Redes de Computadores

1st Semester 2016/2017
LEIC Alameda

Programming using the *Sockets* interface

"RC Translate"

Language translation is being integrated into many applications. Often the operation of the translation system relies on a network connection to the translation engine, which receives a translation request and then returns the translation result. Recent systems accept not only textual input but also images of text, returning the result in the same format (text or image) as the input request.

1. Introduction

The goal of this project is to develop a simple networking application that allows users to submit: (i) textual translation requests; or (ii) image translation requests, and to receive back from the server the corresponding replies.
For this purpose, the students will develop the user application and two server applications: (i) one unique centralized Translation Contact Server (TCS); and (ii) several instances of Translation Servers (TRS), each dedicated to translate from a given language to Portuguese. The user application and the servers may operate on different machines.
The user application first contacts the TCS, which has a well-known URL, asking for a list of the available translation languages. For each language in the list, there is a TRS server instance (previously registered with the TCS) that will answer translation requests.
When the user issues a translation request, the TCS is contacted by the user application asking for the IP address and port of the TRS server to be contacted for the translation.
The appointed TRS is then automatically contacted by the user application, which identifies whether a textual (*t*) or a file (*f*) base translation is requested. When the TRS receives a file with imaged text, that file is stored on a local directory and the reply is another file with the imaged text translation, obtained also from a local directory.
The language identifiers are the Portuguese words of the language names, and will have a maximum length of 20 characters. The directory where the TRS runs should contain two text files, "*text_translation.txt*" and "*file_translation.txt*".
Each line of the *text_translation.txt* file contains (separated by spaces):

- a first string in the TRS language.
- a second string, corresponding to the first one translated to Portuguese.

Each line of the *file_translation.txt* file contains (separated by spaces):

- a first string identifying the received filename, containing the image of the text in the TRS language.
- a second string identifying the name of the file with the image of the text translated to Portuguese.

The TCS server will contain a list (e.g., stored in memory or in a file, e.g. "*languages.txt*"), where the TCS will update the information related to the available translation languages and the IP and port numbers where the corresponding TCP servers are operating. This is information is provided to the TCS when each TRS is started.
Students should develop the three components of the project: (i) TCS server; (ii) TRS server; (iii) user application.
The project tests will include one TCS server and at least two TRS servers, capable of operating on different machines. Several instances of the user application will also be executed simultaneously. Each TRS server should contain the translation of at least 20 words and 5 files with sentences' images.

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 TCS accepts user requests and communicates with TRS servers using UDP. The user can issue translation requests to the TRS servers after establishing a TCP connection. The TRS replies are also sent back to user application using the TCP connections.

2. Project Specification

2.1 User Application

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

`./user [-n TCSname] [-p TCSport]`,

where:

TCSname is the name of the machine where the translation contact server (TCS) runs. This is an optional argument. If this argument is omitted, the TCS should be running on the same machine.

TCSport is the well-known port where the TCS server accepts user requests, in UDP. 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:

- `list` – following this instruction the user application should contact the TCS, using the UDP protocol, asking for the list of available translation languages. In its reply the TCS provides the list of available translation languages (*L1, L2, ... Ln*), eventually by checking the "languages.txt" file. These languages will be displayed to the user as a numbered list.
- `request` *n t* W_1 W_2 ... W_N or
  `request` *n f filename* – following this instruction the student application communicates in UDP with the TCS server, providing the desired language name (L_n), asking for the IP address and TCP port number of the TRS server. Once the reply from the TCS server is received, then the user application automatically communicates in TCP with the identified TRS server. If a textual translation (*t*) is requested the user provides the list of *N* words (W_1 W_2 ... W_N) to be translated. If an image file translation (*f*) is requested the user sends the contents of the file named *filename*
  The TRS replies with a list of translated words, that should be shown in the users' screen, or with the transmission of the image file with the translation, and the user should be notified when the file transfer concludes.
  It can be assumed that no more than 10 words are sent in each `request` instruction.
- `exit` – the user application terminates.

2.2 *Translation Contact Server* (TCS)

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

`./TCS [-p TCSport]`,

where:

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

The translation contact server (TCS) makes available a server with well-known port *TCSport*, supported in UDP, to answer user requests for available translation languages and the respective translation servers (TRS). The TCS also accepts registration and deregistration requests sent by TRS servers, storing information about the available languages and respective translation servers in a list (eventually stored in a file, e.g. "*languages.txt*").
The TCS server outputs to the screen the received requests and the IP and port originating those requests.
Each received request should be responded immediately.

2.3 Translation Server (TRS)

The program implementing the *Translation Server (TRS)* should be invoked using the command:

`./TRS language [-p TRSport] [-n TCSname] [-e TCSport],`

where:

`language` is the language from which this server translates into Portuguese.

TRSport is the well-known port where the TRS server accepts requests from users. This is an optional argument. If omitted, it assumes the value 59000.

TCSname is the name of the machine where the translation contact server (TCS) runs. This is an optional argument. If this argument is omitted, the TCS should be running on the same machine.

TCSport is the well-known port where the TCS server accepts requests, in UDP. This is an optional argument. If omitted, it assumes the value 58000+GN, where GN is the group number.

The TRS server provides services to the user application using the TCP protocol with port *TRSport*.
The TRS accepts user requests for translation of a set of words, or of the text image included in a file. For the translations the TRS consults the two text files, "*text_translation.txt*" and "*file_translation.txt*", as mentioned in page 1.
The TRS outputs to the screen the received requests and the IP and port which originated those requests.

3. Communication Protocols Specification

3.1 *User–TCS* Protocol (in UDP)

The user program, following the `list` instruction interacts with the TCS server in UDP according to the following requests and replies:

a) `ULQ`
Following the `list` instruction, the user application sends a user list query message to the TCS server, asking the list of available translation languages in the TRS servers.

b) `ULR` n_L *L1 L2 … Ln*$_L$
In reply to a `ULQ` request the TCS server replies in UDP indicating the number (n_L) and the list of available languages (*L1 L2 … Ln*$_L$), where *Ln* is a string of characters specifying the name of the language number `n`, in Portuguese. These languages are displayed to the user as a numbered list.
The `ULR` reply is sent by the TCS immediately after receiving the `ULQ` request.
If the `ULQ` request cannot be answered (e.g., no language servers available) the reply will be "`ULR EOF`". If the `ULQ` request is not correctly formulated the reply is "`ULR ERR`".
To simplify, assume that n_L value is at most 99, and that each language name (*Ln*) contains no more than 20 characters.

c) `UNQ` *Ln*
Following the `request` instruction, the user sends a request to the TCS asking for the translation server (TRS) address details. This request contains the identification of the desired language (*Ln*).

d) `UNR` *IPTRS portTRS*
In reply to a `UNQ` request, the TCS server replies indicating the IP address (*IPTRS*) and the TCP port number (*portTRS*) where the Translation Server (TRS) is available. The reply `UNR` is sent by the TCS immediately after receiving the `UNQ` request.
If the `UNQ` request cannot be answered (e.g., invalid language name) the reply will "`UNR EOF`". If the `UNQ` request is not correctly formulated the reply is "`UNR 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`".

3.2 *User–TRS* Protocol (in TCP)

For the user to ask for a translation (following the `request` instruction), a connection with the selected TRS server is established in TCP.
The communication protocol includes the following requests and replies:

a) `TRQ` *t N* W_1 W_2 … W_N or `TRQ` *f filename size data*
The user requests the TRS to translate the provided text words (*t*) or image file (*f*).
In the first case (*t*), *N* specifies the number of words that are being sent. W_1 W_2 … W_N are strings specifying each of the *N* words, separated by spaces. It can be assumed that each word (*Wn*) contains no more than 30 characters. No more than 10 words are sent in each `request` instruction.
In the second case (*f*), the file with *filename*, specified in the `request` instruction is transmitted. The file *size* in Bytes is sent, followed by the corresponding *data*.

b) `TRR` *t N* W_1 W_2 … W_N or `TRR` *f filename size data*
Following a `TRQ` request the response of the TRS is the translated text (*t*) or image file (*f*).
In the first case (*t*), *N* specifies the number of words in the translation. W_1 W_2 … W_N are strings specifying each of the *N* words, separated by spaces. Each word (*Wn*) contains no more than 30 characters and no more than 10 words are sent.
In the second case (*f*), the identified translation file *filename* is transmitted. The file *size* in Bytes is sent, followed by the corresponding *data*.
If the `TRQ` request is not well formulated the answer will be "`TRR ERR`".
If no translation is available the answer will be "`TRR NTA`".

Separation between two items is a single space. Messages end with the character "`\n`".

3.3 *TRS – TCS* Protocol (in UDP)

When the TRS starts/ends it needs to register/deregister itself with the TCS server.
Communication uses the UDP protocol and includes the following requests and replies:

a) `SRG` *language IPTRS portTRS*
The TRS informs the TCS that it implements translation for *language*, and operates using IP address (*IPTRS*) and TCP port number (*portTRS*).

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

c) `SUN` *language IPTRS portTRS*
The TRS informs the TCS that it stopped translating *language*, and that *IPTRS* and *portTRS* should be removed from list of available languages.

d) `SUR` *status*
The TCS confirms (*status* = *OK*) or declines (*status* = *NOK*) the `SUN` message. If there is a protocol (syntax) error the answer will be "`SUR 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()`, `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 *TCS server* and the *TRS 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 all auxiliary files needed for the project operation (such as the various "*text_translation.txt*" and "*file_translation.txt*" files).

6.3 Submission

The project submission is done by e-mail to the lab teacher, **no later than October 14, 2016, 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 the user offer new pairs of words and their translations to be included in the "*text_translation.txt*" and/or "*file_translation.txt*" files of a TRS?