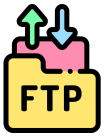
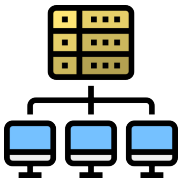
U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Report :



File Transfert Protocol
&
Network Configuration



• Pedro TEIXERA — up201606587@fe.up.pt
• João FONSECA — up201603954@fe.up.pt
• Nowen ADAMCZAK — up201902081@fe.up.pt

Index

N.B : We can find the globale behaviour of our code (Writter and Reader) in annexes.

1 - Introduction

This project is split in two parts. First we have to develop an application to download a file located on a **FTP server.** For that we have to use the RFC corresponding to the FTP protocol to create a client allowed to download files with FTP protocol. On the other hand we will configure a network with switch and router, to implement virtual LANs, NAT ,etc. At the end of this project we will be able to download a file with our own FTP application through our own network configuration. To understand what happened in the network we use Wireshark to listen the different interfaces and see what packets are sent.

Part 1 - Development of download application

1.1 - The aim

The aim of this application is to download a file located on a FEUP FTP server. As describe on the following schema (Fig .1) :

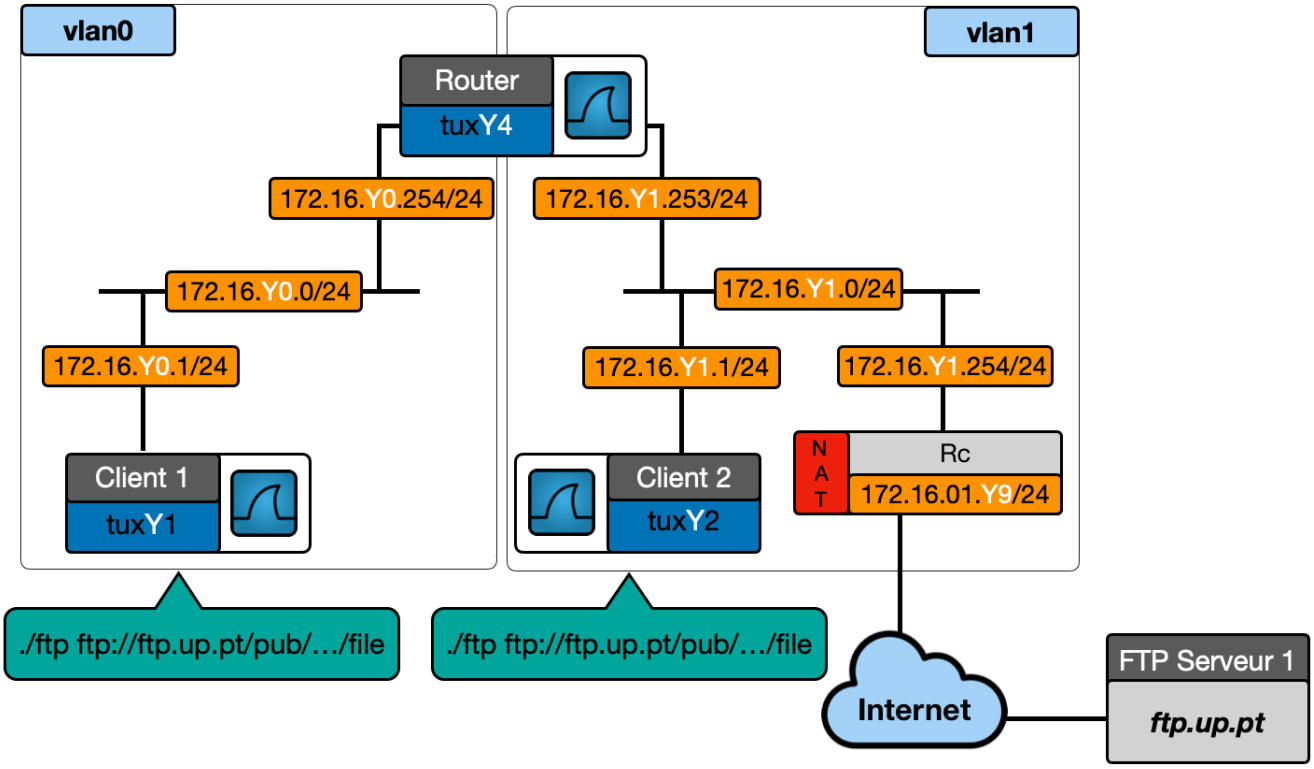


Fig. 1

- This represent the network for an experience where two FTP clients want to download the same file in the same time with our ***ftp*** application coded in C.
- The letter 'Y' can change according to the computers in the lab.
- This network represent two different VLAN routed by **tuxY4** configured as a router.
- **Rc** is a commercial router use to create a **NAT** to access internet.
- **Wireshark** is use on every computer to analyse **TCP/IP** packets (ICMP, ARP, etc.).
- The client (on computer tuxY1 and tuxY2) will use our FTP C program to download a file from a server connected to internet.

1.2 - The FTP protocol

To understand how works the FTP protocol we had searched for the RFC corresponding to the File Transport Protocol (RFC 3659). We also use **telnet** and **Wireshark** to communicate and analyse packets with a private ftp server.

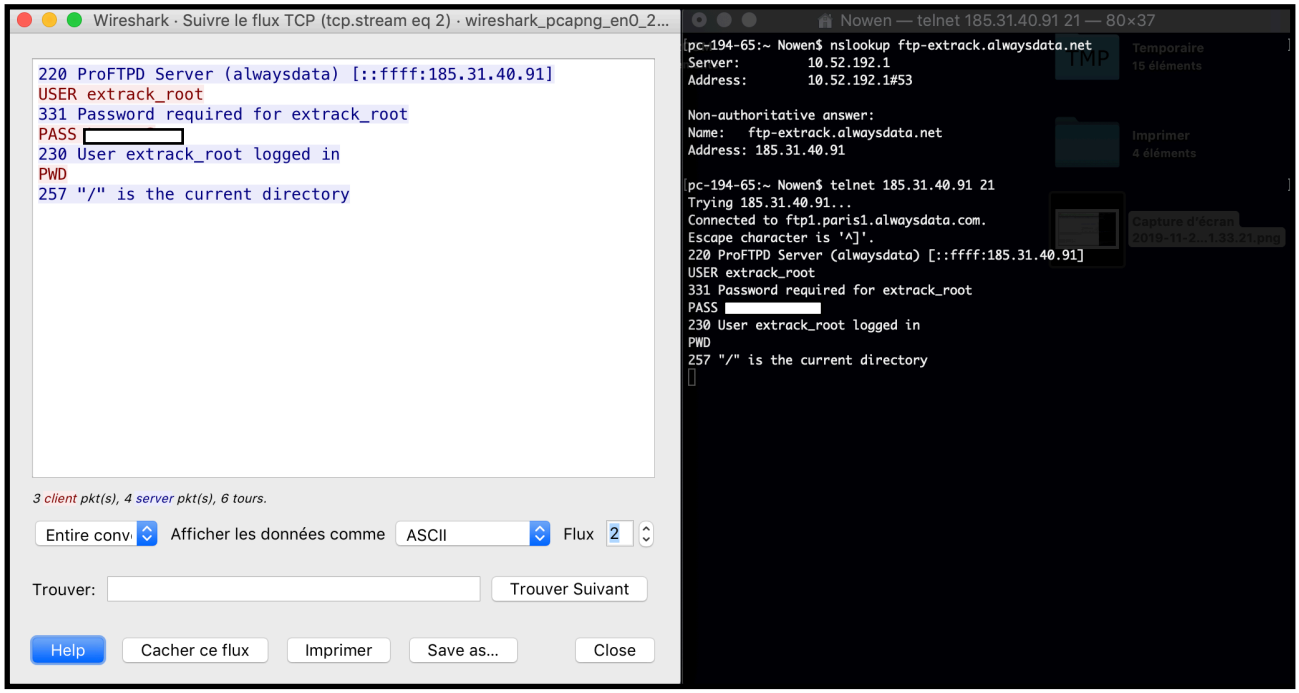


Fig. 2

- First ***nslookup*** is use to know the IP of our private server.
- Next, with the IP and **telnet** protocol we can connect to the FTP server with the port 21 (default port for FTP). We also tried with **netcat** (nc on Mac) but this protocol doesn't send '\r\n' at the end of each message so the FTP server never answered.
- On the right we send FTP command such as USER, PASS, etc. and on the left we can the the same communication through Wireshark.

The most important thing to understand with FTP is that protocol uses two differents socket to send file. One for the command and one for the data. The FTP protocol offers two different mode to iniate those sockets, the **active** and **passive** mode.

Actif mode :

- The client sends the port (with the command **PORT X**) that he wants to use for receiving the data.

Passif mode :

- The client chooses to use passive mode (with the commande **EPSV**). The server will send the port that he wants to use to create the data-socket with the client.

- The following figure explain how works the differents socket for the **passive mode** :

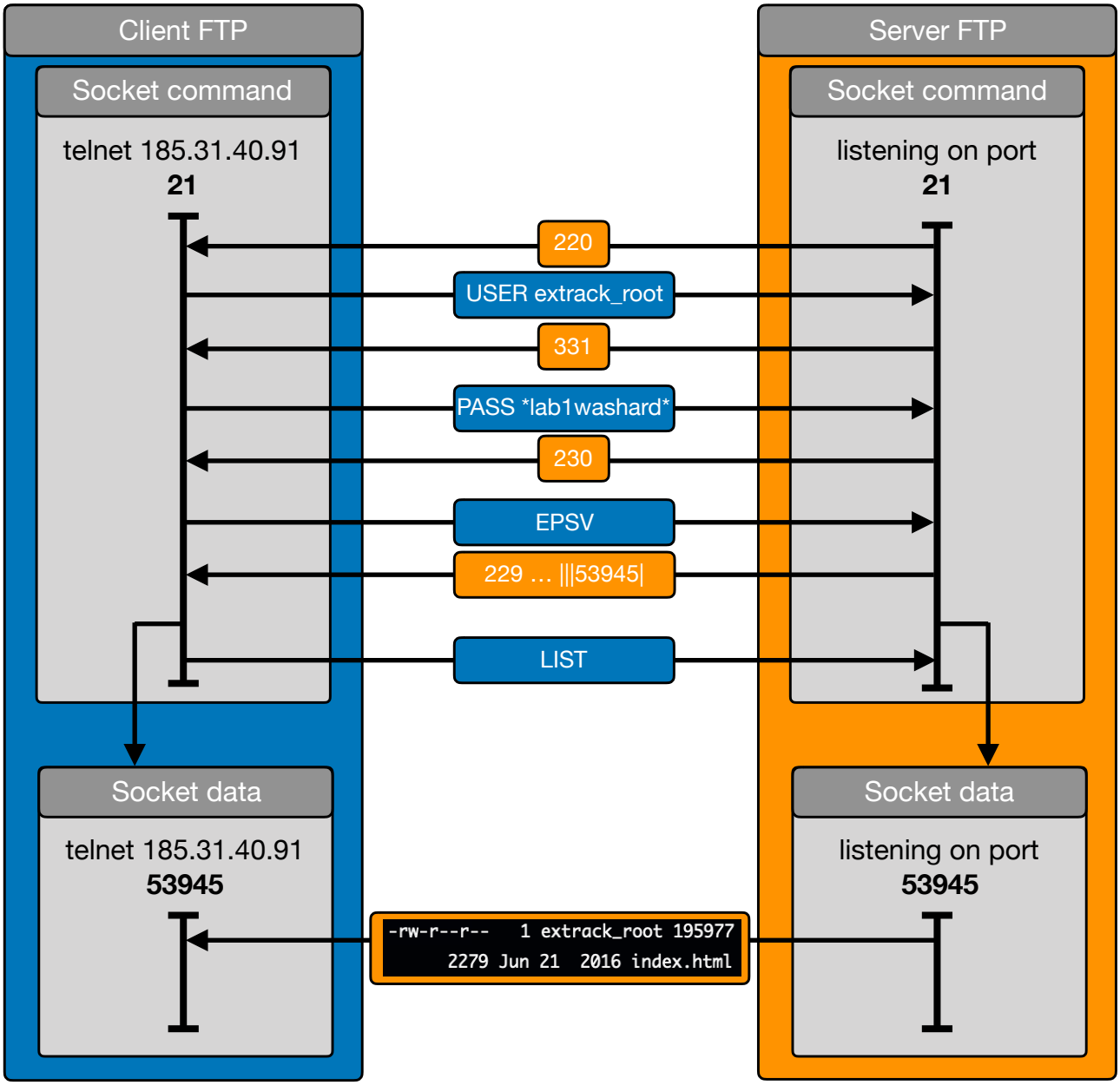


Fig. 3

- The client open the sokcet with telnet, the ip and the port 21, the server ack with 220 to say hello.
- The user loggin correctly and send the EPSV to open the extended passiv mode.
- The server answer with the port to use for the data socket.
- The client open the socket with telnet, the **same** ip and the **new port**.
- The client can now send command which need data socket such as **LIST** (correspond to « ls » on Linux)
- The server answer with the current file's list through the data socket.

• Now we will use the passive mode to take the file “**www/R/C/O/M/index.html**“ :

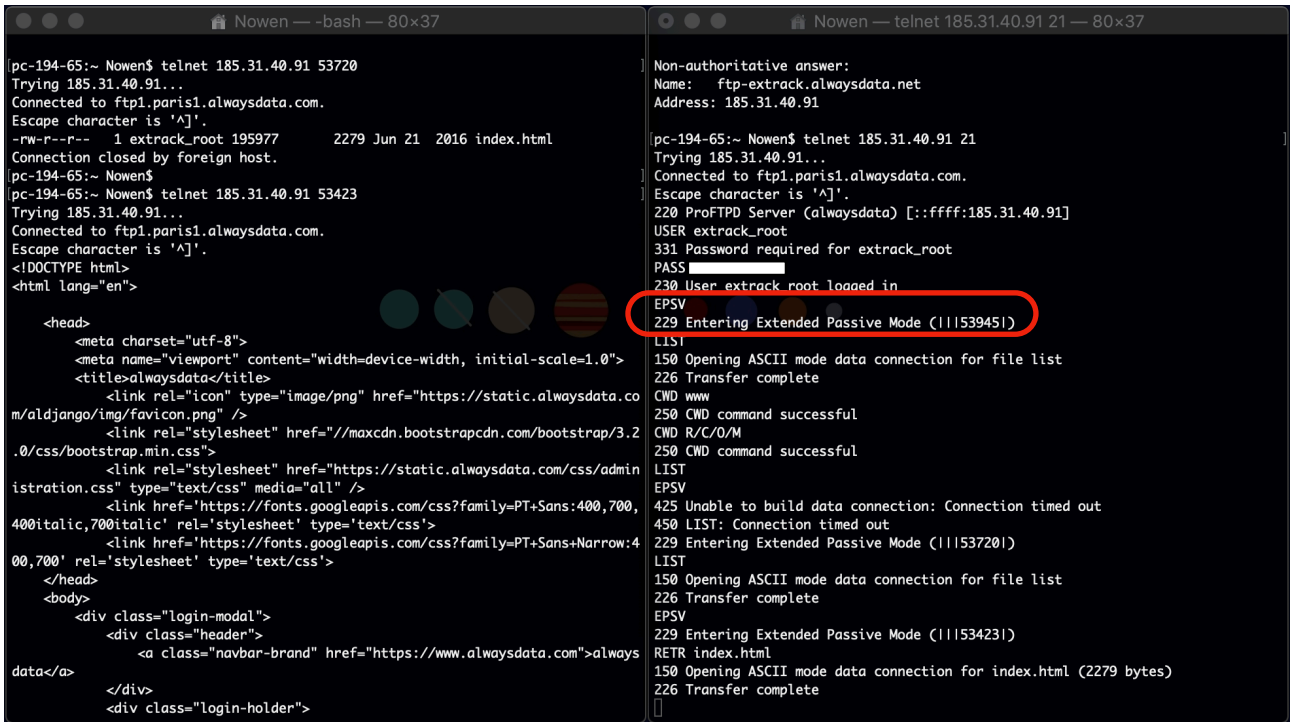


Fig. 4

- The client open the sokcet with telnet, the ip and the port 21, the server ack with 220 to say hello.
- The user loggin correctly and send the EPSV to open the extended passiv mode.
- The server answer with the port to use for the data socket.
- The client open the socket with telnet, the **same** ip and the **new port**.
- The client can now send command which need data socket such as **LIST** (correspond to « ls » on Linux)
- The server answer with the current file’s list through the data socket.
- Next, the client have to reach the file he wants with the command **CWD** (like cd)
- The socket is automaticaly closed after each use.
- The user use the command **RETR** to retrieve the file **index.html**.
- The data are sent directly via the data socket which can be read and use to recreate the file on the client’s local disk.

1.3 - The FTP application

After understood how works the FTP protocol we can start to code the FTP application in C. First we will explain the structure of the code and how we thought the developpement to solve the problem.

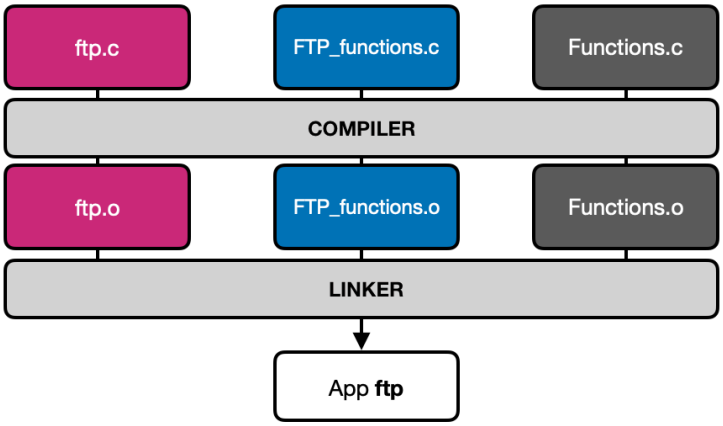


Fig. 5 Headers file are no represented here.

The file **functions.c** is use for the following purpose :

```c
typedef struct URL URL;
struct URL {
  char*  user;
  char*  password;
  char*  host;
  char*  ip;
  char** path;
  char*  filename;
  int    path_size;
  int    port;
};

void setURL(URL* url,char* input);
void printURL(URL url);

char*  getIp(char* server_name);
char** parseur(char* string,char* separator,int* path_size);
```

Fig. 6

The structure URL is use to have all the information needed to **connect** the client to the FTP server. With the function **setURL** (which use **parseur**) we are able to cut the input given by the user to fill the URL structure. as explain inf **Fig. 7**.

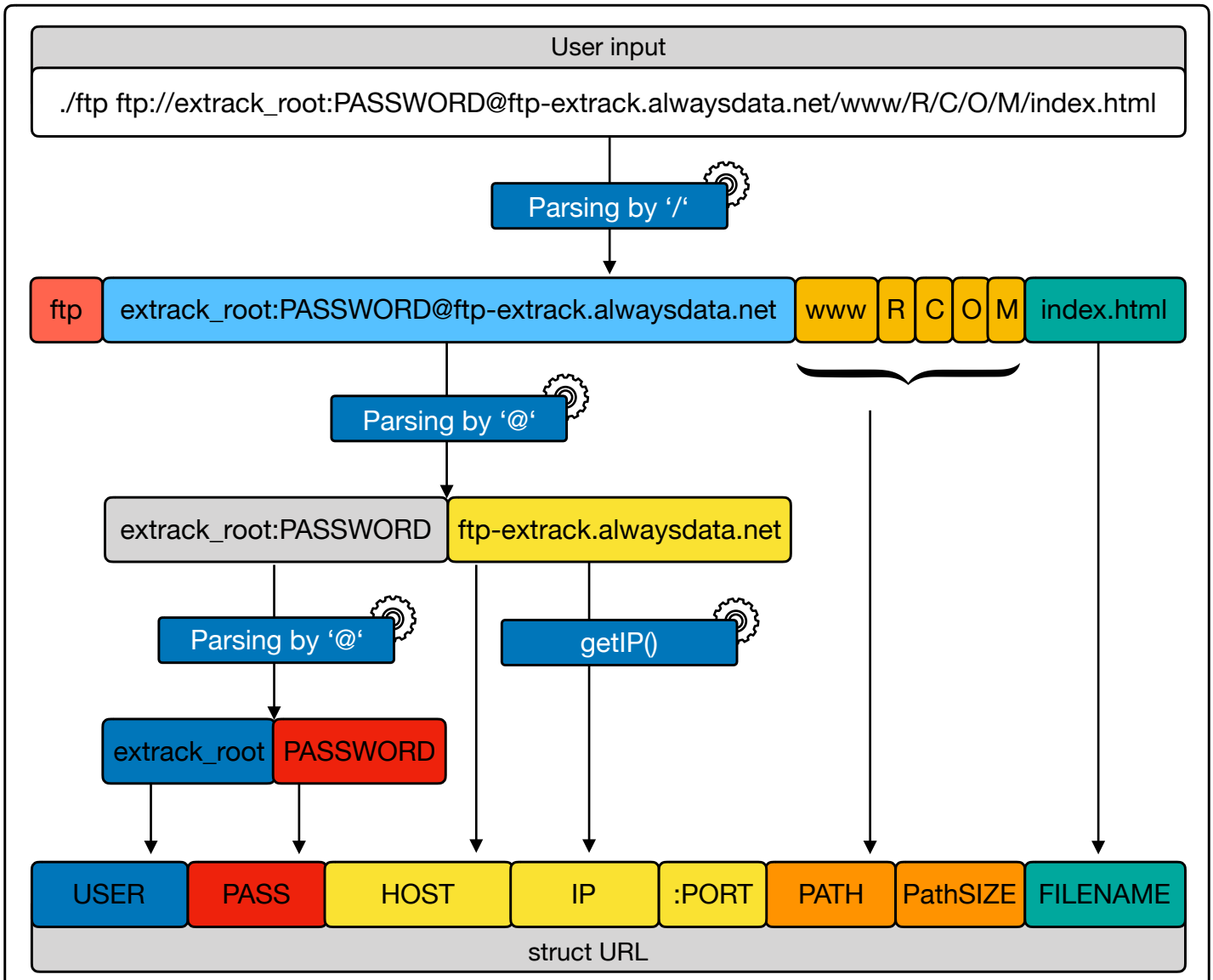


Fig. 7

• This schema describe the function **setURL()**.
• The field Path-size allowed to send the right number of **CWD** command to change the path in the server.

• To create the client we had created some primitiv functions such as **connect**, **disconnect**, **write** and **read** to communicate with the FTP server. We had implemented some code control to check if the server answer well to our requests. We also have an anonyme mode, in case of the user doesn't want to log to the server. So the following commands works :

```
./ftp ftp://ftp.up.pt/pub/scientific/stats/years.json
./ftp ftp://extrack_root:LprosoS@ftp-extrack.alwaysdata.net/www/R/C/O/M/index.html
./ftp ftp://extrack_root@ftp-extrack.alwaysdata.net/www/R/C/O/M/index.html
```

• Contrary to the manual mode with telnet, we can decide when we close the data socket.

1.4 - Example

The following figure (Fig .8) is an example of downloading a file located on **FEUP** server:

```
./ftp ftp://ftp.up.pt/pub/scientific/stats/years.json
```

```
pc-194-65:Lab2 Nowen$ ls
FTP_functions.c         download_bar.h          lab2.pdf
FTP_functions.h         ftp                     rcom2015-ietfAppExp.pdf
Makefile                ftp.c                   rfc959-FTP.txt
Report                  functions.c
download_bar.c          functions.h
pc-194-65:Lab2 Nowen$ make test
./ftp ftp://ftp.up.pt/pub/scientific/stats/years.json

---------------- Anonymous mode

       /-------------------URL-----------------\
       |- User = anonymous                     |
       |- Pass = Nowen@                        |
       |- Host = ftp.up.pt                     |
       |- Ip   = 193.137.29.15                 |
       |- Port = 21                            |
       |- Path = /pub/scientific/stats         |
       |- Size = 4                             |
       |- File = years.json                    |
       \---------------------------------------/


---------------- Connection open !

<-----|         220
|----->         USER anonymous
<-----|         331 Please specify the password.
|----->         PASS Nowen@
<-----|         230 Login successful.
|----->         EPSV
<-----|         229 Entering Extended Passive Mode (|||56382|)

---------------- IP:Port = 193.137.29.15:56382

---------------- Connection open !

|----->         CWD pub
<-----|         250 Directory successfully changed.
|----->         CWD scientific
<-----|         250 Directory successfully changed.
|----->         CWD stats
<-----|         250 Directory successfully changed.
|----->         RETR years.json
<-----|         150 Opening BINARY mode data connection for years.json (98 bytes).

---------------- Byte readed : 98/98

<-----|         226 Transfer complete.

---------------- File received !

---------------- Connection closed !

#./ftp ftp://extrack_root:LprosoS@ftp-extrack.alwaysdata.net/www/R/C/O/M/index.html
#./ftp ftp://extrack_root@ftp-extrack.alwaysdata.net/www/R/C/O/M/index.html
pc-194-65:Lab2 Nowen$ ls
FTP_functions.c         download_bar.h          lab2.pdf
FTP_functions.h         ftp                     rcom2015-ietfAppExp.pdf
Makefile                ftp.c                   rfc959-FTP.txt
Report                  functions.c             years.json
download_bar.c          functions.h
pc-194-65:Lab2 Nowen$
```

The file is not present on the repository

The file is **present** on the repository

We also downloaded the same file with the usual FTP protocol and calculated the **sha256** on both file to check if they are exactly the same.

```
pc-194-65:Lab2 Nowen$ shasum years.json
5ac981f4b12a801aee9837a57398d008292f6303  years.json
pc-194-65:Lab2 Nowen$
```

```
Labs — -bash — 88
pc-194-65:Labs Nowen$ shasum years.json
5ac981f4b12a801aee9837a57398d008292f6303  years.json
pc-194-65:Labs Nowen$
```