Department of Electrical and Computer Engineering
Concordia University

Communication Networks and Protocols - COEN 445

Fall 2016

Project: Publish and Chat

Designed by: Ferhat Khendek

1. Introduction

The project consists of implementing in, Java, C++, or C#, a Publish and Chat System (PCS), over UDP. The description of the protocols to implement is given in the following section while the requirements are stated in Section 3.

2. Publish and Chat

The Publish and Chat System (PCS) (Fig. 1.) consists of several clients/users (at least 10) and several servers (at least 2). The PCS allows for the clients to register and publish their location (IP Address and Port#), their status (On/Off), the users allowed to chat with them and be given the published information. Every user is allowed to register and publish information with one server only. A server has a maximum capacity of 5 users. Users can change their published information as they wish. When User X wants to chat with User Y, X has first to retrieve the information Y has published. The users communicate with a very simple protocol over UDP. The system is described in the following subsections.

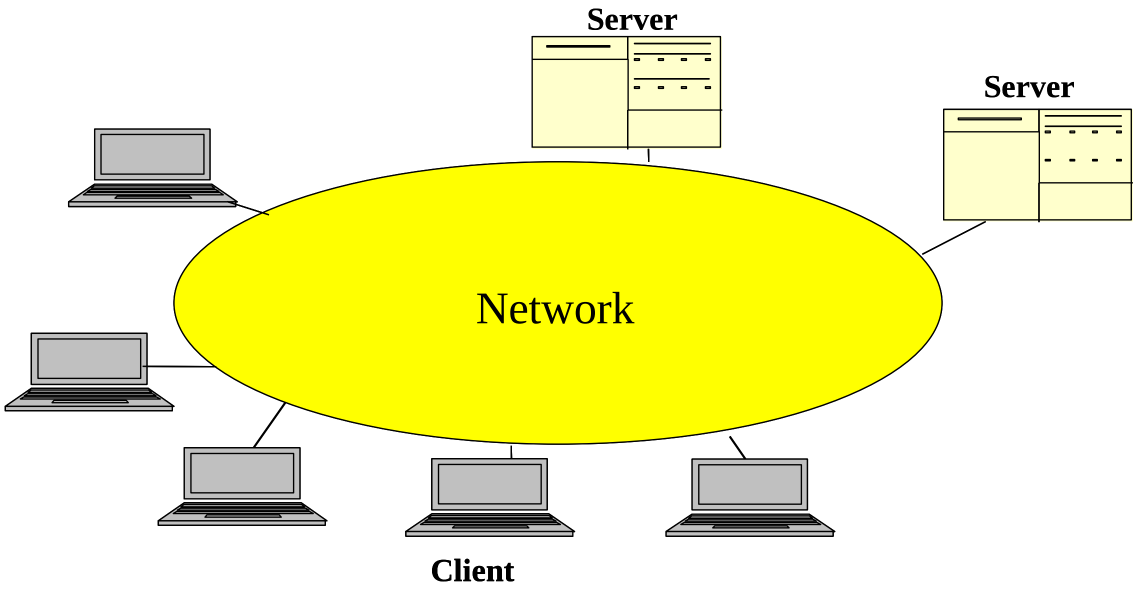

Fig. 1. Overall System

2.1. Registration

A client (or a User) has to register with one server before he can publish or get information about other users in the system. A User cannot register with more than one server. For registering a User has to send his Name and IP Address to the Server (see REGISTER

REGISTER	RQ#	Name	IP Address

Message).
REGISTER Message

A Server cannot have more than 5 users registered at a time. In other words, a Server has a maximum capacity of 5 users. Upon reception of this message a Server, can accept or refuse the new registration. If the number of registered users in less than 5, then the new user is

REGISTERED	RQ#

accepted and the following message is sent back from the Server to the User:
REGISTERED Message

If the Server cannot accept the new registration, it sends back to the User the IP Addresses and Port# of the next server it knows about using the following message:

REGISTER-DENIED	RQ#	IP Address	Port#

REGISTER-DENIED Message

The RQ# is used to refer to which REGISTER message this denial corresponds to.

Upon reception of REGISTER-DENIED, the User will try to register with the other Server following the same process. However if the User has already tried with this Server before, then he will not try again and just assume all the servers are overloaded to the maximum capacity and give up for now.

Here we are assuming the servers know each other and are organized in a circular list for providing the user with the next one. For instance, Server 1 → Server 2 → Server 3 → Server 1 … (This is done at the initialization of the servers.)

2.2. Publishing

After a successful registration with a Server a User can publish some information about himself. The Server has to keep the information about Users in a table, which also has to be saved in a file, in order to reload it when needed.

In order to communicate with other users, a User has to publish some information. It has to publish the Port # it is listening to, its availability for communication (i.e. Status, which could be On or Off), list of Users the Server is allowed to give this information to.

PUBLISH	RQ#	Name	Port#	Status	List of Names

PUBLISH Message

The Server is not allowed to give any information for Users not listed in “List of Names”.

To acknowledge this publishing request and the publication of the information, the Server responds with the following message, after a correct update of its Users table:

PUBLISHED	RQ#	Name	Port#	Status	List of Names

PUBLISHED Message

This message carries back all the information sent by the User for confirmation purpose.

If for some reasons the information cannot be published, because it is damaged (e.g. Status is neither On or Off, etc.), Server responds with message:

UNPUBLISHED	RQ#

UNPUBLISHED Message

Of course after receiving such a message the User will try to resend the PUBLISH message again.

The PUBLISH message is also used to change the published information. For instance a User can send a new PUBLISH message to change its status, the list of names, or the Port#, or any combination of these parameters.

A User never keeps track locally of the information it has published in the Server. From time to time, a User can send an INFORMReq message to the Server to get this information. The INFORMReq message carries the RQ# and the Name of the user.

INFORMReq	RQ#	Name

INFORMReq Message

The Server responds with

INFORMResp	RQ#	Name	Port#	Status	List of Names

INFORMResp Message

RQ# identifies the request the Server is responding to.

2.3. Retrieving information about Users

Before communicating with a User Y, User X has to find the corresponding information. This information will tell where User Y is (IP address), the port it is listening to. Of course User X will be given this information only if he is allowed to talk to User Y according to the information User Y has published.

To find the information about User Y, Users X sends a FINDReq message to its Server, providing the name of the user it would like to communicate with.

FINDReq	RQ#	Name

FINDReq Message

If the User Y has the same server as X and X is in the allowed list of users of Y, and the status

FINDResp	RQ#	Name	Port#	IP Address

of Y is “On”, then the Server will respond with:
FINDResp Message - 1

The FINDResp message contains the name (say Y), the IP address of Y and its Port#. For the communication between X and Y, see Section 2.4.

If the User Y has the same server as X and X is in the allowed list of users of Y, and the status of Y is “off”, then the Server will respond with:

FINDResp	RQ#	Name	Off

FINDResp Message – 2

In this case, X understands that he can talk to Y, but Y is off for now. He can try later on to find this information again.

In the case where Y is registered with the same server as X, but X is not in the list of users allowed to receive the information about Y, the Server responds with:

FINDDenied	RQ#	Name

FINDDenied Message

With a FINDDenied response, X stops and never gets in touch with Y. Of course he can try for other users, and try later on for Y.

In the case where Y is not registered with the server of X, the later will responds with the IP Address and Port# of the next server it knows about.

REFER	RQ#	IP Address	Port#

REFER Message

X will then try with the new server, and so on as in the case of registration.

2.4. Communication between Users

In order to communicate with Y, X has to find how to reach Y. When this is done (see previous sections), X sends a Chat message to Y and both can exchange Chat messages over UDP.

CHAT	Name	IP Address	Port#	Text

CHAT Message

Le Chat Message carries the Name, IP Address and Port# of the sender of the message as well as a Text with a maximum size of 140 bytes. If the user is sending more than 140 bytes at a time, then this message has to be segmented at the sender side and reassembled at the receiver side.

The chat goes on until when of them (X or Y) says BYE to the other one and receives BYE from the other one, or receives BYE and sends back the BYE message.

BYE	Name	IP Address	Port#

BYE Message

As for the Chat message the BYE message carries the Name, IP Address and Port# of the sender.

3. Requirements

Project should be done in groups of 2 students. You should send, by September 27, your group list including student names, ID numbers and **ECE** email addresses to khendek@ece.concordia.ca.

Design and Implement the Users and Servers that follow the set of protocols aforementioned. The coding of the protocol messages is part of your design, i.e. you have to come out with the appropriate coding of the messages. You can decide to use simple text message, etc.

The information stored in the server should be persistent, i.e. if a server crashes and is restored it will recover the information about the registered users.
Reporting: Servers and Users should be reporting their communications with other entities to the user using a log file or printing directly into the screen. In other words, during the demonstration, I would like to see the messages sent and received, progress and failures.

Assumptions/Error/Exception Handling
You should be aware that the description as it is does not state everything. For instance what happens if a User receives a response with a RQ# that does not correspond to any of its (pending) requests?

Is a User allowed to change its published information while it communicating with other Users?

State and document clearly any assumption you make beyond the assumptions made by the instructors.

Extra: Notice that in this project we do not require authentication of users. Extra marks will be given to students who add an authentication scheme to the PCS system.
Also, a GUI is not required as long as we can run the clients and the servers and see what is going on with the messages. Extra marks will be given if you decide to build a GUI.

You should hand in a report, by Week 13, where you document clearly your assumptions, design decisions, code and experiments. You should also state clearly the contributions of every member of the group. Every student has to contribute **technically** (designing and implementing the protocol) to the project.

A demo will be held during Week 13 of this fall term. During the demo the members of the group should all be ready to answer questions from the instructor. We will be running at least 10 Users and 2 Servers on different machines and tests all the messages.

During the demo we may also go through the code itself and the report.

The project will be discussed further in class.