BLG 433

Computer Communications

Message Board Application

Cemal Turkoglu

150140719

01.04.2018

Message Board Application

1) Introduction

In this project a simple protocol is implemented for message board application (MBA). n this application, a message board server (MBS) waits on a well-known UDP end-point (i.e., IP address and port number). Each message board client (MBC) can register itself to the MBS, create new message board(s), get the messages on a message board, and add messages to a message board.

2) Development Environment

The project is developed in Linux Mint x64, via Python 2.7. Project includes Server.py and Client.py source codes. Only requirement is python socket package. To run the server

python server.py

A client needs server running to get servise.To run client

python client.py

3) Data Structures

In the server part of application there are 2 important data structures.

- RegisteredUsers dictionary which keeps track of users registered to system. Format is:

  registereUsers{username, password}

- messageBoards variable is a dictionarys of dictionary data structure. Format is:

  messageBoards:{messageboard:{message,user,..},..}

  here user is the one who added message to message board.

4) Algorithms

4.1)Client

create a udp socket

ask username and passwords from user to registered

send REG command to server

while not exit

ask user another operation

operations:

1) Create message board

ask mb name from user

send CREATE command

2) List message boards

send LIST:BOARDS command to server

list boards

3) Add a message to a message board

get necessary information from user

send ADD command to server

reflect errors to user

4) List messages on a message board

get necessary information from user

send LIST:MESSAGES command to server

get messages from server 1 by 1

receiving -1 as MESSAGE_NUM means last message has arrived, do not wait packages anymore

reflect errors to user

5) Add another user

// this is extra operation to register another user

sending REG command to servers as above

0) Exit

Exiting the program

4.2) Server

Create a UDP Socket

Wait for request from user forever

When a request came, there can be 5 different command in it

1) REG

check registeredUser dictionary to add new user

send ACCEPT if it is not already in the list

otherwise send REJECT message to Client

2) CREATE

check messagesBoards dictionary to add new mb

if it is possible initialize a message board with given name

and send ACCEPT to client

otherwise send REJECT messages

3) LIST:BOARDS

iterate over messageBoards and send available mbs to client

4) ADD

control messageBoard and registeredUser to validate given username/password and message board name

if it is valid, add new message to regarding messageBoard with the user who is adding the messages

otherwise send regarding REJECT code to client

5) LIST:MESSAGES

control messageBoard and registeredUser to validate given username/password and message board name

if it is valid, iterate over regarding messageBoard and send messages and users who added the message as 1 by 1.

sent MESSAGE_NUM=-1 as sentinel to warn the client that all messages are sent and no more packages will be sent.

5) Conclusion

We have implemented simple UDP Server and Client Application talks each other by some pre determined protocol. It is easy and fast to transport messages via UDP protocol.

UDP is a stateless protocol and server and client does not need a handshaking. Packages might be lost on the way and there is no validation mechanism to check whether destination has received successfully or not. On the other hand it is very fast protocol compared to TCP.