**Redes de Computadores**

1st Semester 2018/2019
LEIC Alameda

**Programming using the *Sockets* interface**

***"RC Cloud Backup"***

# 1. Introduction

The goal of this project is to develop a simple networking application that allows users to backup the contents of a specified local directory using a cloud service. The cloud storage can be done incrementally, meaning that if some of the files in the directory have the same name, size and date, they don't need to be copied.

For the project, the students will develop: (i) the *user application* (user), and two server applications: (ii) a *Central Server* (CS); and (iii) a *Backup Server* (BS). The user application and the various servers are intended to operate on different machines connected to the Internet.

The operation is as follows.
To perform any cloud backup operation the user first needs to authenticate itself. For this, the user application establishes a TCP session with the CS, which has a well-known URL, providing the user identity (the IST student number – a 5 digit number) and a password (composed of 8 alphanumerical characters, restricted to letters and numbers). The CS will confirm the authentication.

The user can then decide to perform one of the operations:
1 – Register as a new user.
2 – Request the backup of a selected local directory.
3 – List the previously stored directories and files.
4 – Retrieve a previously backed up directory.
5 – Delete the backup for a selected directory.
6 – Delete the registration as a user.

The directories to be backed up should be contained in the current directory from where the user program was invoked, and they should not contain subdirectories. File and directory names should not contain spaces and should have less than 20 characters. In the current directory there should be at most 20 directories, each one with a maximum of 20 files.

When requesting the backup of a local directory the user application sends to the CS a backup request message indicating: the directory name and the number and list of files to be backed up (with name, date, time, size). The CS will check locally (for instance, in a file named "*backup_list.txt*") if this user had previously backed up this directory. If yes, the CS will ask the BS server that contains the backup (for instance, after consulting a file named "*BS_list.txt*") for the files stored in this directory, to find which

of these files should be updated by the user, and replies to the user with the list of the files to be updated, along with the IP address and TCP port number of that BS. The user application then closes the TCP connection with the CS and automatically establishes a new TCP session with the BS, to send the files for backup.
If it is the first time the user requests to backup this directory, the CS identifies one of the available BSs to store the backup. The CS will register the user and its password in a given BS the first time it selects that BS for that user.
The BS will delete a user registration and the corresponding password from its database when the last directory of that user is deleted at this BS.

To retrieve the files from a previous backup, the user application contacts the CS, sending the name of the directory being retrieved. The CS checks which BS contains the backup and returns the corresponding IP and TCP port. The user application then closes the TCP connection with the CS and automatically establishes a new TCP session with the BS, to retrieve the files, after authentication.
To delete the files from a previous backup, the user application contacts the CS, sending the name of the directory to delete. The CS informs the BS containing the backup to delete the information and confirms the deletion to the user application.

The project tests will include one CS server and at least two BS servers, capable of operating on different machines. Several instances of the user application will also be executed simultaneously. For this project the number of BS servers will never exceed 20.

For the implementation, the application layer protocols operate according to the client-server paradigm, using the transport layer services made available by the socket interface, using the TCP and UDP protocols.
The CS accepts user requests using TCP. The communication between the user application and BS servers uses TCP. The communication between the CS and BS servers is done using UDP.

# 2. Project Specification

## 2.1 User Application

The program implementing the user application should be invoked using the command:

`./user [-n CSname] [-p CSport]`,

where:

`CSname` is the name of the machine where the central server (CS) runs. This is an optional argument. If this argument is omitted, the CS should be running on the same machine.

`CSport` is the well-known port where the CS server accepts user requests, in TCP. This is an optional argument. If omitted, it assumes the value 58000+GN, where GN is the group number.

Once the user program is running, it waits for the user to indicate the action to take, notably:

- `login user pass` – following this instruction the user application establishes a TCP session with the CS, sending the user's name `user` (the 5 digit IST student number) and a password `pass` (composed of 8 alphanumerical characters, restricted to letters and numbers), for validation by the CS. The result of the CS validation should be displayed to the user.

- `deluser` – if the user has previously logged in successfully, and there are no directories stored in the backup servers for this user, the username used for the login is removed from the list of valid users in the CS. (Notice that the BS will automatically remove a user when the last of the previously stored directories is deleted).

- `backup dir` – following this instruction the user application should contact the CS, sending the name of the directory, `dir`, located at current working directory (from where the user application was invoked), and the list of files (and their details) to be backed up. The CS will reply with the IP and TCP port number of the BS selected for the backup, as well as the list of files (may be a subset of the original list) to be sent by the user application to the BS. The user application then closes the TCP connection with the CS and establishes a new TCP connection with the BS. After authentication (using the user and password previously provided by the user) the user application transfers the listed files to the BS. When completed, the directory name and the list of backed up files will be displayed to the user.

- `restore dir` – following this instruction the user application requests the CS server to send the contents of the directory named `dir`, previously backed up, and stores the received files in the local directory with the same name. The CS checks which BS contains the backup and returns the corresponding IP and TCP port. The user application then closes the TCP connection with the CS and automatically establishes a new TCP session with the BS, to retrieve the files, after authentication. The directory name and the list of file names will be displayed to the user, as well as the success status of the operation.

- `dirlist` – following this instruction the user application should contact the CS, asking for a list of directories already stored in the backup system. The CS returns the information, which is displayed to the user.

- `filelist dir` – following this instruction the user application should contact the CS, sending the name of a directory `dir` stored in the backup system, asking for the list of files stored in that directory. The CS returns the information, which is displayed to the user. (Each directory contains a maximum of 20 files.)

- `delete dir` – following this instruction the user application requests the CS server to delete the backup of directory `dir`. The CS forwards the request to the appropriate BS. The BS will automatically remove a user from its list when the last of the previously stored directories is deleted. After completing the deletion of the directory, the CS informs the user application. The information is displayed to the user.

- `logout` – following this instruction the user application should forget the information about the previous logged in user. This allows using the application with a different username.

- `exit` – the user application terminates.

The login command only serves for the user to keep local information about its username and password and the logged state.
Since the CS and BS servers don't keep the state of each user across sessions, each time the user issues a command the user application must send again the `username+password` to the CS or to the BS.

## 2.2 *Central Server* (CS)

The program implementing the *Central Server* should be invoked using the command:

```
./CS [-p CSport],
```

where:

`CSport` is the well-known port where the CS server accepts requests, in TCP. This is an optional argument. If omitted, it assumes the value 58000+GN, where GN is the number of the group.

The central server (CS) makes available a server with well-known port `CSport`, supported in TCP, to answer user requests for the file processing tasks.
The central server (CS) makes available another server, supported in UDP, also with well-known port `CSport`, to manage the registration and deregistration requests sent by BS servers. Information about the user directories backed up in each BS server is kept on the CS (in memory or in a file, such as "*backup_list.txt*").
The CS server outputs to the screen the received requests (user, directory and type of request) and the IP and port originating those requests.
Each received request should start being processed once it is received.

## 2.3 Backup Server (BS)

The program implementing the *Backup Server (BS)* should be invoked using the command:

```
./BS [-b BSport] [-n CSname] [-p CSport],
```

where:

`BSport` is the well-known port where the BS server accepts TCP requests from the user application. This is an optional argument. If omitted, it assumes the value 59000.

`CSname` is the name of the machine where the central server (CS) runs. This is an optional argument. If this argument is omitted, the CS should be running on the same machine.

`CSport` is the well-known port where the CS server accepts requests. This is an optional argument. If omitted, it assumes the value 58000+GN, where GN is the group number.

The backup server (BS) makes available a TCP server with well-known port `BSport`, to answer user requests for the file processing tasks, and a UDP server, also with well-known port `BSport`, to handle messages coming from the CS server for processing.
When started, the BS communicates with the CS, in UDP, for the registration of its availability, using UDP. When terminated (with `^C`) the BS unregisters itself with the CS. The CS can ask the BS to register a user, to check a directory contents or to delete a directory.
The BS server provides services to the user application using the TCP protocol with port `BSport`. The BS accepts user application requests in TCP for file backup tasks.
The BS outputs to the screen the received user requests (user, directory and type of request).

# 3. Communication Protocols Specification

## 3.1 *User–CS* Protocol (in TCP)

The user application interacts with the CS server in TCP according to the following request and reply protocol messages:

**a)** `AUT` *`user pass`*
Following the `login` instruction, the user application sends the user's name *`user`* and password *`pass`*, for validation by the CS server.

**b)** `AUR` *`status`*
In reply to an `AUT` request the CS server replies in TCP indicating the status of the authentication request:

- if the `AUT` request was successful (valid username and password) the *`status`* is "`OK`";
- if the username exists but the password is incorrect the *`status`* is "`NOK`";
- otherwise a new user is created, with the supplied password, and the *`status`* will be "`NEW`".

**c)** `DLU`
Following the `deluser` instruction, the user application informs the CS server that the username, used in the previous `login` instruction, should be removed from the list of valid users if there is no information stored for this user.

**d)** `DLR` *`status`*
In reply to a `DLU` request the CS server replies in TCP indicating the status of the deletion request. If the `DLU` request was successful (username was removed) the *`status`* is "`OK`"; otherwise, notably if the user still has information stored, the *`status`* is "`NOK`".

**e)** `BCK` *`dir N (filename date_time size)*`*
Following the `backup` instruction, the user application sends a request to the CS asking to backup the directory *`dir`*, which contains *`N`* files, followed by the details of each one: *`filename`*, *`date_time`* and *`size`*. *`date_time`* is given in the format `dd.mm.yyyy hh:mm:ss`.

**f)** `BKR` *`IPBS portBS N (filename date_time size)*`*
The CS reply to a `BCK` request includes the IP address *`IPBS`* and TCP port number *`portBS`* of the BS server that will store the requested backup. Moreover, the CS also returns the number *`N`* and details *`(filename date_time size)*`* of the files to be sent by the user application to the BS (which may be a subset of the original list).
This reply is provided after checking if this is the first time a backup of this user is being sent to the selected BS:

- If the user in not yet registered in this BS, then the CS exchanges *`LSU-LUR`* messages with the BS, and the full list of files is returned.
- Otherwise, the CS exchanges *`LSF-LFD`* messages with the BS, to receive the list of missing or changed files in the backup.

After receiving this protocol message, the user application closes the TCP connection with the CS and automatically establishes a new TCP session with the identified BS. It then sends an `AUT` message to the BS and, if the `AUR` reply status is "`OK`", the user application sends the `UPL` message to upload the *`N`* files to the BS server.
If the `BCK` request cannot be answered (e.g., no BS available) the reply will be "`BKR EOF`". If the `BCK` request is not correctly formulated the reply is "`BKR ERR`".

**g)** `RST` *`dir`*
Following the `restore` instruction, the user application sends a request to the CS asking to retrieve the backed up contents of directory *`dir`*.

**h)** `RSR` *`IPBS portBS`*
The CS reply to a `RST` request includes the IP address *`IPBS`* and TCP port number *`portBS`* of the BS server that contains the files to be retrieved.
After receiving this protocol message, the user application closes the TCP connection with the CS and automatically establishes a new TCP session with the identified BS. It then sends an `AUT` message to the BS and, if the `AUR` reply status is "`OK`", the user application sends to the BS server the `RSB` message to retrieve the files.
If the `RST` request cannot be answered (e.g., no BS available) the reply will be "`RSR EOF`". If the `BCK` request is not correctly formulated the reply is "`RSR ERR`".

**i)** `LSD`
Following the `dirlist` instruction, the user application request the CS server to return the list of directories already backed up by this user.

**j)** `LDR` *`N (dirname)*`*
In reply to a `LSD` request the CS server replies in TCP indicating the list of *`N`* directories, *`(dirname)*`*, previously backed up by this user (e.g., after consulting the file "*backup_list.txt*"). If the `LSD` request was unsuccessful *`N`* takes value "`0`".

**k)** `LSF` *`dir`*
Following the `filelist` instruction, the user application requests the CS server to return the list of files stored in the previously backed directory *`dir`*.

**l)** `LFD BSip BSport` *`N (filename date_time size)*`*
In reply to a `LSF` request the CS contacts the BS containing the backup of the selected user directory using the `LSF` request. After the reply is received from the BS, the CS replies to the user with the ip and TCPport of the BS containing the backup (to be displayed to the user), the number of files in the directory, *`N`*, and the list of files, for each including: *`filename`*, *`date_time`* and *`size`*. *`date_time`* is given in the format `dd.mm.yyyy hh:mm:ss`.
If the `LSF` request cannot be answered the reply will be "`LFD NOK`".

**m)** `DEL` *`dir`*
Following the `delete` instruction, the user application sends a request to the CS asking to delete the backed up contents of directory *`dir`*.

**n)** `DDR` *`status`*
The CS reply to a `DEL` request is a confirmation of the delete operation. Before replying, the CS asks the BS containing the backup to delete the specified directory using the `DLB` message.
If the `DEL` request was successful the *`status`* will be "`OK`", otherwise the *`status`* will be "`NOK`".

If an unexpected protocol message is received, the reply will be "`ERR`".
In the above messages the separation between any two items consists of a single space.
Each message of request or reply ends with the character "`\n`".

Each of the messages `DLU`; `BCK`; `RST`; `LSD`; `LSF`; `DEL`, must be preceded by an authentication (`AUT`) message within the same TCP session.
For instance, if the user wants to list the files of a given directory, the sequence of messages will be:

| | |
|---|---|
| `AUT user pass` | (User to CS) |
| `AUR OK` | (CS to User) |
| `LSF dir` | (User to CS) |
| `LFD …` | (CS to User) |

## 3.2 *User–BS* Protocol (in TCP)

The user application interacts with the BS server in TCP according to the following request and reply protocol messages:

**a)** `AUT` *`user pass`*
When needing to communicate with the BS the user application needs to first authenticate itself. For this purpose the user application sends the user's name *`user`* and password *`pass`*, for validation by the BS server.

**b)** `AUR` *`status`*
In reply to an `AUT` request the BS server replies in TCP indicating the status of the authentication request. If the `AUT` request was successful the *`status`* will be "`OK`", otherwise the *`status`* will be "`NOK`".

**c)** `UPL dir` *`N (filename date_time size data)*`*
Following the `BKR` protocol message, the user application sends the *`N`* files from directory *`dir`* to the BS. The transmission includes, for each file, its details: *`filename`*, *`date_time`*, *`size`* and the contents of the file: *`data`*. *`date_time`* is given in the format `dd.mm.yyyy hh:mm:ss`.

**d)** `UPR` *`status`*
In reply to a `UPL` request, the BS server replies in TCP indicating the status of the file transfer. If the `UPL` request was successful the *`status`* will be "`OK`", otherwise the *`status`* will be "`NOK`".

**e)** `RSB` *`dir`*
Following the `RSR` protocol message, the user application request the BS server to send the files from directory *`dir`*, which will be stored in the local directory specified in the *`restore`* command.

**f)** `RBR` *`N (filename date_time size data)*`*
Following the `RSB` protocol message, the BS server sends the user application the *`N`* files from directory *`dir`*. The transmission includes, for each file, its details: *`filename`*, *`date_time`*, *`size`* and the contents of the file: *`data`*.
If the `RSB` request cannot be answered (e.g., directory not found) the reply is "`RBR EOF`". If the `RSB` request is not correctly formulated the reply is "`RBR ERR`".

If an unexpected protocol message is received, the reply will be "`ERR`".
Separation between two items is a single space. Messages end with the character "`\n`".

Each of the messages `UPL`; `RSB`, must be preceded by an authentication (`AUT`) message within the same TCP session.
For instance, if the user wants to list the files of a given directory, the sequence of messages will be:

| | |
|---|---|
| `AUT user pass` | (User to BS) |
| `AUR OK` | (BS to User) |
| `UPL …` | (User to BS) |
| `UPR OK` | (BS to User) |

## 3.3 *BS* – *CS* Protocol (in UDP)

When the BS starts/ends it needs to register/deregister itself with the CS server. Also the registration of a user, checking a directory contents and delete operations can be asked to the BS by the CS.
Communication uses the UDP protocol and includes the following requests and replies:

**a)** `REG` *`IPBS portBS`*
The BS informs the CS that it is now running and available to accept backup operations on IP address *`IPBS`* and TCP port number *`portBS`*.

**b)** `RGR` *`status`*
The CS confirms (*`status = OK`*) or declines (*`status = NOK`*) the `REG` message. If there is a protocol (syntax) error the answer will be "`RGR ERR`".

**c)** `UNR` *`IPBS portBS`*
The BS informs the CS that it stopped operating and therefore *`IPBS`* and *`portBS`* should be removed from list of available BS servers.

**d)** `UAR` *`status`*
The CS confirms (*`status = OK`*) or declines (*`status = NOK`*) the `UNR` message. If there is a protocol (syntax) error the answer will be "`UAR ERR`".

**e)** `LSF` *`user dir`*
Following the `LSF` request from the user application to the CS server, the CS asks the BS containing the backup of directory *`dir`* to return the list of stored files.

**f)** `LFD` *`N (filename date_time size)*`*
In reply to a `LSF` request made by the CS, the BS sends the list of *`N`* files in directory `dir` of user `user`, for each including: *`filename`*, *`date_time`* and *`size`*. *`date_time`* is given in the format `dd.mm.yyyy hh:mm:ss`.

**g)** `LSU` *`user pass`*
Whenever the CS needs to inform a BS of a user to be added to its users list, the respective username *`user`* and password *`pass`* are sent to the BS.

**h)** `LUR` *`status`*
The BS confirms (*`status`* = *OK*) or declines (*`status`* = *NOK*) having updated its user list. If there is some error the answer will be "`LUR ERR`".

**i)** `DLB` *`user dir`*
The CS server, after receiving the protocol message `DEL` from the user application, asks the BS to delete the directory `dir` of user `user`.

**j)** `DBR` *`status`*
In response to the protocol message `DLB`, the BS server informs the CS about the status of removing directory `dir` of user `user`. If the `DLB` request was successful the *`status`* will be "`OK`", otherwise the *`status`* will be "`NOK`".

If an unexpected protocol message is received, the reply will be "`ERR`".
Separation between two items is a single space. Messages end with the character "`\n`".

# 4. Development

## 4.1 Development and test environment

Make sure your code compiles and executes correctly in the development environment available in lab LT5.

## 4.2 Programming

The operation of your program should be based on the following set of system calls:

- Computer name: `gethostname()`.
- Remote computer IP address from its name: `gethostbyname()`.
- UDP server management: `socket()`,`bind()`,`close()`.
- UDP client management: `socket()`,`close()`.
- UDP communication: `sendto()`,`recvfrom()`.
- TCP server management: `socket()`, `bind()`, `listen()`, `accept()`, `fork()`, `close()`.
- TCP client management: `socket()`,`connect()`,`close()`.
- TCP communication: `write()`,`read()`.

## 4.3 Implementation notes

Developed code should be adequately structured and commented.
The `read()` and `write()` system calls may read and write, respectively, a smaller number of bytes than solicited – you need to ensure that your implementation still works correctly.
Both the client and server processes should terminate gracefully at least in the following failure situations:

- wrong protocol messages received from the corresponding peer entity;
- error conditions from the system calls.

# 5 Bibliography

- W. Richard Stevens, Unix Network Programming: Networking APIs: Sockets and XTI (Volume 1), 2nd edition, Prentice-Hall PTR, 1998, ISBN 0-13-490012-X, chap. 5.
- D. E. Comer, Computer Networks and Internets, 2nd edition, Prentice Hall, Inc, 1999, ISBN 0-13-084222-2, chap. 24.
- Michael J. Donahoo, Kenneth L. Calvert, TCP/IP Sockets in C: Practical Guide for Programmers, Morgan Kaufmann, ISBN 1558608265, 2000
- On-line manual, `man` command
- Code Complete - http://www.cc2e.com/
- http://developerweb.net/viewforum.php?id=70

# 6 Project Submission

## 6.1 Code

The project submission should include the source code of the programs implementing the *user*, the *CS server* and the *BS server*, as well as the corresponding *Makefile*.
The makefile should compile the code and place the executables in the current directory.

## 6.2 Auxiliary Files

Together with the project submission you should also include any auxiliary files needed for the project operation together with a *readme.txt* file.

## 6.3 Submission

The project submission is done by e-mail to the lab teacher, **no later than October 12, 2018, at 23:59 PM**.
You should create a single `zip` archive containing all the source code, makefile and all auxiliary files required for executing the project. The archive should be prepared to be opened to the current directory and compiled with the command `make`.
The name of the archive should follow the format: **`proj_"group number".zip`**

# 7 Questions

You are encouraged to ask your questions to the teachers in the scheduled foreseen for that effect.

# 8 Open Issues

You are encouraged to think about how to extend this protocol in order to make it more generic. For instance, how could a redundant backup server be created?