**POLITEHNICA University of Timisoara**
**College of Computer Science and Engineering**

# Download Manager

## [The Red Hot Oompa-Loompas]

**Team members:**
Chereches Vasile, grupa 3.1
Szanto Karoly, grupa 5.2
Tamas-Selicean Domitian, grupa 5.2
Tureanu Amalia Viviana, grupa 5.2

# Table of Contents

## 1. Introduction

The main goal is to realize a client-server based download manager. The client should have the option to download a specified file from a server using multiple streams. Optionally, the user could provide a list of servers (in case one server doesn't have the specified file or the current server goes down, the client has the ability to connect to another server).

## 2. Installation

The software sources are available. Optionally we can provide a shell script that will do for you, but this is without warranty that will work on your system. A prerequisite is to have a C compiler installed on your system.

## 3. Usage

The client takes 4 arguments from the command line:

1. path to a file containing a list of servers to which the client can connect
2. the path of the file to be downloaded
3. the path of the file to be saved on the client side;
4. the number of segments the client will use to download the file;

## 4. Protocol

The communication between the client and the server is done by sending messages in a specific order from one to the other.

When one of the peers has to send a message, they will actually send a pair of messages:

- the first one represents the size of the actual message; it is of type (char *) and has a size of 3 bytes;
- the second one is the actual message: it can be a text message or data message.

After the client has connected to the server, the following sequence of messages will be exchanged:

1. client: “ask <filename>” -> the client sends a request to the server to make sure the server has the file
2. if the server doesn't have the file it will respond with “Error: no such file !” and the client will terminate the communication session; if the server has it, will send “Success: file does exist !”
3. server: “size <filesize> <fileCRC>”
4. after receiving the message no 3., the client will open the request number of segments (processes) that communicate with the same server which will follow exactly 1., 2. and 3.
5. client: “start <offset> <segmentsize>”
6. after receiving the start message, the server will send the corresponding file block
7. the client receives it and the communication is done.

## 5. Description

The client is a multi-server client, with capabilities to resume download.

In order to do this, the client will save each segment in a different file under the following directory tree:

-> <filename>.dir
    \-> <filename>.<index>.tmp
    \-> <filename>.<index>
    \-> <filename>.no

For each new downloaded file, there will be a directory created for it. The name of the directory is the name of the file, and ".dir" appended. In this tree, there will be a "<filename>.no" file created, which contains the number of segments requested by the user.

For each segment, there are 2 different files created:

- <filename>.<index> will store the offset of the segment from the original file and the segment length.
- <filename>.<index>.tmp will contain the actual segment data

After all the segments have been downloaded, the download finalizer is called. It will read the directory, will create the copy of the file, and append in order each segment to it. After the copy is created, the temporary directory is deleted with all its contents.