A Distributed Notification Framework using Java RMI

Notification framework

The source and sink java classes, *NotificationSource.java* and *NotificationSink.java*, both are implementations of their respective interfaces, *NotificationSourceInterface.java* and *NotificationSinkInterface.java*. The sink acts as an interface between the client and the source whilst the source acts as an interface between the sink and the server. The sinks subscribe/register with the source through a special method which adds each sink to an array list maintained by the source. The source sends serialised notification objects to the sinks in this array list whenever a particular event happens. This notification object contains useful data for the sinks which receive it.

Application

The application I have built is a simple chatroom program using a command line interface. Whenever the client is run and successfully connects with the server through the sink and source, the user is prompted to enter a username and can then send messages to all other subscribers of that particular source.

The client and sink as well as the server and source communicate bidirectionally. The sink communicates directly with the source by calling its methods whereas the source communicates with the sink through one update method that passes a notification object. Both source and sink inherit from their interfaces. The following is a class diagram demonstrating these links between the classes:

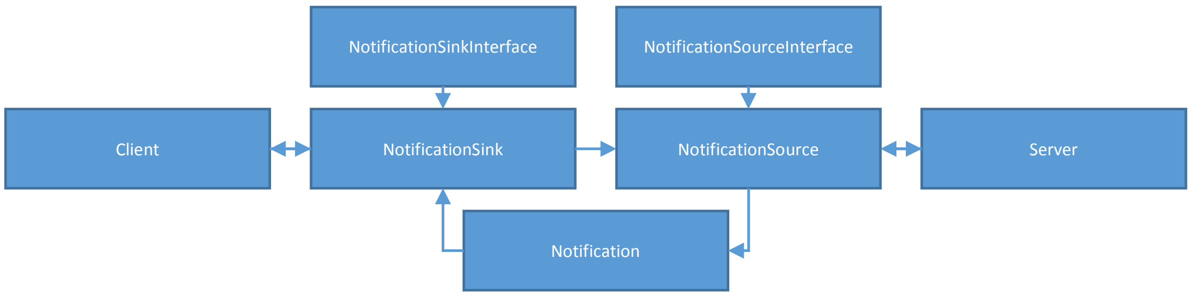
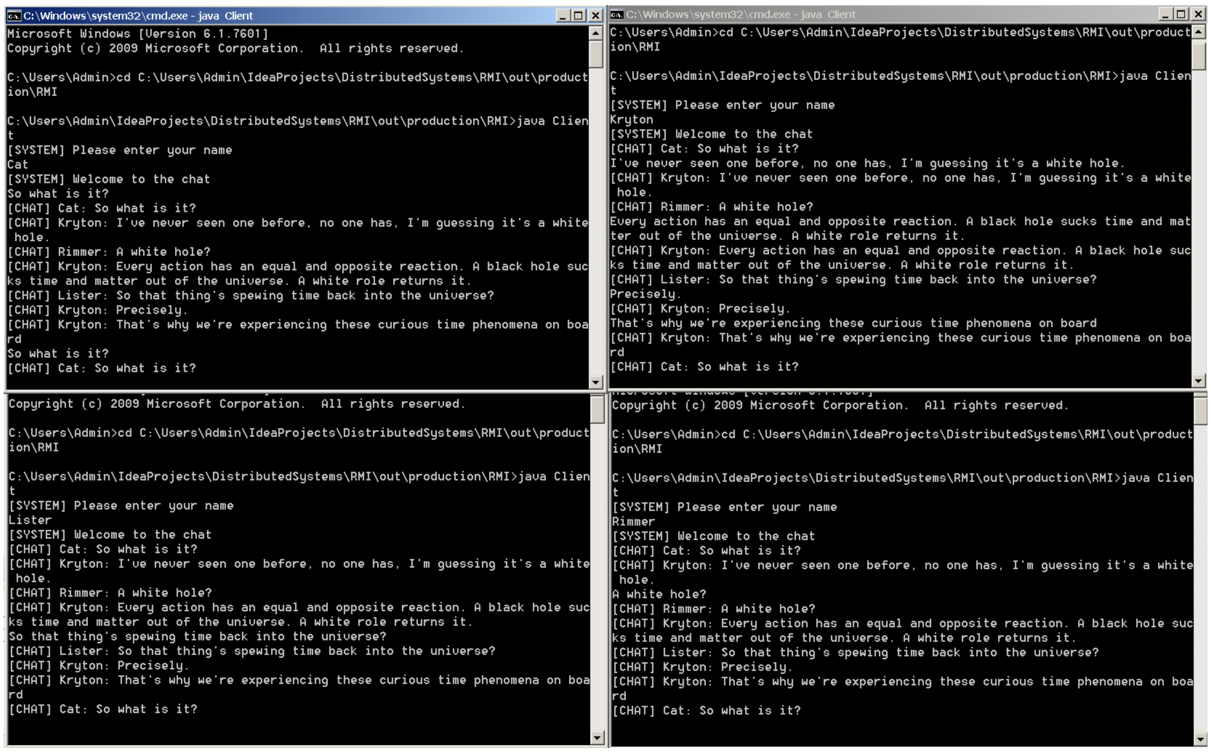


FIG. 1: A CLASS DIAGRAM

The application was thoroughly tested through imaginary conversations held between a large number of participants. This was done by running the server class, thus creating a server and source with which to hold this extraordinary chat, and then running several clients in different command windows. Though these results and analysis of the structure of the application, it was obviously apparent that the application worked successfully and messages were successfully passed through RMI communication. Below is a screenshot of this process, showing the perspectives of the four users:



```
C:\Windows\system32\cmd.exe - java Client
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Admin>cd C:\Users\Admin\IdeaProjects\DistributedSystems\RMI\out\product
ion\RMI

C:\Users\Admin\IdeaProjects\DistributedSystems\RMI\out\production\RMI>java Clien
t
[SYSTEM] Please enter your name
Cat
[SYSTEM] Welcome to the chat
So what is it?
[CHAT] Cat: So what is it?
[CHAT] Kryton: I've never seen one before, no one has, I'm guessing it's a white
 hole.
[CHAT] Rimmer: A white hole?
[CHAT] Kryton: Every action has an equal and opposite reaction. A black hole suc
ks time and matter out of the universe. A white role returns it.
[CHAT] Lister: So that thing's spewing time back into the universe?
[CHAT] Kryton: Precisely.
[CHAT] Kryton: That's why we're experiencing these curious time phenomena on boa
rd
So what is it?
[CHAT] Cat: So what is it?
```

```
C:\Windows\system32\cmd.exe - java Client
C:\Users\Admin>cd C:\Users\Admin\IdeaProjects\DistributedSystems\RMI\out\product
ion\RMI

C:\Users\Admin\IdeaProjects\DistributedSystems\RMI\out\production\RMI>java Clien
t
[SYSTEM] Please enter your name
Kryton
[SYSTEM] Welcome to the chat
[CHAT] Cat: So what is it?
I've never seen one before, no one has, I'm guessing it's a white hole.
[CHAT] Kryton: I've never seen one before, no one has, I'm guessing it's a white
 hole.
[CHAT] Rimmer: A white hole?
Every action has an equal and opposite reaction. A black hole sucks time and mat
ter out of the universe. A white role returns it.
[CHAT] Kryton: Every action has an equal and opposite reaction. A black hole suc
ks time and matter out of the universe. A white role returns it.
[CHAT] Lister: So that thing's spewing time back into the universe?
Precisely.
[CHAT] Kryton: Precisely.
That's why we're experiencing these curious time phenomena on board
[CHAT] Kryton: That's why we're experiencing these curious time phenomena on boa
rd
[CHAT] Cat: So what is it?
```

```
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Admin>cd C:\Users\Admin\IdeaProjects\DistributedSystems\RMI\out\product
ion\RMI

C:\Users\Admin\IdeaProjects\DistributedSystems\RMI\out\production\RMI>java Clien
t
[SYSTEM] Please enter your name
Lister
[SYSTEM] Welcome to the chat
[CHAT] Cat: So what is it?
[CHAT] Kryton: I've never seen one before, no one has, I'm guessing it's a white
 hole.
[CHAT] Rimmer: A white hole?
[CHAT] Kryton: Every action has an equal and opposite reaction. A black hole suc
ks time and matter out of the universe. A white role returns it.
So that thing's spewing time back into the universe?
[CHAT] Lister: So that thing's spewing time back into the universe?
[CHAT] Kryton: Precisely.
[CHAT] Kryton: That's why we're experiencing these curious time phenomena on boa
rd
[CHAT] Cat: So what is it?
```

```
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Admin>cd C:\Users\Admin\IdeaProjects\DistributedSystems\RMI\out\product
ion\RMI

C:\Users\Admin\IdeaProjects\DistributedSystems\RMI\out\production\RMI>java Clien
t
[SYSTEM] Please enter your name
Rimmer
[SYSTEM] Welcome to the chat
[CHAT] Cat: So what is it?
[CHAT] Kryton: I've never seen one before, no one has, I'm guessing it's a white
 hole.
A white hole?
[CHAT] Rimmer: A white hole?
[CHAT] Kryton: Every action has an equal and opposite reaction. A black hole suc
ks time and matter out of the universe. A white role returns it.
[CHAT] Lister: So that thing's spewing time back into the universe?
[CHAT] Kryton: Precisely.
[CHAT] Kryton: That's why we're experiencing these curious time phenomena on boa
rd
[CHAT] Cat: So what is it?
```

FIG. 2 A TYPICAL CONVERSATION IN THE CHAT APPLICATION

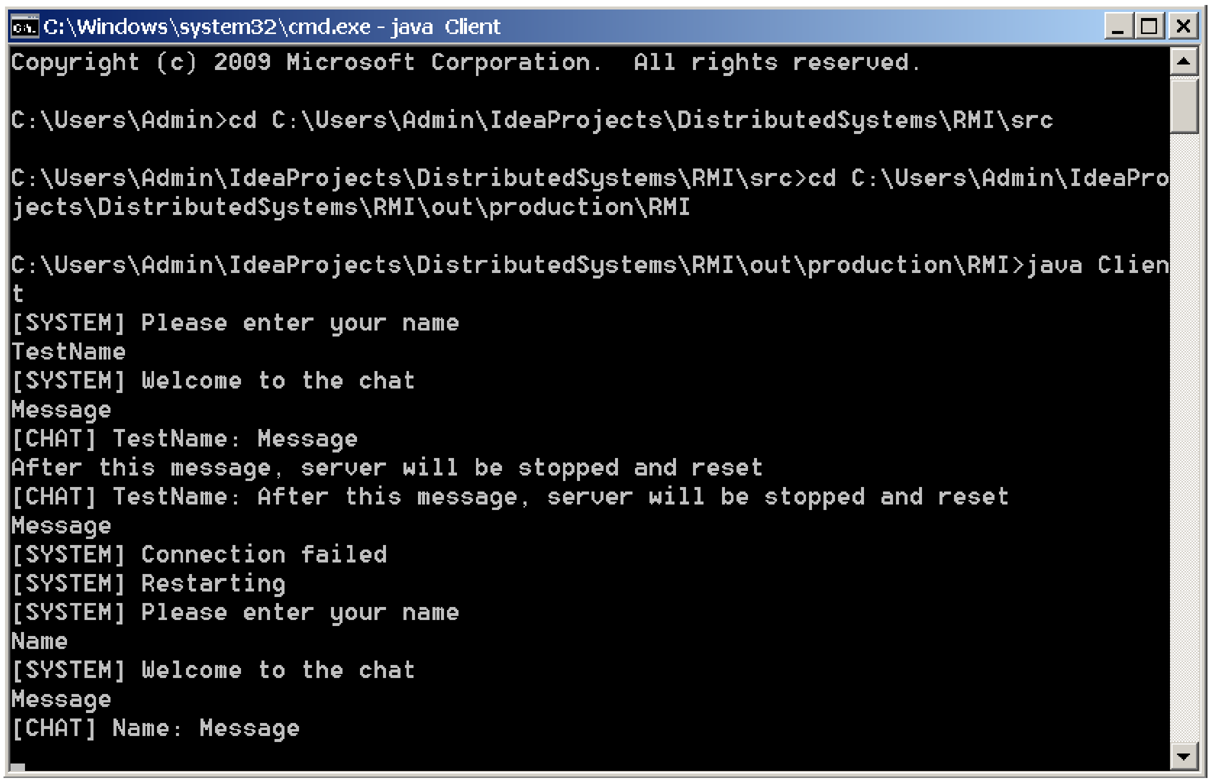
Multiple Sources & Sinks

The application can handle multiple sinks at once connecting to a single source. Without this feature, it would be a very solipsistic chatroom. This can be demonstrated by running the server and then running the client multiple times, using many different usernames to differentiate the many instances of the client connecting to the same source. Evidence of this was shown in the previous section.

With regards to a single sink connecting to multiple sources, the basic backbone of this is implemented with the server receiving an additional constructor method to making an additional source. However, it was difficult to envision a way to logically and intuitively add another chatroom source to the command line interface in a way that makes sense. The fundamentals are there through the createNewSink methods and minimum modification would be required.

Lost Connections

The program has rudimentary procedures for restoring connection after a connection failure. In the main client loop, if a RemoteException is encountered when the user tries to send a message, the program will reset and try to reconnect to the server it just failed connection with. In most instances, it will succeed. Since it connects to the same registry location, this is the same chatroom and the same source as before. This is demonstrated in the screenshots below, where, after the second message is sent, the server program is stopped and restarted:



```
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Admin>cd C:\Users\Admin\IdeaProjects\DistributedSystems\RMI\src

C:\Users\Admin\IdeaProjects\DistributedSystems\RMI\src>cd C:\Users\Admin\IdeaPro
jects\DistributedSystems\RMI\out\production\RMI

C:\Users\Admin\IdeaProjects\DistributedSystems\RMI\out\production\RMI>java Clien
t
[SYSTEM] Please enter your name
TestName
[SYSTEM] Welcome to the chat
Message
[CHAT] TestName: Message
After this message, server will be stopped and reset
[CHAT] TestName: After this message, server will be stopped and reset
Message
[SYSTEM] Connection failed
[SYSTEM] Restarting
[SYSTEM] Please enter your name
Name
[SYSTEM] Welcome to the chat
Message
[CHAT] Name: Message
```

FIG. 3 A DEMONSTRATION OF THE RECONNECT ABILITIES OF THE APPLICATION

Future Work

The application can be modified slightly to enable usage over multiple machines. The files *Client.class*, *Notification.class*, *NotificationSink.class*, *NotificationSinkInterface.class*, and *NotificationSourceInterface.class* would be held and ran by the users of the chatroom whilst the files *Server.class*, *Notification.class*, *NotificationSource.class*, *NotificationSourceInterface.class*, and *NotificationSinkInterface.class* would be held and ran by the server of the chatroom. Some minor alterations would be need to made so that the sink and source connect over IP and not through the local registry. It would be in this case that managing lost connections would be more important and greater care would be needed to ensure data that arrives is valid, correct, and sanitised.

Conclusions

With regards to whether a chatroom application lends itself well to RMI, the answer is a maybe. Whilst an advantage of RMI over Sockets is that with RMI one doesn't require a protocol to interpret bytes, a simple chat application has no bytes that need to be interpreted because all bytes send from client program to server program would be a string message to be sent to all other clients. RMI perhaps overcomplicates the matter by forcing the use of interfaces. This uses up more space on disk and likely is not faster or simpler. Having said that, it's was not really much more complicated to program or understand. A much bigger and more serious problem is that RMI does not work well over the internet due to the presence of firewalls and routers which, according to the SIMON developer wiki, have issues with RMI using more than one connection and makes RMI completely unsuitable for internet applications since it may require every client to reconfigure their firewall.

To be completely honest, networking and its related subjects is my least favourite thing to program by far and I had immense dislike for the crazy errors java threw at me throughout this exercise (some which just magically seemed to fix themselves, some of which were caused by very weird processes). At least the silver lining to this was that I felt like a god when it finally worked.

To conclude, this is a solid RMI implementation for a successful albeit unambitious chatroom application.