfer Protocol (FTP) is the standard mechanism provided by TCP/IP for copying a file from one host to another.

The topics discussed in this section include:

Connections
Communication
Command Processing
File Transfer
Anonymous FTP

Note:

FTP uses the services of TCP. It needs two TCP connections.

The well-known port 21 is used for the control connection and the well-known port 20 for the data connection.

Figure 19.1 *FTP*

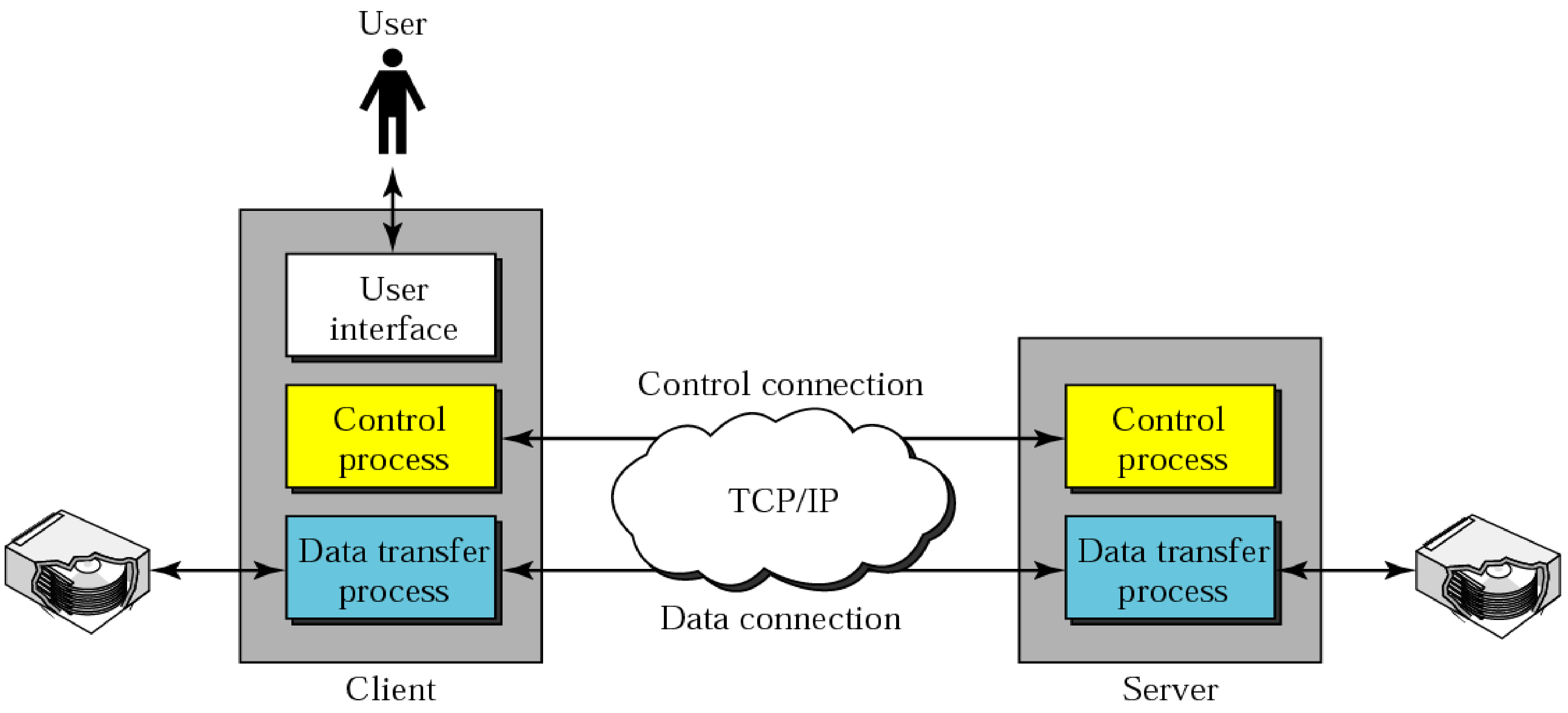
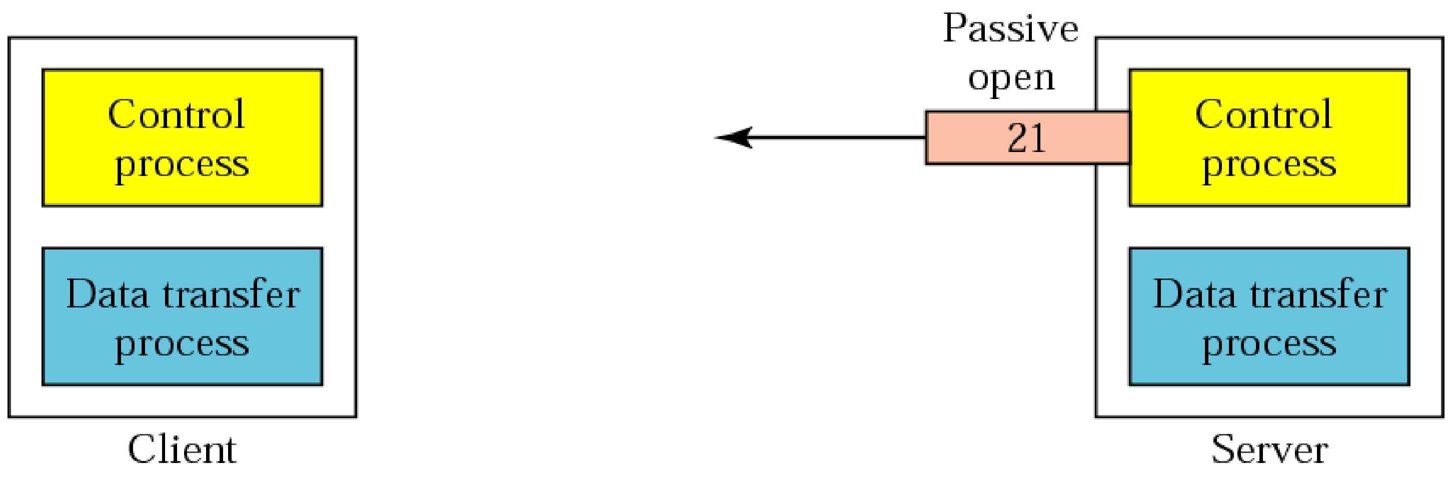


Figure 19.2 *Opening the control connection*



a. Passive open by server

Figure 19.3 *Creating the data connection*

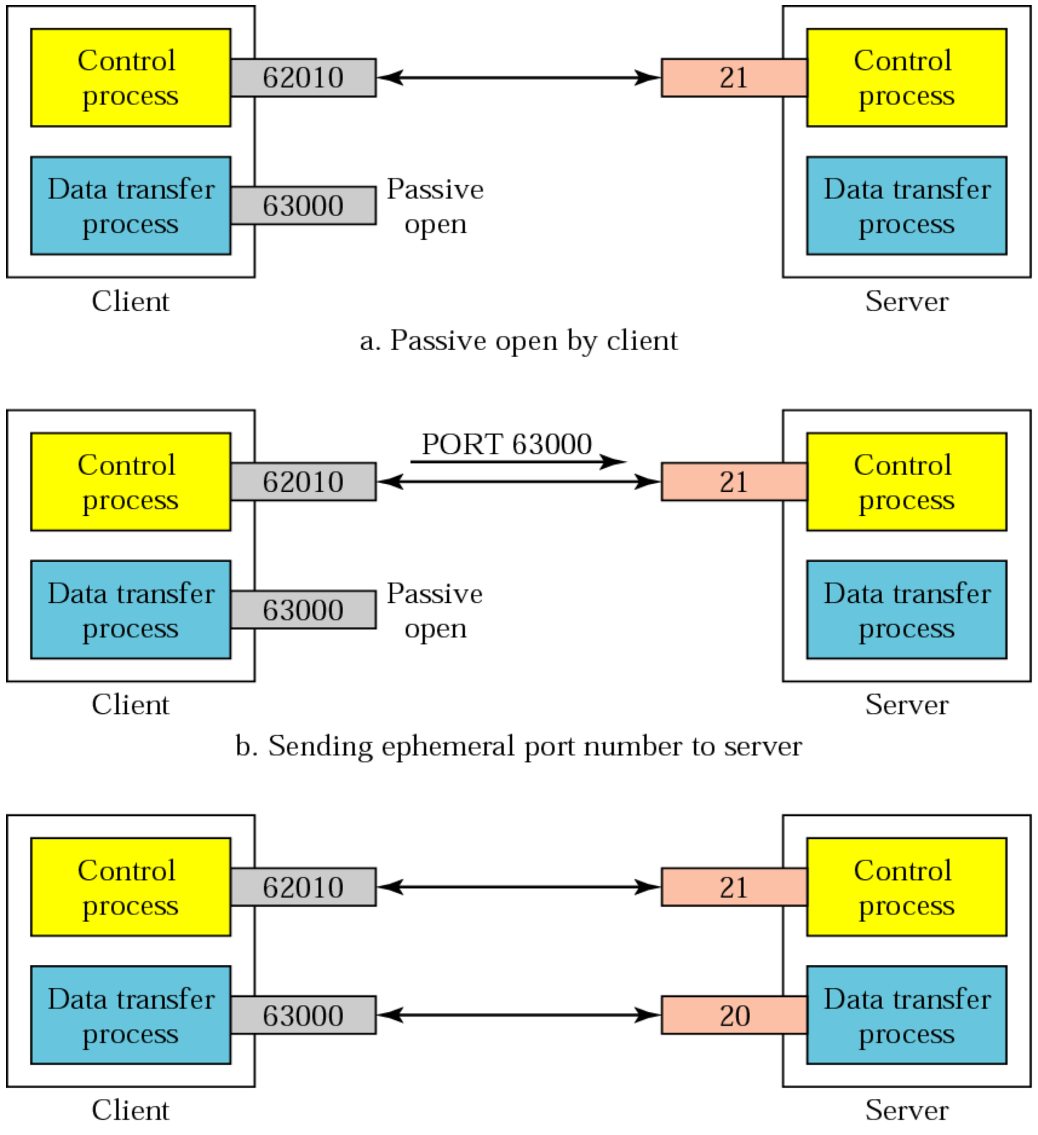


Figure 19.4 *Using the control connection*

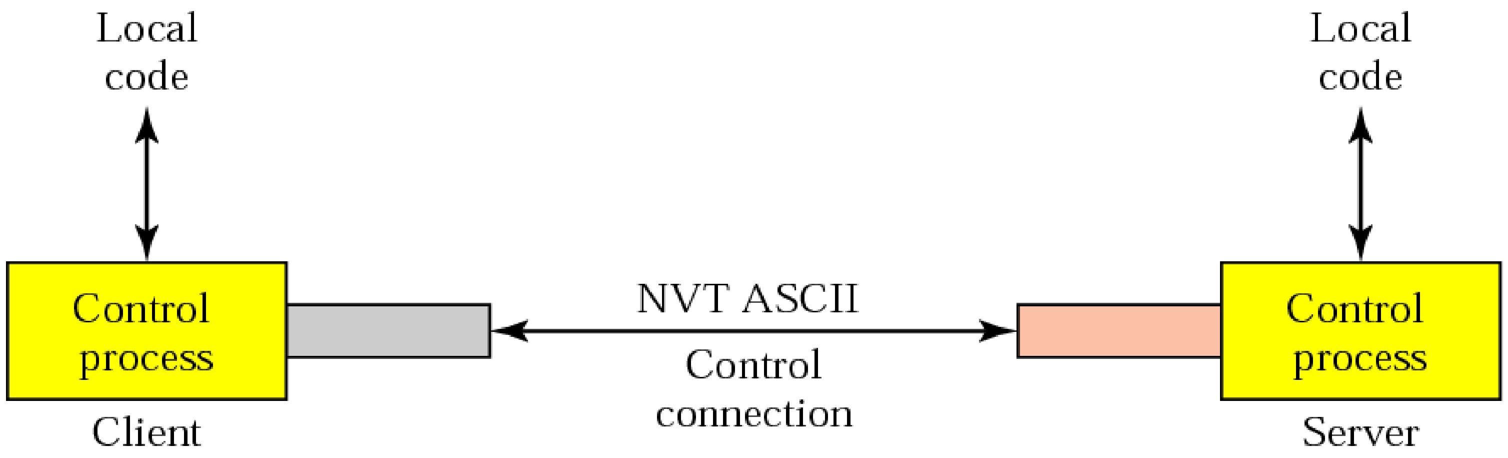


Figure 19.5 *Using the data connection*

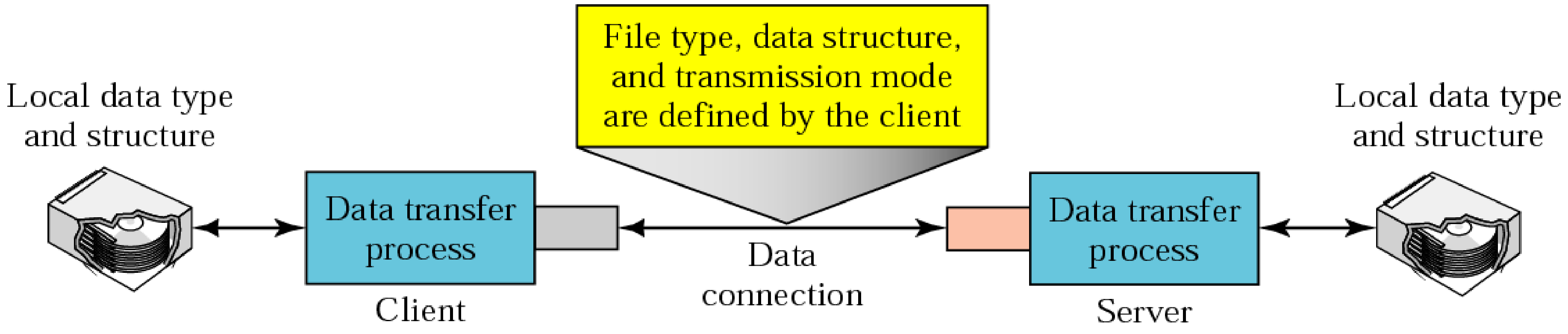
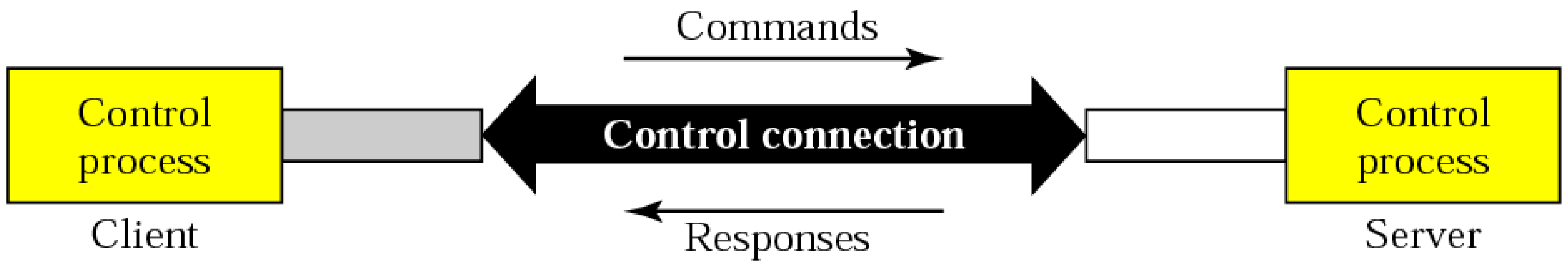

File type, data structure, and transmission mode are defined by the client
Local data type and structure
Data transfer process
Client
Data connection
Data transfer process
Server
Local data type and structure

Figure 19.6 *Command processing*



Example or Application of FTP

FileZilla

Table 19.1 Access commands

Command	*Argument(s)*	*Description*
USER	User id	User information
PASS	User password	Password
ACCT	Account to be charged	Account information
REIN		Reinitialize
QUIT		Log out of the system
ABOR		Abort the previous command

Table 19.2 File management commands

Command	*Argument(s)*	*Description*
CWD	Directory name	Change to another directory
CDUP		Change to the parent directory
DELE	File name	Delete a file
LIST	Directory name	List subdirectories or files
NLIST	Directory name	List the names of subdirectories or files without other attributes
MKD	Directory name	Create a new directory
PWD		Display name of current directory
RMD	Directory name	Delete a directory
RNFR	File name (old file name)	Identify a file to be renamed
RNTO	File name (new file name)	Rename the file
SMNT	File system name	Mount a file system

Table 19.3 Data formatting commands

Command	*Argument(s)*	*Description*
TYPE	A (ASCII), E (EBCDIC), I (Image), N (Nonprint), or T (TELNET)	Define the file type and if necessary the print format
STRU	F (File), R (Record), or P (Page)	Define the organization of the data
MODE	S (Stream), B (Block), or C (Compressed)	Define the transmission mode

Table 19.4 Port defining commands

Command	*Argument(s)*	*Description*
PORT	6-digit identifier	Client chooses a port
PASV		Server chooses a port

Table 19.5 File transfer commands

Command	*Argument(s)*	*Description*
RETR	File name(s)	Retrieve files; file(s) are transferred from server to the client
STOR	File name(s)	Store files; file(s) are transferred from the client to the server
APPE	File name(s)	Similar to STOR except if the file exists, data must be appended to it
STOU	File name(s)	Same as STOR except that the file name will be unique in the directory; however, the existing file should not be overwritten

Table 19.5 File transfer commands (continued)

Command	*Argument(s)*	*Description*
ALLO	File name(s)	Allocate storage space for the files at the server
REST	File name(s)	Position the file marker at a specified data point
STAT	File name(s)	Return the status of files

Table 19.6 Miscellaneous commands

Command	*Argument(s)*	*Description*
HELP		Ask information about the server
NOOP		Check if server is alive
SITE	Commands	Specify the site-specific commands
SYST		Ask about operating system used by the server

Table 19.7 Responses

Code	*Description*
	Positive Preliminary Reply
120	Service will be ready shortly
125	Data connection open; data transfer will start shortly
150	File status is OK; data connection will be open shortly

Table 19.7 Responses (continued)

Code	*Description*
Positive Completion Reply	
200	Command OK
211	System status or help reply
212	Directory status
213	File status
214	Help message
215	Naming the system type (operating system)
220	Service ready
221	Service closing
225	Data connection open
226	Closing data connection
227	Entering passive mode; server sends its IP address and port number
230	User login OK
250	Request file action OK

Table 19.7 Responses (continued)

Code	*Description*
	Positive Intermediate Reply
331	User name OK; password is needed
332	Need account for logging
350	The file action is pending; more information needed

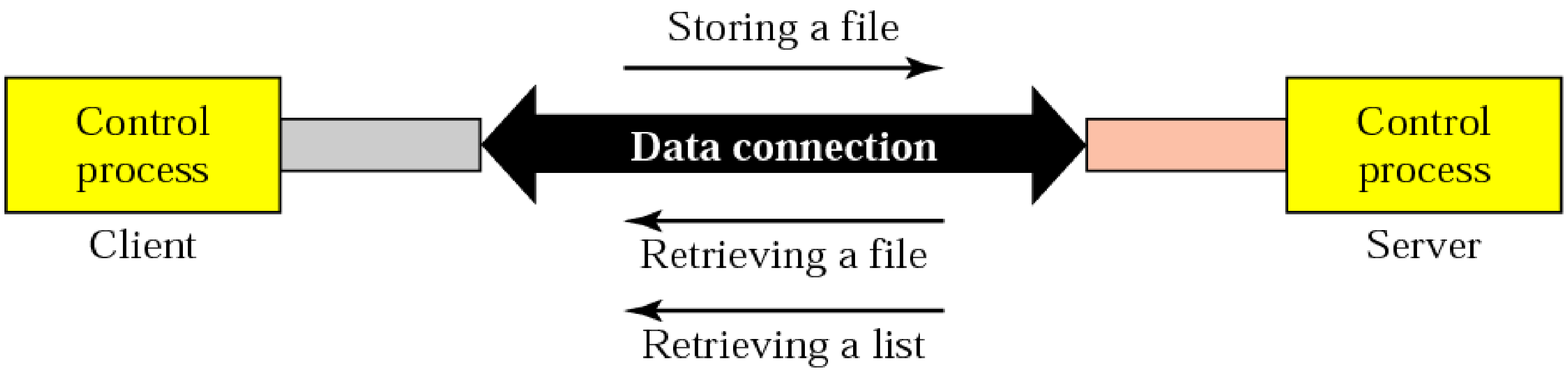
Table 19.7 Responses (continued)

Code	*Description*
	Transient Negative Completion Reply
425	Cannot open data connection
426	Connection closed; transfer aborted
450	File action not taken; file not available
451	Action aborted; local error
452	Action aborted; insufficient storage
	Permanent Negative Completion Reply
500	Syntax error; unrecognized command

Table 19.7 Responses (continued)

Code	*Description*
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command parameter not implemented
530	User not logged in
532	Need account for storing file
550	Action is not done; file unavailable
552	Requested action aborted; exceeded storage allocation
553	Requested action not taken; file name not allowed

Figure 19.7 *File transfer*



EXAMPLE 1

Figure 19.8 shows an example of using FTP for retrieving a list of items in a directory.

1. *After the control connection to port 21 is created, the FTP server sends the 220 (service ready) response on the control connection.*
2. *The client sends the USER command.*
3. *The server responds with 331 (user name is OK, password is required).*
4. *The client sends the PASS command.*
5. *The server responds with 230 (user login is OK)*

See Next Slide

EXAMPLE 1 (CONTINUED)

6. *The client issues a passive open on an ephemeral port for the data connection and sends the PORT command (over the control connection) to give this port number to the server.*
7. *The server does not open the connection at this time, but it prepares itself for issuing an active open on the data connection between port 20 (server side) and the ephemeral port received from the client. It sends response 150 (data connection will open shortly).*
8. *The client sends the LIST message.*
9. *Now the server responds with 125 and opens the data connection.*

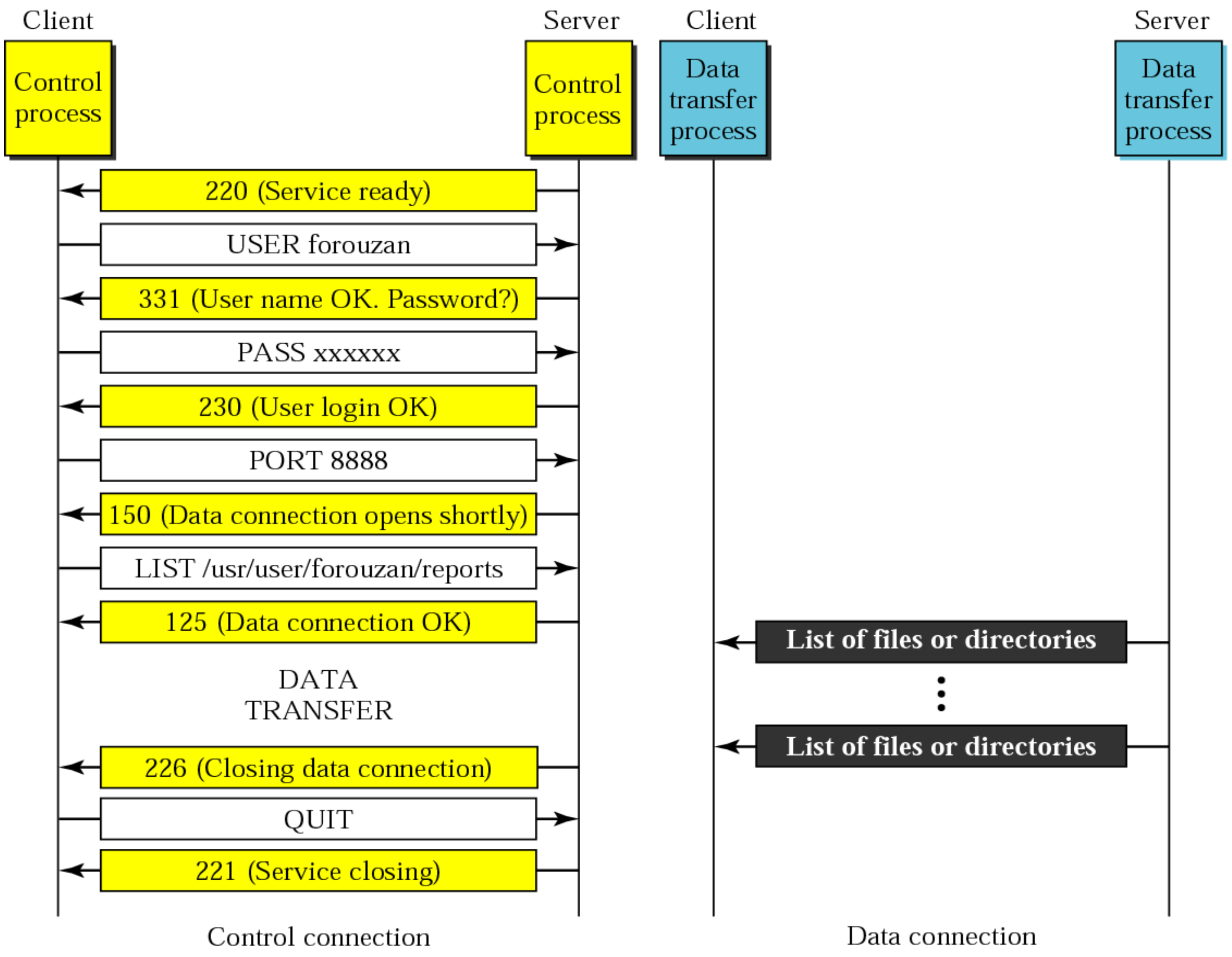
See Next Slide

EXAMPLE 1 (CONTINUED)

10. *The server then sends the list of the files or directories (as a file) on the data connection. When the whole list (file) is sent, the server responds with 226 (closing data connection) over the control connection.*
11. *The client now has two choices. It can use the QUIT command to request the closing of the control connection or it can send another command to start another activity (and eventually open another data connection). In our example, the client sends a QUIT command.*
12. *After receiving the QUIT command, the server responds with 221 (service closing) and then closes the control connection.*

See Next Slide

Figure 19.8 *Example 1*



EXAMPLE 2

The following shows an actual FTP session that parallels Example 1. The colored lines show the responses from the server control connection; the black lines show the commands sent by the client. The lines in white with black background shows data transfer.

```
$ ftp voyager.deanza.fhda.edu
Connected to voyager.deanza.fhda.edu.
220 (vsFTPd 1.2.1)
530 Please login with USER and PASS.
Name (voyager.deanza.fhda.edu:forouzan): forouzan
331 Please specify the password.
```

See Next Slide

EXAMPLE 2

```
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls reports
227 Entering Passive Mode (153,18,17,11,238,169)
150 Here comes the directory listing.
```

```
drwxr-xr-x 2 3027 411 4096 Sep 24 2002 business
drwxr-xr-x 2 3027 411 4096 Sep 24 2002 personal
drwxr-xr-x 2 3027 411 4096 Sep 24 2002 school
```

```
226 Directory send OK.
ftp> quit
221 Goodbye.
```

EXAMPLE 3

Figure 19.9 shows an example of how an image (binary) file is stored.

1. *After the control connection to port 21 is created, the FTP server sends the 220 (service ready) response on the control connection.*
2. *The client sends the USER command.*
3. *The server responds with 331 (user name is OK, a password is required).*
4. *The client sends the PASS command.*
5. *The server responds with 230 (user login is OK).*
6. *The client issues a passive open on an ephemeral port for the data connection and sends the PORT command (over the control connection) to give this port number to the server.*

See Next Slide

EXAMPLE 3 (CONTINUED)

7. *The server does not open the connection at this time, but prepares itself for issuing an active open on the data connection between port 20 (server side) and the ephemeral port received from the client. It sends the response 150 (data connection will open shortly).*
8. *The client sends the TYPE command.*
9. *The server responds with the response 200 (command OK).*
10. *The client sends the STRU command.*
11. *The server responds with 200 (command OK).*
12. *The client sends the STOR command.*
13. *The server opens the data connection and sends the response 250.*

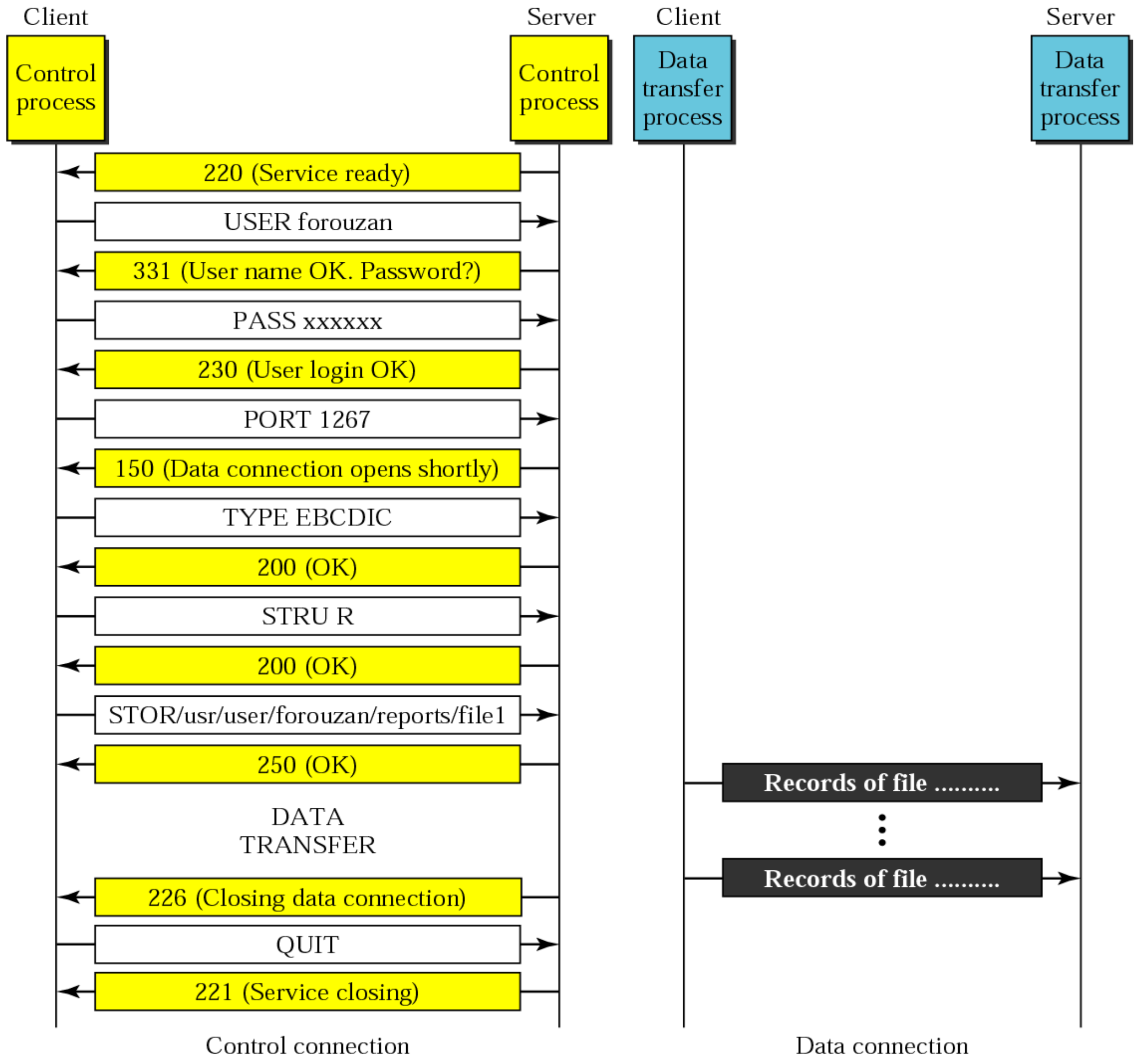
See Next Slide

EXAMPLE 3 (CONTINUED)

14. *The client sends the file on the data connection. After the entire file is sent, the data connection is closed. Closing the data connection means end-of-file.*
15. *The server sends the response 226 on the control connection.*
16. *The client sends the QUIT command or uses other commands to open another data connection for transferring another file. In our example, the QUIT command is sent.*
17. *The server responds with 221 (service closing) and it closes the control connection.*

See Next Slide

Figure 19.9 *Example 3*



EXAMPLE 4

We show an example of anonymous FTP. We assume that some public data are available at internic.net.

```
$ ftp internic.net
Connected to internic.net
220 Server ready
Name: anonymous
331 Guest login OK, send "guest" as password
Password: guest
ftp > pwd
257 '/' is current directory
```

See Next Slide

EXAMPLE 4

```
bin
. . .
. . .
. . .
```

```
ftp > close
221 Goodbye
ftp > quit
```

19.2 TRIVIAL FILE TRANSFER PROTOCOL (TFTP)

Trivial File Transfer Protocol (TFTP) is a simple file transfer protocol without the sophisticated features of FTP.

The topics discussed in this section include:

Messages
Connection
Data Transfer
UDP Ports
TFTP Example
TFTP Options
Security
Applications

Note:

TFTP uses the services of UDP on the well-known port 69.

Figure 19.10 *Message categories*

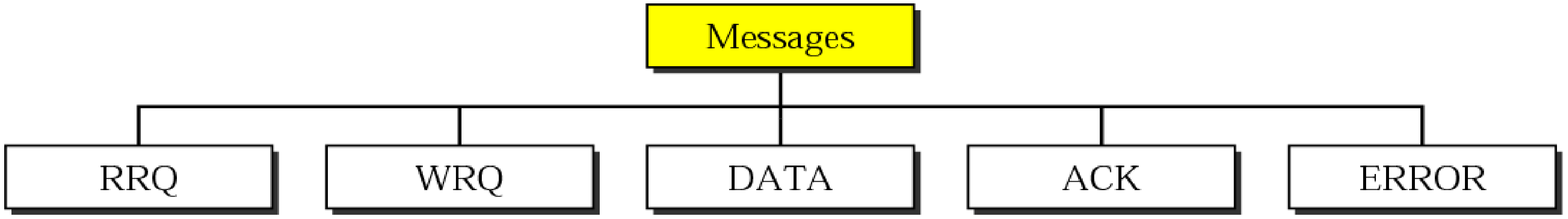


Figure 19.11 *RRQ format*

OpCode = 1	File name	All 0s	Mode	All 0s
2 bytes	Variable	1 byte	Variable	1 byte

Figure 19.12 *WRQ format*

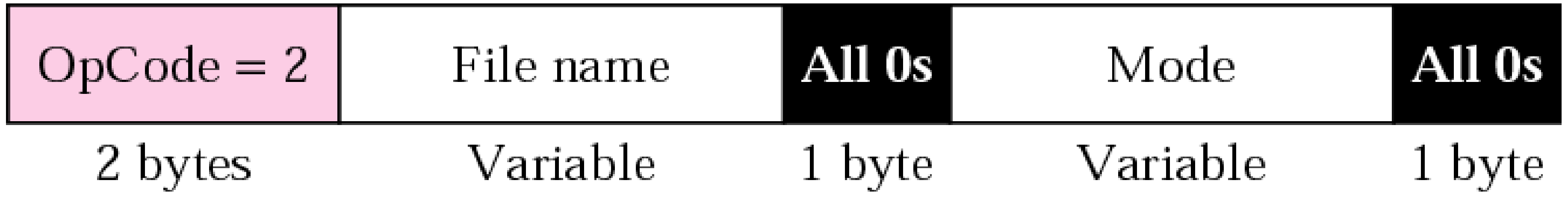


Figure 19.13 *DATA format*

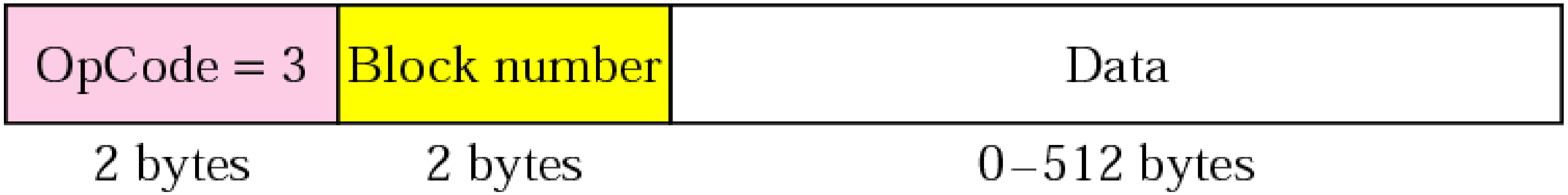


Figure 19.14 *ACK format*

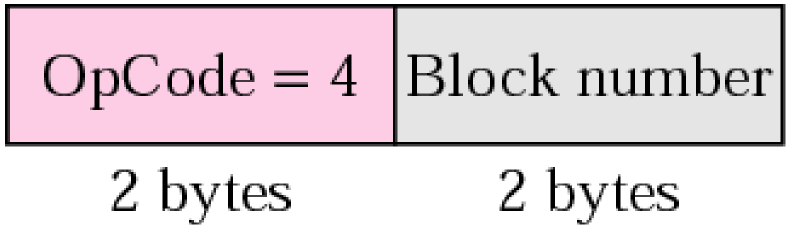


Figure 19.15 *ERROR format*

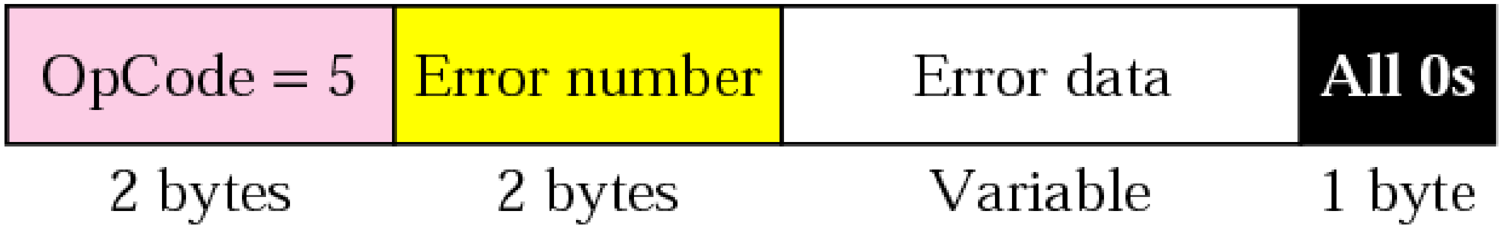
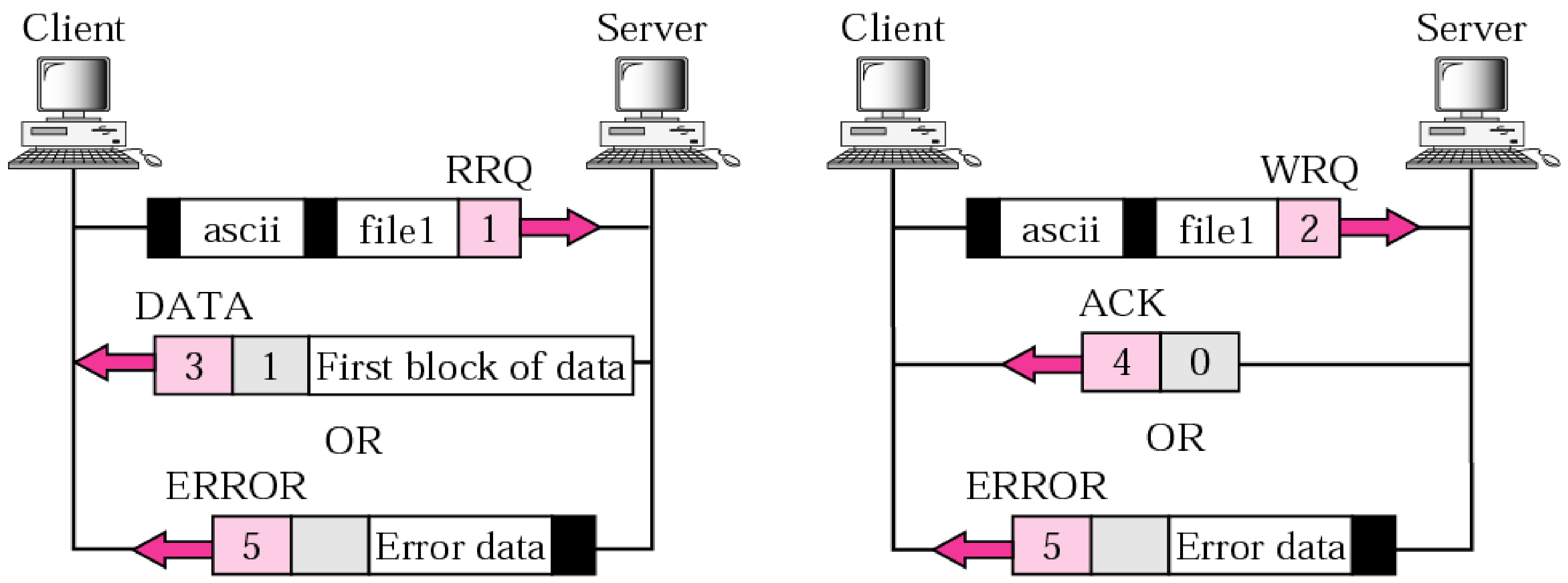


Table 19.8 Error numbers and their meanings

Number	*Meaning*
0	Not defined
1	File not found
2	Access violation
3	Disk full or quota on disk exceeded
4	Illegal operation
5	Unknown port number
6	File already exists
7	No such user

Figure 19.16 *Connection establishment*



a. Connection for reading b. Connection for writing

Figure 19.17 *Sorcerer's apprentice bug*

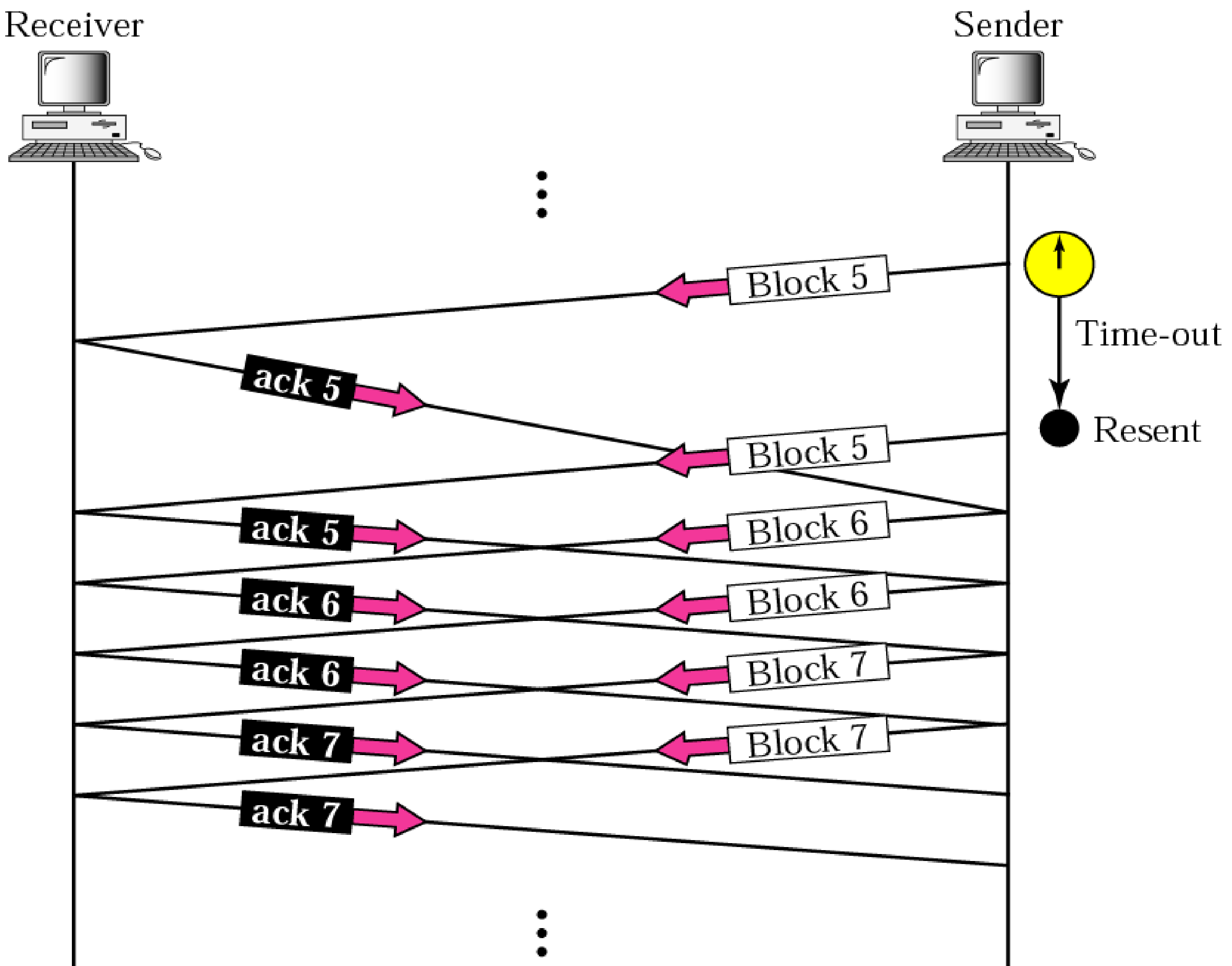


Figure 19.18 *UDP port numbers used by TFTP*

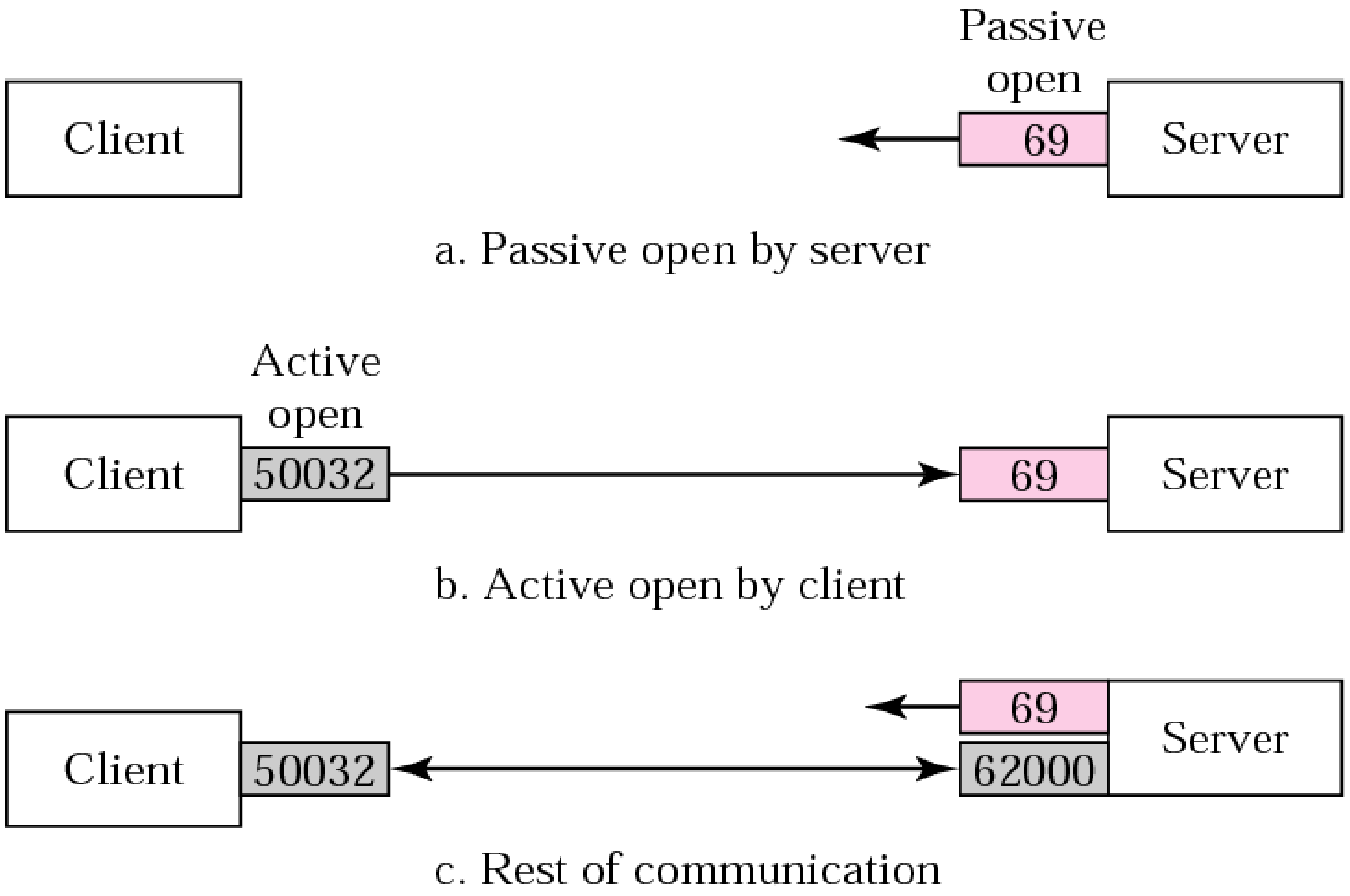


Figure 19.19 *TFTP example*

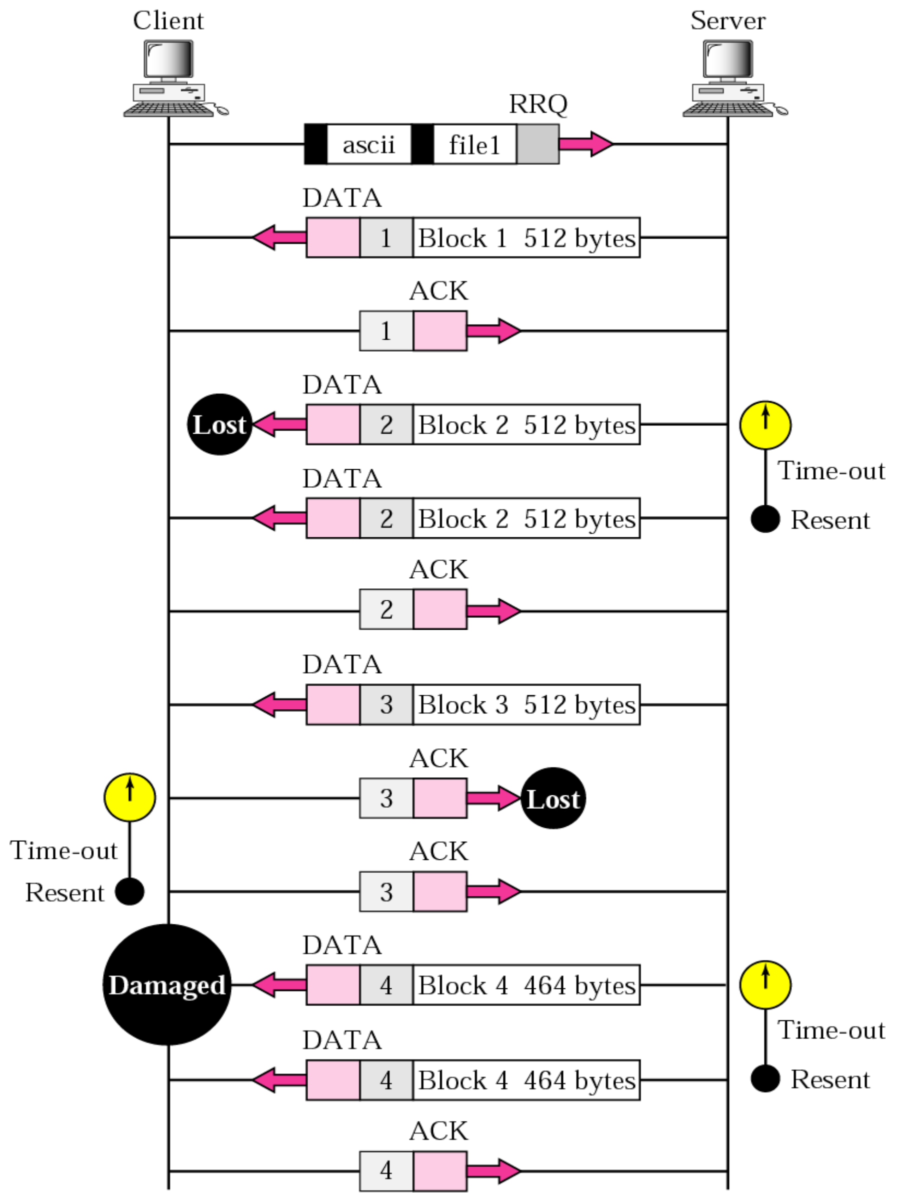
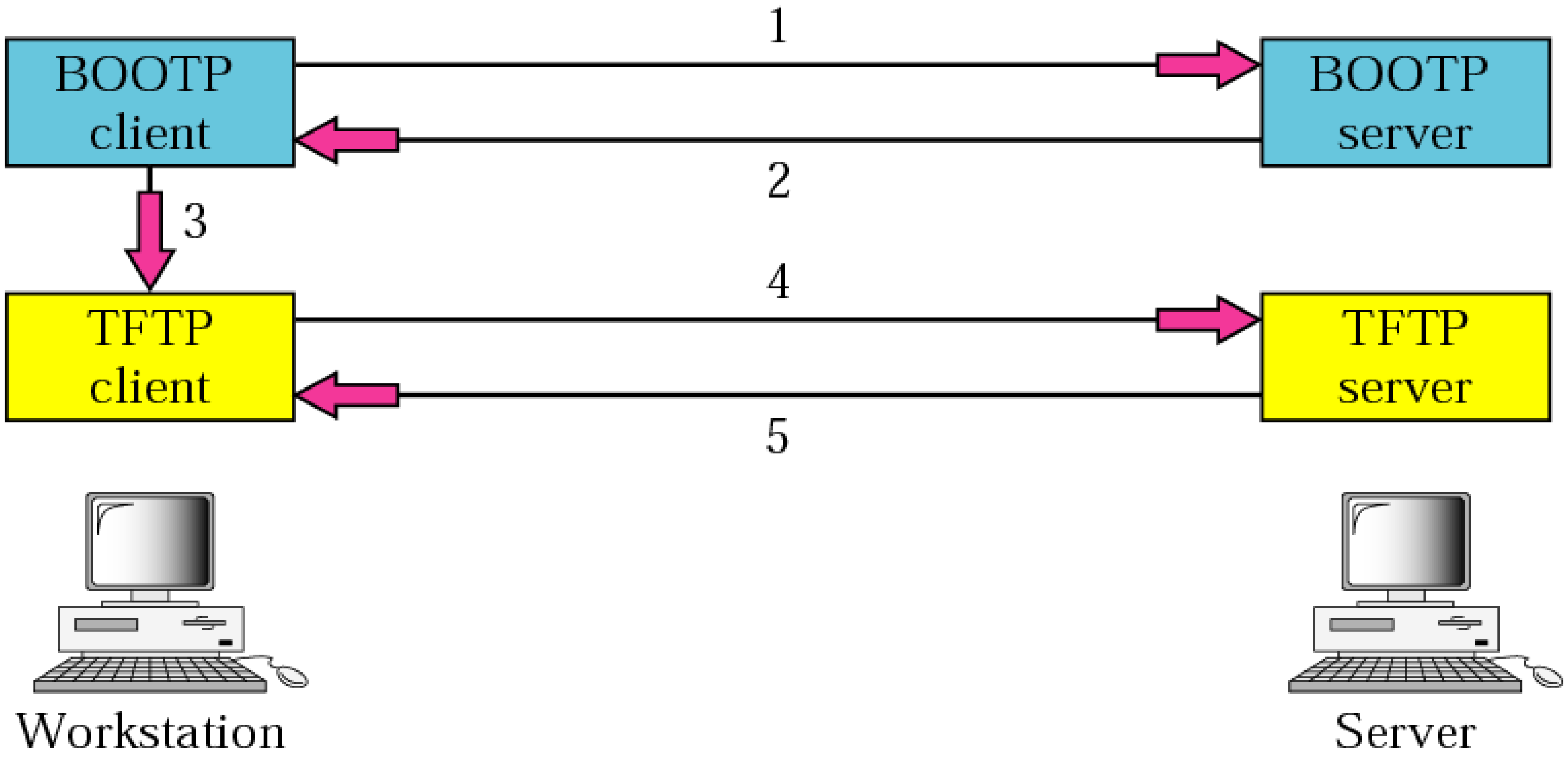


Figure 19.20 *Use of TFTP with BOOTP*


1
BOOTP client
BOOTP server
2
3
4
TFTP client
TFTP server
5
Workstation
Server