Problem 2: Distributed Chat Service

Chris Baumler

Description:

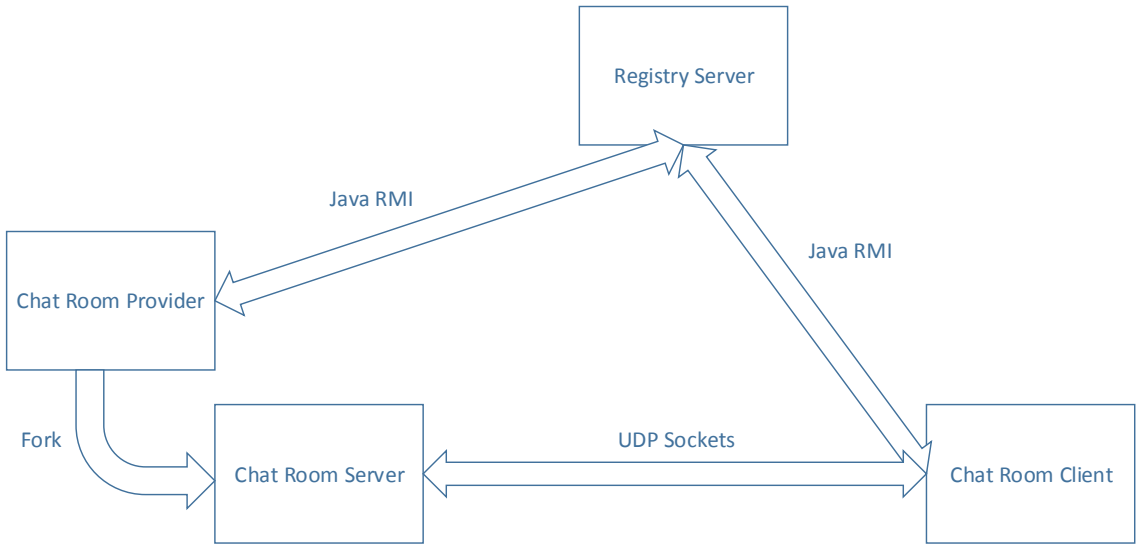
The distributed chat service consists of four component types: a registry server, a chat room provider, a chat room server, and a chat client. These components interact using Java RMI and UDP sockets to allow users to join one or more chat rooms and send and receive messages.

Registry Server: A single registry server exists as a database that maintains the location, name, and other information about the chat clients and chat room servers that have registered with it.

Chat Room Provider: The chat room provider is responsible for registering chat room servers with the registry server and creating new threads to execute the chat room servers.

Chat Room Server: Chat room servers listen for clients to join or leave and keep track of clients in a local database. Chat messages are forwarded to all joined clients.

Chat Room Client: The chat clients register with the registry server and receive inputs from the user to join chat rooms and send messages to chat rooms. They also display messages received from chat rooms.



Requirements:

The registry server **shall** provide a Java Remote Method Invocation (RMI) interface for a client to register an entity.

The registry server **shall** provide a Java RMI interface for a client to unregister an entity.

The registry server **shall** provide a Java RMI interface for a client to query the registry for information about the registered entities.

The registry server **shall** provide a Java RMI interface for a client to retrieve a list of the registered chat room entities.

The registry server **shall** maintain a database of all registered entities including each entity's address and port and whether the entity is a chat room.

The chat room client **shall** register with the registry server using the provided Java RMI interface.

The chat room client **shall** query the registry server for the currently available chat rooms using the provided Java RMI interface and display them for the user when commanded by the user.

The chat room client **shall** send a Universal Datagram Protocol (UDP) message to a chat room server requesting to join the chat room when commanded by the user.

The chat room client **shall** send a UDP message to a chat room server requesting to leave a chat room when commanded by the user.

The chat room client **shall** designate a chat room to receive user specified chat messages when commanded by the user.

The chat room client **shall** send user specified chat messages to the designated chat room servers using UDP.

The chat room client **shall** receive and display chat messages from chat rooms that have been joined.

The chat room client **shall** allow both display of incoming messages and input of outgoing messages in a single console.

The chat room client **shall** display errors to the console.

The chat room client **shall** unregister with the registry server upon exit.

The chat room provider **shall** register one or more chat rooms with the registry server based on its configuration.

After registering a chat room, the chat room provider **shall** create a new thread to run the chat room server.

The chat room provider **shall** unregister a chat room with the registry server when the chat room server thread finishes.

The chat room server **shall** listen on a well-known port for UDP messages from chat clients.

The chat room server **shall** maintain a database of chat clients that have joined its chat room.

When a join request is received from a client, the chat room server **shall** add the client to its database and advertise its well-known port to the client.

When a leave request is received from a client, the chat room server **shall** remove the client from its database.

When a chat message is received, the chat room server **shall** forward the message to all of the chat clients listed in the database.

Instructions for Use:

Launch the Registry Server:
```
java –jar RegistryServer.jar
```

Launch the chat room provider:
(Specify as many chat room name/port pairs as desired)
```
java –jar ChatRoomProvider.jar ExampleRoom1 5678 ExampleRoom2 8765
```

Launch the chat room client:
(Specify the desired port number)
```
java –jar ChatRoomClient.jar 9999
```

Using the chat room client:

Enter a username on startup. This can be whatever the user wants.

Commands (key words are not case sensitive):

SHOW
Display a list of the available chat rooms that can be joined.

JOIN x
(x is the full name of the chat room as indicated by the SHOW command)
Join a chat room. This will cause chats from this chat room to be displayed by the client. However, this will not cause the user's chats to be published to the chat room.

LEAVE x
(x is the full name of the chat room as indicated by the SHOW command)
Leave a chat room. This will cause chats from this chat room to no longer be displayed by the client.

ACTIVE x
(x is the full name of the chat room as indicated by the SHOW command)
Set a chat room as the active chat room. This means that the user's chats will be published to the chat room.

QUIT
Exit the chat client.

Measurements:

Procedure 1:
Using a test bed project (TestBed.java), the latency of a registration/de-registration cycle was measured. First, a specified number of chat rooms were registered. Then the start time was recorded and an additional chat room was registered/de-registered for a specified number of cycles. After this, the end time was recorded, and the elapsed time was computed. The elapsed time was divided by the number of cycles to determine the average registration/de-registration cycle latency.

Results:
After performing the test procedure multiple times, the registration/de-registration latency varied between 0.8 ms and 1 ms. Increasing the number of chat rooms in the system between 1 and 1,000,00 did not appear to affect the latency.

Procedure 2:
Using a test bed project (TestBed.java), the latency of an inquiry was measured. First, a specified number of chat rooms were registered. Then the start time was recorded and information was requested for a single chat room a specified number of times. After this, the end time was recorded, and the elapsed time was computed. The elapsed time was divided by the number of cycles to determine the average inquiry latency.

Results:
After performing the test procedure multiple times, the inquiry latency varied from as low as 0.35 ms to as high as 2.847 ms. Increasing the number of chat rooms in the system between 1 and 1,000,000 caused the latency to increase.