Redes de Computadores (RCOMP) – 2016/2017

Project 2 proposal

- Network applications development in C and Java languages.

General terms for carrying out the project 2

- Implemented by a team of three or four students, they all must belong to the same PL class.

- Class 2DN students (CDIO-IL) will not be carrying out this project.

- Support for the project is provided in laboratory classes, however, most development effort will be extra classes.

- Project submission is due until **May 21**, at 23.55.

- Submission is made on the Moodle service. Once the deadline expires, submissions are blocked.

- Students with special status are allowed to make the submission until June 11, at 23.55.

- The submission is composed by a written project report and source code files, it should be in ZIP format, using the filename:

{PL Class}_{Student Number}_{Student Number}_{Student Number}[_{Student Number}].zip

, e.g. **2DE_1022222_1033333_1044444_1055555.zip**

- Project 2 final assessment is dependent on the mandatory presentation to be held in the second PL class on the week after the submission deadline (all group members must attend).

- Students with special status submitting after the normal deadline, will have their presentations scheduled subsequently by the PL teacher.

- Students with special status, and wishing to take advantage of the extended deadline, must settle separate groups. They can never integrate groups with members without special status.

Project´s summary

Title: **Peer-to-peer file download**

This project consists of developing a network application, either in C or Java language. It's a peer-to-peer network application, this means it's a single application, but several instances of it will communicate with each other.

The application main objective is providing download access to some local files. Once it starts it will periodically announce to the network a list of local filenames it is willing to serve.

Other application instances, will listen for these announcements, and build a list of available filenames on the network. The application's user may then select one available file to be downloaded.

Detailed requirements and guidelines

The application announces the list of filenames it makes available by sending **UDP datagrams** to the **broadcast address.** Announces are sent to a **fixed pre-settled port number** where all instances are receiving and collecting information. One announcement is to be sent **every thirty seconds.** The working environment is supposed to be a local network and operations are circumscribed to the broadcast domain.

All instances are continuously receiving announces in the pre-settled UDP port number, by doing so, they have a permanently updated list of available files to be presented to the user. Announces must be refreshed within **thirty five seconds**, otherwise corresponding files are to be removed from the list.

One basic issues to be handled is that when an instance makes a broadcast announcement, it will receive its own announcement, of course, it should ignore.

The application user can see the list of available files and select one for download, the **file download** is to be implemented through a **TCP connection** made to the instance that has announced to be providing the file. The TCP port number instances use to accept TCP connections is not pre-settled, each instance will use a dynamically assigned TCP port number.

When the application starts, it will create a TCP socket and dynamically assign to it a port number. The assigned **TCP port will subsequently be included in UDP filenames list announcement**, thus other instances know to which port number they should establish the TCP connection for file download.

Announced filenames are raw, no folder names are to be included. The list of files each instance provides are all files present on a designated folder, for instance a folder named **/shared**. Before sending an announcement the application must read the folder to provide an updated filenames list to other instances, only plain file objects are to be included, folders and other file-system objects should be ignored.

Remember UDP imposes a limit to the datagram size, the application must check if that limit is being surpassed when building the datagram from the filenames list. In that case, it will simply truncate the filename list, and a warning message should be displayed on the application console.

When a user selects a file for download, the downloaded file is to be stored in a designated folder, for instance a folder named **/download**. The original filename is preserved, if the filename already exists it will be overwritten.

One imperative phase prior to developing a network application from scratch is writing an application protocol specification. **This will be considerably taken in account in the project assessment.** Application protocols specification are mandatory to be detailed on the submitted written report.

For this application, two different network application protocols are involved, one UDP based on regard of the announcements, and another TCP based on regard of file download.

Application protocols specification must include all message formats and dialogue phases.

Because different teams will be sharing the same network and possibly nodes (SSH servers), to avoid interferences between them, each will have an assigned **UDP port number.** Assigned ranges for each PL class are as follows:

PL Class	**UDP port numbers range**
2DA	32001 - 32010
2DB	32011 - 32020
2DC	32021 - 32030
2DD	32031 - 32040
2DE	32041 - 32050
2DF	32051 - 32060
2DG	32061 - 32070
2DH	32071 - 32080
2DI	32081 - 32090
2DJ	32091 — 32100
2DK	32101 - 32110
2DL	32111 - 32120
2DN	32121 - 32130
2NA	32131 - 32140
2NB	32141 - 32150

Within each PL class the teacher will then assign a port number to each team.

As far as the user interface is useable, it will not be taken in account for project assessment.

Some additional, non-mandatory, features to be taken into account on project assessment (additional value factors):

- Overcome the limit on the filenames list size due to the UDP datagram size limit. Suggestion: split it into several datagrams.
- Strategies to overcome the broadcast domain confinement. Suggestion: create a list of remote nodes (unicast addresses) and send announcements not only to the local broadcast address, but also to each remote node.

Notice additional features are intended to compensate failures in other items, thus the final project grade will always be less or equal 100%. Also, even without implementing additional features, students may achieve a grade of 100%.

Relative weights on Project 2 assessment	
Written Report general assessment (organization and technical quality)	**10%**
Application protocols specification	**20%**
Network Application code explanation	**10%**
Network Application coding quality	**10%**
Network Application implemented features	**40%**
Network application usability	**10%**
Additional features	**5%**