First version of RFC:

Happy chatting room

Description:

This is a virtual chatting room serving for two or three people communicating with each other. The chatting pairs must run this program on their computer simultaneously. Once they establish connections with each other, they can share text messages.

User Interface:

- **Part one (Establish connections):**

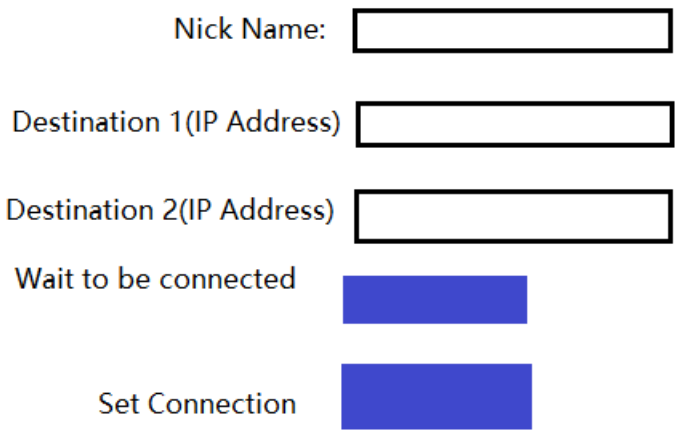
There should be the following functions:

1: Input a nick name representing the host when chatting with others.

2: Input other users' IP addresses to be connected to.

3: There should be at least two buttons. One button is for initializing connections with other users. Another is for waiting the connection request from other users. These two buttons corresponding to different state of your host.

It looks like the following figures:



- **Part two (virtual chatting room):**

There should be the following functions:

1: Display the message that the users share with each other.

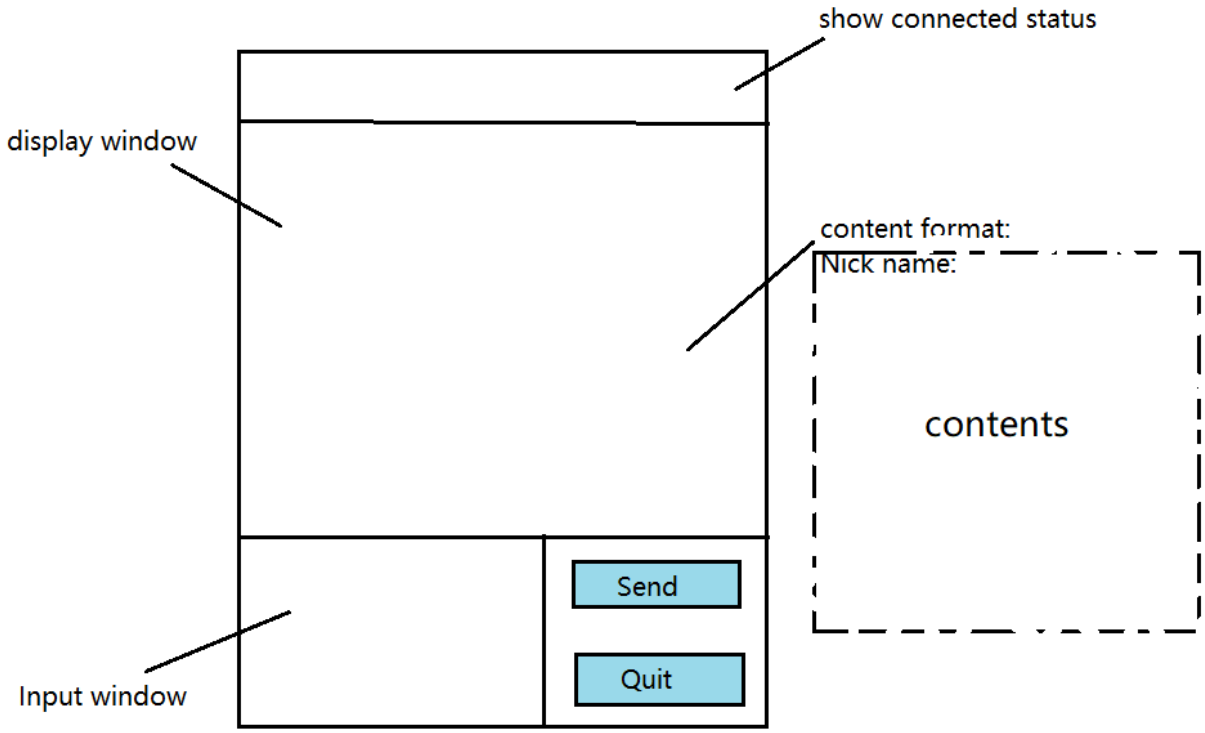
2: Display the connection status. (who are active in this chatting room now.)

3: Input the text message and send it to other users.

4: The format of the message should be: Sender's nick name + contents

5: At least two buttons, one is for sending the message that user inputs, another is for quitting the chatting room

It looks like the following figure:



How does this application work?

Three users must run this application simultaneously. They would enter the first part of the UI. Two users need to offer their nick name and click on "Wait to be connected" button. Another user must be the initializer who need to offer the nick name, IP addresses of other users, and click on Connect button. Once they are connected, they would all enter into the second part of the UI. Then they can start sharing message!

How to set up connections among these three users?

The initializer would use TCP protocol to set up connections with other two users respectively.

How to transport packets among them?

- **For the initializer:**

All the packets that the initializer received or packaged are putting into a queue. And the initializer periodically pulls a packet from the queue and sends that packet to other two users. Then the initializer extracts data from that packet and displays the data on the UI.

- **For other two users:**

(1) Once a user sends a message out, the message would be packaged into one or multiple packets being sent to the initializer.

(2) Once a user receives a packet from the initializer, it will extract the data from that packet and display it on the UI.

This strategy is designed to guarantee the messages displayed in each user's UI are same to each other.

Error recovery:

If the initializer can only get connected to one user, they can still enter into the virtual room. And the initializer will continue trying to connect to another user.

If the initializer cannot get connected to any users, there should be an alert: connection failed!

Graded comments and peer review:

Graded comments for First version:

Prof. Heller will be updating grade. (3) This is so incomplete it is hard to grade this. I see two users must be on at the same time, how does one know? what is the syntax of the messages, what port do you plan to use, how does this end are just a few of the items missing

For form and structure, the submission does not follow the form and structure for a RFC.There is no table of contents, no information about author and other components required in a RFC document. Might want to meet me to discuss how you can improve. (3)

Graded comments for revised version：

OK to implement

The WID grader will add points for form and format

Form and format - 3/5
RFC largely conforms to standard RFC format. However, following can be improved. Wrong usage of fonts. Images need to be titled. References need to be numbered.

Student comments:

Wu Yang:

Top of Form

Hi Zhao,

I think you clearly show the design ideas in the file, I can understand the construction process of the chat room very intuitively.
Your understanding of the chat room construction makes me very inspired.

Kebo Duan

A good and clear proposal, I have a few questions,

1) Can the chat room accommodate more than three people? Why?

2) What are the port numbers of destination 1 and destination 2? Should they be the same? How does the current user know these port numbers?

3) The initializer looks like a server that forwards all the messages. However, instead of letting all the other members connect to initializer, you let initializer start all the connections. Why?

BTW, we need to use the RFC template.

Final version of the RFC

CSCI 6431: Computer Networks Group Qi Zhao

Request for Comments The George Washington University

Category: Experimental October 22 2018

Three People chatting room Protocol(TPCRP)

Abstract

This document specifies the application layer and transport layer of a web application, which serves as a virtual chatting room in which up to three people can chat with each other.

This document defines a few necessary functions and the format of the User Interface(UI). As for the transport layer, this document specifies how the application set connection with other hosts and how the packets are transferred among those hosts.

Status of this Memo

This document is the first version of the TPCRP, which requests discussion and suggestion for improvements. It has not been implemented yet. Distribution of this memo is unlimited.

Table of Contents

1. Introduction 10
2. Terminology 10
3. Architecture overview 10
4. User Interface: 10
4.1 First Window (Establish connections) 10
4.2 Second window (virtual chatting room) 11
5. Back-end 12
Event1: mouse click 12
Event2: Button function 12
Event3: packet arrive 13
Event4: Broadcast 13
6. Transport layer: 13
6.1 Connections setting up: 13
6.2 Packets transferring 13
7. Example 13
8. Exception handle 13
8.1 The user lose connection. 13
8.2 One User has very high latency 14
9. Informative References 14
10. Author's Address 14

1. Introduction

Most chat applications are centralized. For the developers, that means most of cost are spent on storing, processing and forwarding data. While the 'Three people chatting room" specified in this document is decentralized. And it can serve two or three people communicating with each other without relying on any servers. Because it is very cheap to operate this application, "Three people chatting room" could be a very promising application. As for the limitations, the chatting pairs must run this program on their computer simultaneously. But once they establish connections with each other, they can share text messages almost freely.

2. Terminology

I. Connection status
This term is used to indicate which destination host is connected to the current host.
II. Initializer
This term represents the host who initialize the connection with other hosts.
III. Virtual chatting room:
This term represents this application itself.
IV. Common Users
Common Users represent the hosts who are not initializer.

3. Architecture overview

- User Interface:
  A few fields for inputting or outputting something
  A few buttons for execute some necessary functions.
- Back end:
  Handle the sending and receiving of packets.
  Handle some exceptions caused by unexcepted events.
- Transport layer:
  TCP protocol
  The hosts set up connections with each other
  The transferring of the packets among the hosts.

4. User Interface:

4.1 First Window (Establish connections)

There should be the following functions:

a. Input a nick name representing the host when chatting with others (restricted to 50 characters).
b. Input other users' IP addresses to be connected to (It must employ the format of IP address like 192.168.1.1).
c. There should be at least two buttons. One button is for initializing connections with other users. Another is for waiting the connection request from other users. These two buttons corresponding to different state of your host.

It looks like the Figure 1:

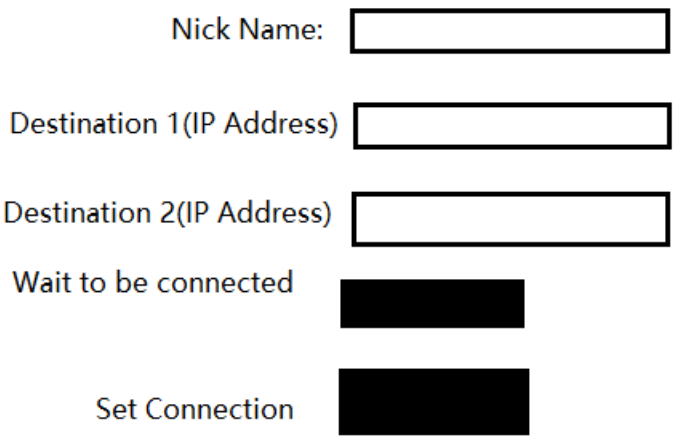


Figure 1

4.2 Second window (virtual chatting room)

There should be the following functions:

a. Display the message that the users share with each other (ASCII Base).
b. Display the connection status. (who are active in this chatting room now.)
c. Input the text message and send it to other users (restricted to 500 characters).
d. The format of the message should be: Sender's nick name + contents
e. At least two buttons, one is for sending the message that user inputs, another is for quitting the chatting room

It looks like the figure 2:

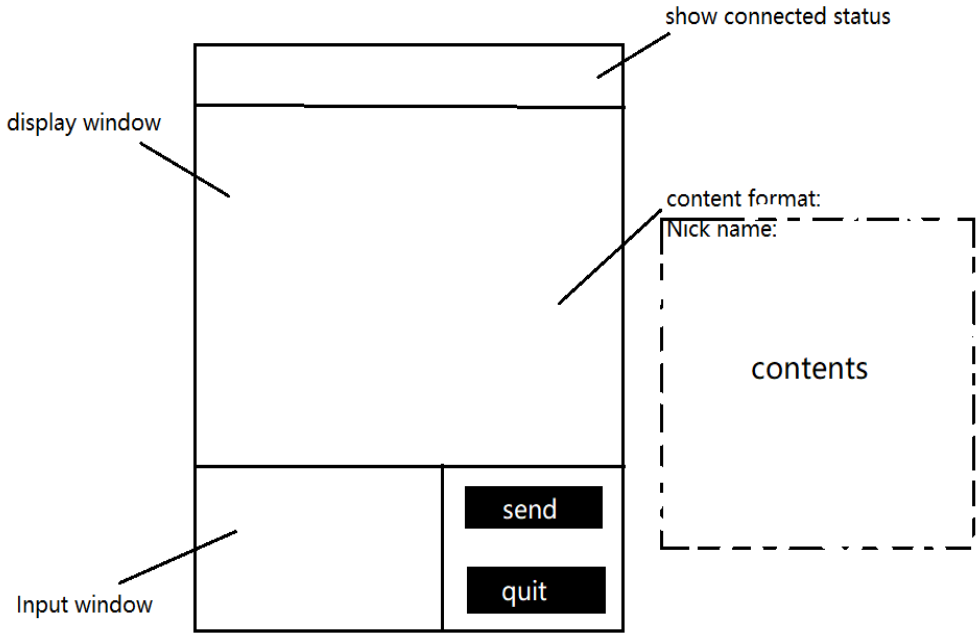


Figure 2

5. Back-end

There are a few events that the developer must handle in some specified ways.

Event1: mouse click

The user could use mouse click to choose a field to be filled with, and use keyboard to input characters into that field.

The user could also use mouse click to click the button and run the corresponding function.

Event2: Button function

Set connection (First Window): Once the User click this button, the host enter into the state- initializing the connections with other users.

Wait to be connected (First Window): Once the User click this button, the host enter into the state- Open its port 20000, and keep in the listening mode.

Send(Virtual chatting room): Once the User click this button, the message inputted in the input window will be packaged and sent to the initializer.

Quit(Virtual chatting room): If the initializer click this button, the chatting room will be closed. All the connections will be disrupted. If common users click this button, the connection between the initializer and that user will be disrupted. But the connection between the initializer and the other common users will still be remained.

Event3: packet arrive

Once a packet arrives, for the initializer, it will store the packet temporarily in a specific buffer(Queue). But for the common users, it will extract the data and send the data to display window.

Event4: Broadcast

The initializer periodically checks its buffer and extract a packet, which will be displayed in the initializer's window and sent to other users.

6. Transport layer:

6.1 Connections setting up:

The initializer would use TCP protocol to set up connections with other two users respectively. So, there are totally two connections, one is for initializer and a host, another is for initializer and the other host.

6.2 Packets transferring

In order to guarantee the messages displayed in each user's UI are same to each other.

- For the initializer:
  All the packets that the initializer received or packaged are putting into a queue. And the initializer periodically pulls a packet from the queue and sends that packet to other two users. Then the initializer extracts data from that packet and displays the data on the UI.
- For other two users:

A. Once a user sends a message out, the message would be packaged into one or multiple packets being sent to the initializer.
B. Once a user receives a packet from the initializer, it will extract the data from that packet and display it on the UI.

7. Example

Here is an example to illustrate how this application work.

We assume User A, B, C want to create a virtual chatting room and chat with each other. Here are the procedures:

1) A, B, C must make sure that they run this application simultaneously.
2) They need to enter the first window of the UI. We assume A is the initializer. Thus B,C users need to fill their nick name fields and click on "Wait to be connected" button. User A needs to fill the nick name and IP addresses of other users to the corresponding fields, and click on Connect button.
3) Once they are connected, they would all enter into the second window of the UI. Then A, B, or C could input some message in the input field and click on send button. The message would be send to other users and displayed in the output field of the three chatting windows.

8. Exception handle

8.1 The user lose connection.

For the initializer, all the users would lose connection.

For other user, the connection between the initializer and it will be lost. But other connections will be remained.

8.2 One User has very high latency.

Assume A,B,C are connected to each other and A is the initializer. If B has very high latency while C didn't, then we change the initializer from A to C and try to connect them again.

9. Informative References

1. [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, .

2. [RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, .

10. Author's Address

Qi Zhao

The George Washington University

Department of Computer Science

Washington, DC

Mobile number: 2027046018

Email ID: zhaoqi@gwu.edu