LABORATORY WORK NO. 2
APPLICATION LAYER PROTOCOLS USED IN INTERNET

1. Objectives

The objective of this laboratory work consists of getting familiar with the most used application layer protocols: FTP, TELNET, Finger and understanding the specific functioning mechanisms of these protocols.

2. Theoretical considerations

The most known and used application layer protocols are Telnet – offers interactive access at a remote terminal, respectively Ftp – for file transfers. The characteristics of the application layer protocols:

- they are designed based on the transport layer protocols TCP and UDP; many application layer protocols use TCP, but there are applications designed using the UDP protocol due to the high performance offered by the reduction of the load due to the prior establishment of the connection;
- the design model is the client-server mode.

2.1. The transport layer protocols

The TCP protocol is a connection oriented protocol, explicitly designed for ensuring an unfailing flux of bytes from one end of the connection to the other in an unsafe inter-network. An inter-network differs from a proper network by the fact that certain parts of it can differ substantially in topology, bandwidth, delays or packet dimension. The existing implementations are based on sliding window mechanisms.

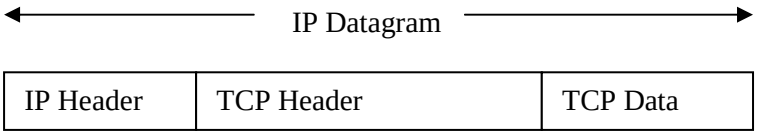

Figure 2.1 *TCP encapsulation in an IP datagram*

The TCP header structure is described in what follows. The **Source port** and **Destination port** fields identify the final points of the connection. The **Sequence number** and **Confirmation number** fields have the meaning of their usual functions. Notice that the last one indicates the next byte waited for and not the last correctly received byte. **The length of the TCP header** indicates the number of 32 bit words that are contained in the TCP header. This option is important because the **Options** field is of variable length.

6 one bit indicators follow: **URG** represents urgent, and is set if the **Urgent Indicator** is valid. The **ACK** bit is 1 if the confirmation field is valid. The **PSH** bit tells the receiver to deliver the information to the application, without memorizing it if it is 1. The RST bit is used to abolish a connection which has become unusable. The SYN bit is used for establishing a connection. It can indicate a connection request if the ACK bit is 0 and an accepted connection if the ACK bit is 1. The FIN bit is used for terminating a connection. The acknowledgement technique used is piggybacking for the transmission from the opposed direction, representing the sequence number of the first byte the emitter expects to receive.

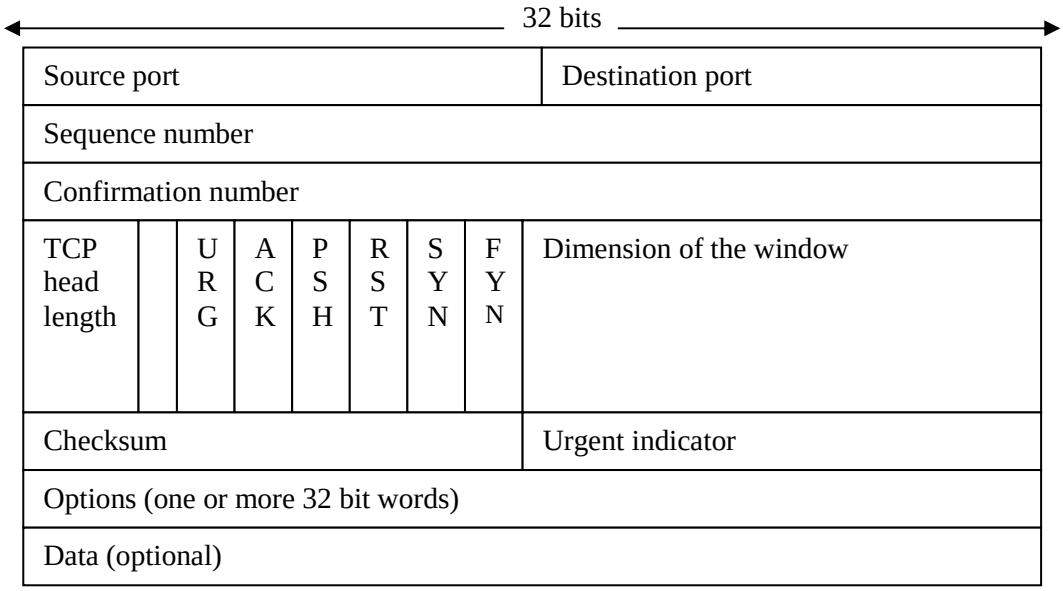

Figure 2.2 *The structure of a TCP packet*

The field **Dimension of the window** indicates the number of bytes that can be sent, starting with the confirmed byte. The maximum dimension of the

reception window depends on the implementation, and the optimal dimension during the transmission is the subject of algorithms.

A **Check field** applicable to the header follows. At the moment of the computation, **Checksum** is set to 0, and the data field is filled with a null extra byte if its length is an odd number. This is computed by the algorithm of the IP protocol and is applied to all the fields of a TCP packet (header and data), including to some special fields which make up a pseudo-header (*TCP pseudo-header*). These fields delivered by the IP protocol to the TCP along with the data segment have the role to prevent an incorrect IP routing.

The **Optional** field has been designed for including a series of extra facilities not covered by the usual header. After that the **Data**, which are optional, follow.
The **Source port** and **Destination port** fields, each of 16 bits, represent the TCP port numbers for the source and respectively destination of the transmission. They are associated to the communication end points. Among these port numbers, some are associated (pre-allocated) to some common applications, such as telnet, ftp.

Service name	**Port name**
Echo	7 (tcp, udp)
Discard	9
Daytime	13 (udp)
ftp	21 (tcp)
telnet	23 (tcp)
Smtp	25 (tcp)
Timed	37 (tcp, udp)
http	80 (tcp)
nntp	119 (tcp)
Login	513
Shell	514

The TCP protocol, having a high complexity, needs for implementing the control operations various specific algorithms and timers. The algorithms used in different implementations for increasing the efficiency, such as: preventing the transmission of small size packets, rapidly freeing the reception window, delayed start for avoiding congestion. Some customarily used timeout specifications are: retransmission timeout, time period between the emission of a TCP packet and the reception of its acknowledgement, TCP connection persistence timer.

The UDP protocol is a connectionless protocol, designed to offer to the applications a modality of transmitting unprocessed encapsulated IP datagrams, which it transmits without prior establishment of a connection. Many client-server applications in which one part emits requests, and the other one emits responses prefer to send UDP datagrams rather than establishing and then freeing a connection. The UDP header is presented in the following figure:

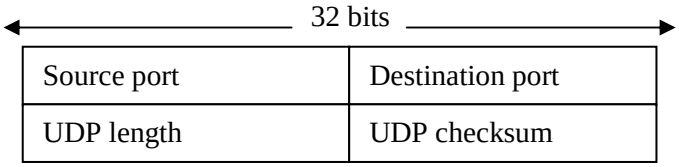

Figure 2.3 *The structure of a UDP packet*

A UDP segment consists of an 8 byte header followed by data; the ports have the same meaning as in the case of the TCP protocol, indicating the final points of the connection. The **UDP length** field includes the 8 bytes of the header plus data, and the field **UDP checksum** represents the checksum applied to the whole segment with the condition that it applies to an even number of bytes.

2.2 Application layer protocols

2.2.1 File transfer

The file transfer application is one of the most common (along with the message exchange through the electronic mail) offering the services most solicited by the large mass of networks users. The main existing implementation, FTP (File Transfer Protocol, described by RFC 959), is one of the oldest, but it still answers to all the requirements appeared for file transfer among computers. The complexity of the FTP protocol consists of the multiple modalities offered for files and file structures manipulation, the work with files of different structure (ASCII, binary, compressed, uncompressed) and of the necessary conversion. The protocol offers security elements, too, by the necessity for the user of knowing a valid account and password on the remote computer.

A simplified version, used by the users especially for software applications downloading (downline loading), is TFTP (Trivial FTP, RFC 783). It is

faster, using as transport support mechanism UDP datagrams and implements only a subset of the FTP services (it is not possible for instance the traversal of the directory hierarchy of the remote computer.

2.2.1.1 FTP

FTP – the protocol used for file transfer (upload – download) in Internet is based on the client-server communication model and uses the port 21. An FTP session is established and maintained until the client finishes the transmission or until communication errors appear. Establishing the connection with the server (an FTP demon which runs on the server) implies the transmission of a login and password. The type of connection known as FTP anonymous implies the transmission as login of the name anonymous and as password of the own e-mail address.

This service allows creating and modifying files, deleting and renaming files and other functions related to the file management. The file transfer implies establishing a secondary connection, in the form of a data link between the two machines, and after the termination of the transfer the connection will be closed automatically.

```
Syntax: FTP host_name
```

After the connection FTP prompter is displayed on the remote machine (after specifying the access name and password) one of the commands of the list is expected:

- connecting to a remote host (*open* – select remote host and initiates the login session);
- selecting a directory (*cd* for choosing the current directory, only for those directories for which the user has access);
- displaying the files available for transfer *Dir*;
- defining the transfer mode (block transfer, data stream modes and ASCII, EBCDIC, image type);
- copying files from and to the remote host (*Get – mget, Put – mput*);
- ending the transfer and disconnecting (*Quit, Close*).

The usage of the proxy transfer mode – allows the clients which have a poor connection to use another server (proxy) for carrying out the transfer.

The communication is carried out using the specifications of the client server model, thus the client launches the request and the server answers. The list of the answer codes in which the bits 2, 3 allow implementing the elaboration of the answers is the following.

Table 2.1 *FTP answer codes*

1xx	Preliminary positive answer
2xx	Completing positive answer
3xx	Intermediary positive answer
4xx	Transitory negative answer completion
5xx	Permanent negative answer completion

2.2.1.2 TFTP

Trivial file transfer protocol (RFC 1350, RFC 2349) is a standard protocol which allows file transfer from a computer on which a TFTP server runs. The protocol is implemented over the UDP protocol. The client's initial request is launched on the port 69 then the port which will be used for the actual communication will be determined. The protocol does not offer support for user authentication, thus being an insecure protocol.

```
Syntax: TFTP – i host_name GET|PUT source destination
```

`i` indicates the type of the transfer;
`host_name` represents the name of the remote machine;
`GET-PUT` remote-local file (determines the direction of the transfer);
`Source-Destination` – represents the full name of the file to be transferred, respectively the destination.

The TFTP protocol may implement a multicast option also, which allows more clients to load simultaneously files from a server using multicast packets. This option is used by identical machines which want to download simultaneously the same configuration from a TFTP server. Because the protocol does not have an authentication mechanism, the server has to restrict the access in writing for the public directories; in this sense there are implementations which offer host lists which have access to a certain TFP server.

2.2.2 Remote Access

The remote access to a computer connected to the network (located anywhere) is accomplished in the form of a terminal session (for this action the terminologies remote login and virtual terminal are also used), the best known protocol for this operation being Telnet (RFC 854).

The two applications used for remote connection using the TCP-IP protocols are: Telnet (Terminal emulation) the protocol designed for allowing connectivity among machines using different operating systems (heterogeneous systems using any type of terminal) and Rlogin, the protocol developed for offering remote connecting functionality only to Unix machines.

2.2.2.1 Telnet

Telnet is one of the most popular remote connection applications, called the remote login application of the Internet based on the client-server paradigm. The minimal functionality of this protocol implements the notion of virtual network terminal, offering support for prior negotiation of the connection. The telnet application allows the client to run programs on the remote computer, thus it transmits the characters received from the keyboard and returns the results of the execution of the program on the remote machine.

A telnet session implies opening a connection between the terminal (or the local machine) and the remote computer, connected to the network; that is why the name of the remote machine must be known. The connection is closed by the user and the context of the local machine is brought back. Any telnet implementation must include emulators for the most various terminals (IBM 3270, Digital VT100, etc.).

```
Syntax: Telnet host_name
```

`Host_name` – represents the name of the remote host on which the telnet server (demon) runs and the terminal type represents the type of the terminal which will be emulated on the remote machine.
The elements on which establishing a connection using the Telnet protocol is based on are the concept of virtual terminal – generic device having the basic structure corresponding to various implementations of real terminals, a symmetric vision of terminals and processes and negotiation of terminal options.

The virtual terminal contains a printer (display) and a keyboard for allowing the transfer and reception of data in ASCII format, operating in half-duplex mode at line level, based on a local echo function (offers reduced load if there is a remote echo function, too).

The internal commands inaccessible to the user have the following structure:

Interp. code	Command code	Negotiated

Telnet uses the byte 0xFF (with the meaning IAC – Interpret as command).

The most important commands are:

- `close` – terminated current connection;
- `open name` – allows establishing a connection with he specified machine;
- *`display`* – displays operating parameters;
- *`mode`* – establishes the transfer mode (line or character mode);
- *`quit`* – exits telnet mode;
- *`send`* – sends characters;
- *set* – sets operating parameters;
- *`status`* – displays status information.

Here is a list of the Telnet commands.

Table 2.2 *Types of Telnet options*

Command	**Code**	**Comment**
SE	240	End of sub-negotiation
NOP	241	No operation
SB	250	Indicates the sub-negotiation of a following option
WILL	251	Indicates the confirmation (request) of use
Wont	252	Indicates the denial of use (or of continuing)
DO	253	Usage request (or usage confirmation)
DON'T	254	Request to stop the usage toward the connected part
Break	243	The character break in NVT
IAC	255	Interpret as command

Option negotiation. The operation of negotiation of the options is symmetric; each communication end can transmit one of the following requests for a certain option:

- `will` – the emitter wants the validation of the option;
- `do` – the emitter wants the receiver to validate the option;
- `wont` – the emitter wants the invalidation of the option;
- `don't` – the emitter wants the receiver to invalidate the option.

The purpose of the Telnet protocol is constituted by the creation of a standard interface for hosts that communicate in a network. A telnet server exists only for Unix machines, on these Unix programs and commands can run, such as mc, pine, ping, Lynx. The processing in the case of the Telnet application is carried out on the remote machine. At the presentation layer (ASCII), the session layer (for transmission) and then at the transport layer the data are segmented and the port address and their correctness will be verified. At the network layer the headers with the source and destination IP addresses will be added, and if the source of the request does not have a physical address it will generate an arp address resolving request.

Table 2.3 *The Telnet options codes*

1	Echo
3	Suppress go ahead
5	Status
6	Timing mark
24	Terminal type
31	Window size
31	Terminal speed
33	Remote flux control
34	Timing mark
36	Line mode
37	Environment variables

The remote machine will execute the command and will use the same mechanisms for transmitting the answer, thus the entire process will be repeated until the client closes the remote session. The Berkeley UNIX processors incorporate the rlogin command, with an analogous functionality.

2.2.2.2 The FINGER protocol

```
Syntax: FINGER user@hostname (or finger hostname)
```

The command allows displaying the information about users for any system which runs a standard finger service. The *user* option is facultative and returns the list of the users of the system identified by hostname. *Hostname* represents the name of the host machine on which the search is desired; if this command is used with an uncooperative host (on which the finger demon does not run) the solicitation will be refused.
The finger service is included in the Windows NT clients, thus information specific to any system which runs a standard finger system can be obtained. The NT server does not include a finger service (this is publicly available at the Windows NT Academic European Center at the address www.emwac.ed.ac.uk).

3. Lab activity

3.1. Test the application layer protocols described in the laboratory work using the on-line documentation facilities.
3.2. Study the differences between telnet and rlogin. Test a Telnet connection session in character, respectively line working mode.
3.3. Carry out different transfers from different addresses using different clients (Linux, Windows).

Notes