Design Document

A Distributed Chat Room Service with Java RMI

Anindita Das
dasanuiit@gmail.com

About This Document

This document describes the design of a chat room service. The service is implemented using Java RMI. The service includes (1) creation and registration of chat rooms and clients, (2) joining and leaving of chat rooms by chat clients, and (3) chat communication among clients.

System Components

The chat room system involves interaction of four distinct components: (1) Registry, (2) Provider, (3) Server, and (4) Client. A key design issue in this project is to clarify the role and functionality of these components. We describe this below. We then discuss how the Java RMI service is used to implement the agents and their communication.

- **Registry:** The registry performs registration and deregistration of chat rooms and chat clients. There is a single registry in the system.
- **Provider:** A chat room provider serves as an interface for initiating, registering, and deregistering chat rooms.
- **Server:** A chat room server is associated with each individual chat room, and is responsible for providing the chat services to the chat clients.
- **Client:** Clients can be registered and deregistered with the registry like chat room servers. They also interact with the different chat room servers to communicate (chat) with each other.

System Design

The system is designed as a messaging service with Java RMI. We now discuss how the functionality and communication of the different components are implemented through RMI.

Registry. The registry is implemented as an RMI server that provides (remote) methods for registering and deregistering chat room servers and clients (as well as methods for getting information about registered servers and clients).

Provider. The provider is designed as a user interface to control registration and deregistration of individual chat room servers. In fact, the provider can be viewed as a simple "server factory", either (1) registering a chat room, or (2) deregistering a registered chat room, at the user's request. A chat room is available to clients only after registration. The registering and deregistering activities simply invoke the registry's remote methods on the chat servers. In order to handle asynchrony induced by communications from multiple chat servers, the provider is implemented as a multithreaded process. In particular, the provider creates a dedicated thread for each chat room (together with a dedicated server for the room) during creation; the main thread of the provider is then reserved to handle user commands.

Server. Each chat room has a dedicated server. Thus the server needs to only manage the interaction of the clients participating in a chat in the room (as well as the provider). The server can be viewed as an RMI server for the chat clients. It implements remote methods for the clients, including "join", "leave", "talk", etc., which are invoked by the chat client.

Client. Clients interact with both the registry and the server as RMI clients. They invoke the remote methods in the registry to register or deregister with the system. Once registered, they invoke the remote methods for a chat room server to participate in the chat room.

We now briefly discuss the design for the "talk" functionality. A chat room server provides a remote talk method that the client invokes to send a message to a chat room. It is then the server's responsibility to ensure that the message is visible to all the clients that have joined the room. This is achieved as follows. The server maintains a list of all active clients in the room (*i.e.*, have joined but not left). Furthermore, a client, on joining a room, creates a callback object shared with the server. When the server subsequently receives a chat message, it iterates through the list of active clients, using the callback object to forward the received message.

An interesting aspect of the design is to handle the situation in which a chat room containing active clients is deregistered by the room provider. In this scenario, it is imperative for the participating clients to be informed that the room is no longer available. To achieve this, the client process looks up the registry for the name of the chat room every time before transmitting a message; when the room is deregistered, this lookup fails informing the client that the room no longer exists.

The implementation is a fairly direct rendition of the above remote methods using Java RMI. There is no restriction on the number of chat room providers, the number of chat rooms for any provider, or the number of clients, other than limitations imposed by system resource. One key assumption is that the registry server must successfully start before any chat room provider or client process is initiated.