A Technical Report of Distributed Whiteboard

COMP90015: Distributed Systems

Lujie Ma

1391364

The University of Melbourne

May 18, 2023

Introduction

Due to the increasingly sophisticated development of Internet technology nowadays, a lot of collaborative work via the Internet is becoming more and more frequent. Our whiteboard application allows multiple users to draw on a shared canvas in real time.

Communication Protocols

In my distributed shared whiteboard, Remote Method Invocation (RMI) is adopted. The Remote Method Invocation (RMI) is a Java API that performs the object-oriented equivalent of remote procedure calls (RPC), with support for the direct transfer of serialized Java classes.

For the server, it exports a remote object that can be invoked remotely. It creates a registry that binds the name “WhiteboardServer” to the remote object. This allows the client can look up and invoke methods on this remote object.

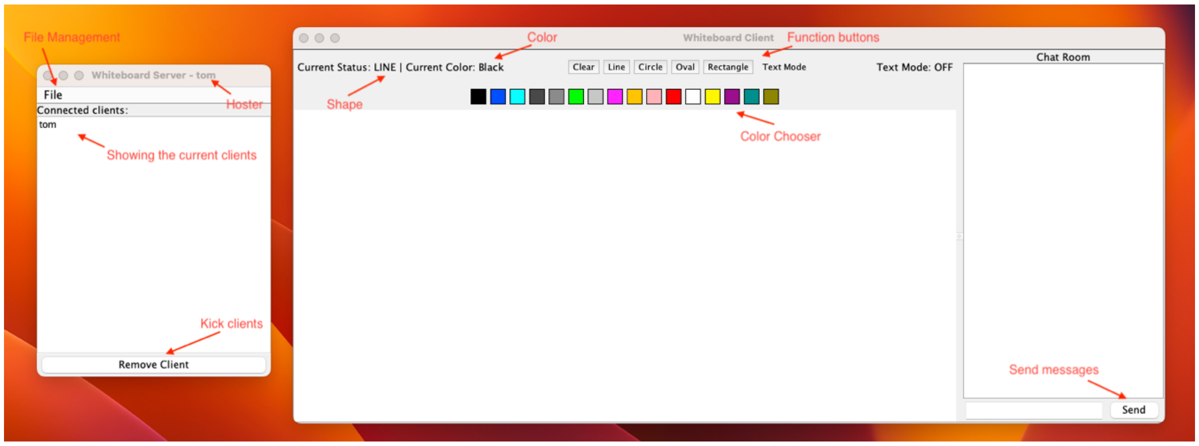
For the client, it looks up the remote objects of the whiteboard clients and invokes methods on these objects to update the clients about the state of the whiteboard. E.g., when a new drawing is added, it invokes the draw and renderDrawings methods on all the client objects to update their views.

In conclusion, the use of RMI allows for real-time, efficient, and reliable communication between multiple JVMs.

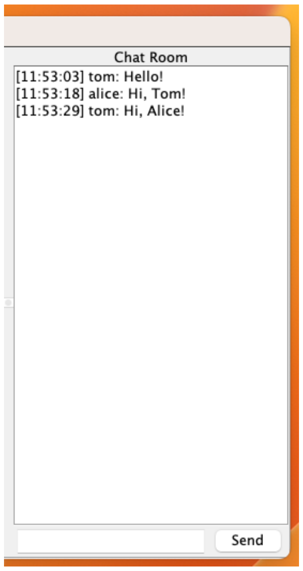
GUI

For the client side:

- Our client GUI is shown in the left part of the picture. We can easily see that there shows the status (shape), current colour and text mode. If the text mode is OFF, users can draw in the canvas, however, if the text mode is ON, users can only type in the canvas, and the drawing tools will not work. Users are free to choose their favourite colours and shapes to paint. Our Shared Whiteboard offers over 16 colours and four different shapes for users to choose from: Line, Circle, Oval, and Rectangle.



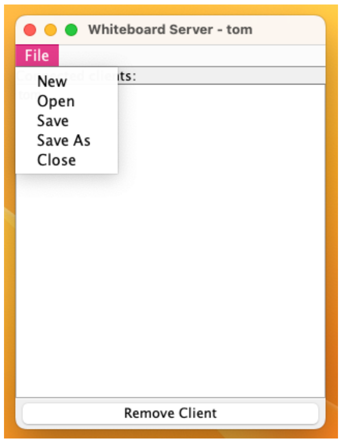
- On the right side of our client GUI is a chat room. A chat room is essential to allow everyone to work together, share their own ideas, and exchange ideas with each other. In the chat room, we give every message a timestamp, and it will display on the left of the users' names. The structure is "[timestamp] + username: + message" .



[11:53:03] tom: Hello!
[11:53:18] alice: Hi, Tom!
[11:53:29] tom: Hi, Alice!

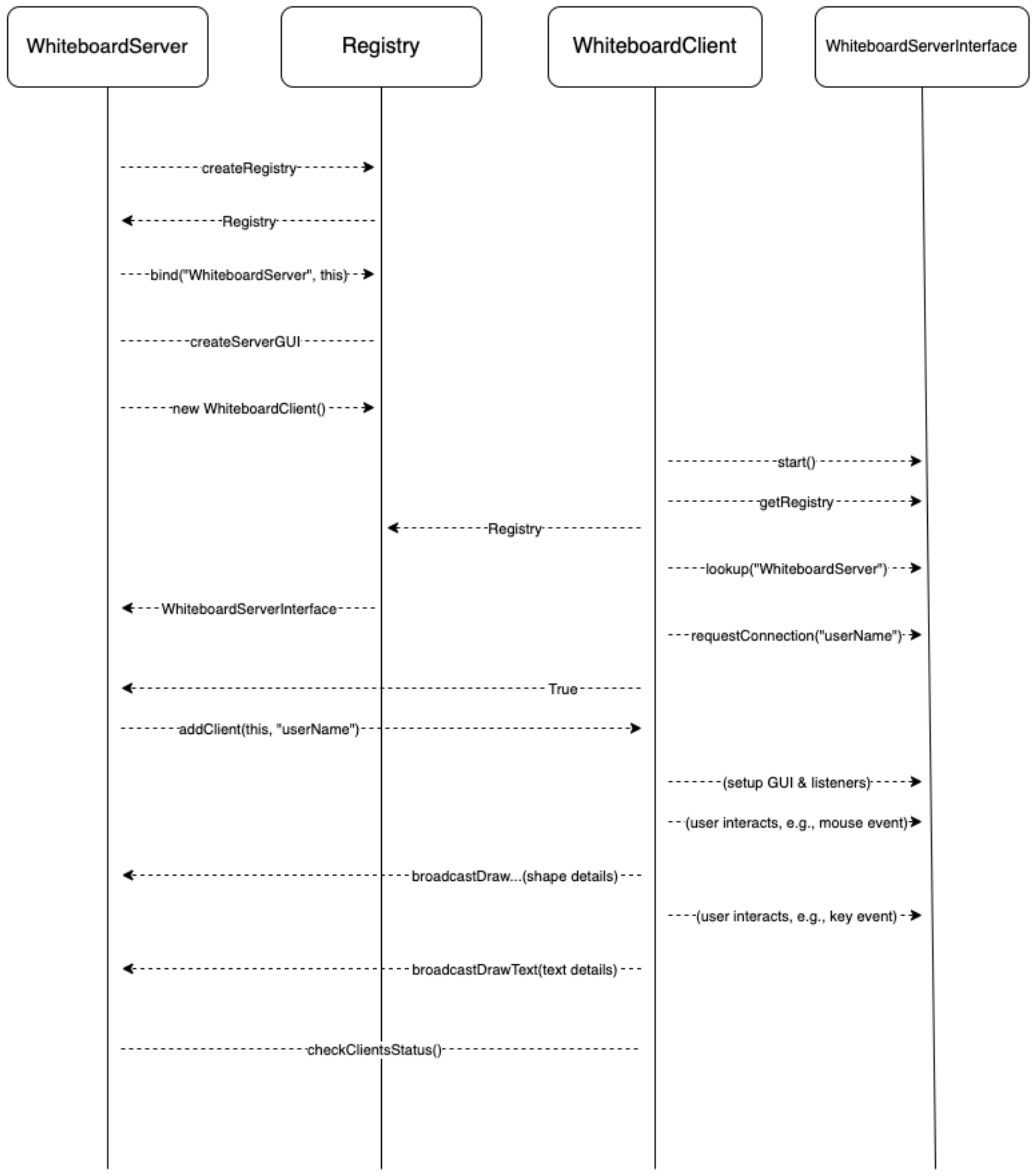
For the server side:

- Each time the server side is started, a console and a canvas are displayed together. The feature of the canvas is the same as the client side. The console integrates some specific features which only the manager can use it. The console will show the current clients’ names that are connected to the server now. The manager is also able to kick the users.



- The file management system is also provided. It has five functions: New, Open, Saves, Save As and Close. The following are explanations of these five functions:
    - New: Clear the current state of the application, allowing the user to start from a clean slate.
    - Open: Open the file dialogue box which allows the user to select a previously saved file. Once the user selects a file, the program should load the data from that file and use it to restore the application state.
    - Save: If the current project has a file associated with it, overwrite that file with the current state of the application. If there is no associated file, this will lead to acting the same with “Save As”.
    - Save As: Opens a file dialogue, lets the user specify a location and filename, and saves the current state of the application to that file.
    - Close: Close the currently open file.

Design Diagram


WhiteboardServer
Registry
WhiteboardClient
WhiteboardServerInterface
createRegistry
Registry
bind("WhiteboardServer", this)
createServerGUI
new WhiteboardClient()
start()
getRegistry
Registry
lookup("WhiteboardServer")
WhiteboardServerInterface
requestConnection("userName")
True
addClient(this, "userName")
(setup GUI & listeners)
(user interacts, e.g., mouse event)
broadcastDraw...(shape details)
(user interacts, e.g., key event)
broadcastDrawText(text details)
checkClientsStatus()

Implementation details

- How to check if the client is connected to the server ?

  On the server side, we create a function to receive responses from the clients. The method creates a timer, fired every 5 milliseconds. It will go through all the clients in the 'clients' list and check the status. If the ping is successful, it means the client is alive. If the ping failed, it means the client is no longer alive and will be removed from the client's list.

  This method serves as a continuous background check ensuring all clients in the system are active and cleaning up any inactive or failed clients.

- How to keep the application is thread-safe?

  To keep our whiteboard is thread-safe, 'CopyOnWriteArrayList' is adopted, which is a thread-safe variant of 'ArrayList'. It makes some operation of adding clients to the array list does not need to be synchronized because 'CopyOnWriteArrayList' ensure that all mutative operations are atomic.

New innovations

In addition to fulfilling the basic requirements, we have implemented some small innovations to enhance the user experience.

- Shape Preview

  Our whiteboard application allows users to visualize their drawing when they are dragging the mouse. It is a user-friendly design addition to our application.
- Clear Canvas

  When the canvas has many drawings on it, we offer a clear button, which can clear all drawings from the canvas.