File Transfer Protocol

Computer Networks
Bachelors in Informatics and Computing Engineering

3LEIC03_G6

Tiago Rodrigues up201907021@fe.up.pt
Mário Travassos up201905871@fe.up.pt

January 14, 2022

Summary

This report will cover the second project proposed for the Computer Networks Curricular Unit, which had the objective of developing an application that would download files using the FTP (File Transfer Protocol) standard, along with setting up a network between the labs computers.

The application supports both anonymous and authenticated downloads, and the arguments must be valid URL's, without the port.

Introduction

The second project had two main objectives: The development of the download application, and the configuration of the computer network. This report will examine both sections, detailing the architecture of the application and the setup used to launch the network. The application can be used to download files from any FTP server, since it adopts the standard URL syntax.

The report is structured as follows: First, the application is described in detail, breaking down the architecture and the more relevant components. Then, a report of a successful download is made. After that, an analysis of the network will take place. Finally, a set of relevant attachments will be included.

Download Application

Architecture

The application is simple by design. The way it operates internally resembles the way files were retrieved using the same protocol, in the first lab experiment that used **telnet**. It connects to the remote server using a socket, and after the connection is established, sends a set of FTP requests that will retrieve the desired file.

Program flow

The first two requests the program makes are authentication ones, which login the user. If the user does not specify a username and password in the URL, the default is "anonymous" and "pass", for the user and password respectively. If the login is successful, the program then requests the server to enter passive mode. The server's response to this command includes the IP address and the port for the file to be transferred from. The program determines this port, connects to it, and then asks the server to begin the transfer of the file specified in the URL. After the file is requested, the program reads the bytes from the socket, and writes them to the directory it was called from, with the same name as the transferred file. If at any point during execution an error occurs or the response from the server is not the expected one, the program terminates gracefully.

Main Modules

The program is logically divided in the following modules:

- parser: Responsible for decomposing the given URL and dividing it into the relevant tokens, saving them for later use.
- connection: Responsible for establishing a connection with the remote server.
- communication: Responsible for sending commands and reading responses, acting accordingly.
- file: Responsible for reading the incoming file and writing it onto the local machine.

There is also a shared file that includes common definitions like server response codes, and a main file that unites the modules described above.

Case study

Network Configuration and Analysis

Conclusions

References

Annexes

Application Source Code

Configuration Commands

Data Logs