Lab 2 Assignment

AGENT TAGGING SYSTEM

Catherine Nawire Rakama | ID2010 Programming of Interactive systems | 06/08/2018

INTRODUCTION

Most interactive systems are widely distributed and may be hosted on different webservers. For such systems to communicate and interact in the distributed environment which has different communication protocols diverse system specifications, there is a need of a communication paradigm. Communication paradigm can be direct (RMI) or Indirect (Message sending). Direct communication paradigm is good because its transparent - remote method call looks as local method call and developers don't need to worry about transport, network, data link or physical layer protocols. Its space uncoupled, the sender needs not to know about receivers IP and port number because RMI uses lookup service to get a reference to the remote object that it needs to communicate with. The down side are; RMI is time coupled, if remote object not found then the transaction is failure and does not guarantee persistence of messages, if both sender and receiver fail for instance. For these reasons majority of solutions today are implemented using message queues sending paradigm e.g. pervasive applications.

This lab implements a distributed application of mobile agents which relies on Jini infrastructure to discover available services in a distributed environment and RMI middleware to remotely invoke method of a program running on a different machine. To simulate the distributed environment, the lab contains codebases with several sub folders/directories. Distributed environment means that the programs are not installed directly on the host machine, instead they run on a web server or different web servers and a bootstrap program (JarRunner.jar) is executed to fetch the real program from the server. The applications (agents) then use Jini look up service discover each other.

Lab requirements/Tasks:

- The challenge is to create the Game of Tag for mobile software agents.
- Tagging can only be done between players in the same Bailiff
- The tag (the 'it' property) must be passed reliably from one player to another. It must not be lost or duplicated during the transaction.
- Unlike seeking to be lonely or seek company, the design choice for this assignment is to implement agents acting randomly.

IMPLEMNTATION AND EXPERIMENTS

I. The challenge is to create the Game of Tag for mobile software agents.

This task has been achieved by use of Jini infrastructure and RMI middleware for distributed environment. The two main components include the mobile software agents(Dexter) which migrate between Bailiffs during execution time. Bailiffs are playfield in which the agents migrate to and sleep for some time before doing another migration. AgentStatusType is an additional component which was extended to the program to as an enum class and is used to display the status of the agent.

II. Tagging can only be done between players in the same Bailiff

To implement this functionality, the program was extended in the following way.

A playfield is made of three Bailiffs, three agents are started, one of them with an "*IT*" status, while the other two have status *TAGGABLE*. Each playfield has a HashMap that stores currently active agents in each Bailiffs.

If an agent has IT status, it first queries the number of active agents in a randomly chosen Bailiff, if the number is greater than zero, it performs a check as to whether ***it is in*** the Bailiff. If ***yes***, then through the Bailiff's interface, the agent gets access to and invokes the ***tag()*** method of the Bailiff and starts the tagging process. If ***No,*** the agent must migrate first by sending a copy of itself to the chosen Bailiff, then terminates execution of the previous object of itself. With this second option, the agent must perform a tagging process once it successfully migrates to the new Bailiff before remote method invocation of its object. This is to ensure benevolence property of the agent, that making sure it completes what it was asked for it(tagging) and not just always migrating but no performing tag. This problem was experienced during implementation where the agent would jump from Bailiff to Bailiff with no successful tagging. It was rectified by having it perform tagging first before recreating itself. The following figures are arranged sequentially to show the tagging process.

Step 1: Agent ...490b is initialized with IT status, finds BailiffRoom3 with two agents and migrates to it.

```
crakama@crakama-Virtual Box: ~/Projects/PIS/taggingsystem/test/bin/unix$ ./dexter.sh -debug -it
 TagGame: f0ec85cb-423e-4f56-aaa8-e2b043f3490b : Status: STATUS_IS_IT    Initialised
 TagGame: f0ec85cb-423e-4f56-aaa8-e2b043f3490b : Status: STATUS_IS_IT    Entering restraint sleep.
 TagGame: f0ec85cb-423e-4f56-aaa8-e2b043f3490b : Status: STATUS_IS_IT    Leaving restraint sleep.
 TagGame: f0ec85cb-423e-4f56-aaa8-e2b043f3490b : Status: STATUS_IS_IT    Found 3 Bailiffs.
 TagGame: f0ec85cb-423e-4f56-aaa8-e2b043f3490b : Status: STATUS_IS_IT    Pings Bailiff-room_3
 TagGame: Ping echo from Bailiff of Room  [room_3]  user=username_3.
Ping Accepted.
 TagGame: Roomroom_3 : has: 2    ACTIVE agents
 TagGame: f0ec85cb-423e-4f56-aaa8-e2b043f3490b : Status: STATUS_IS_IT    Found Remote Bailiff-room_3 with 2 agents, Trying to Migrate
 TagGame: f0ec85cb-423e-4f56-aaa8-e2b043f3490b : Status: STATUS_IS_IT    Successfully Migrated to Bailiff-room_3 to TAG a victim
crakama@crakama-Virtual Box: ~/Projects/PIS/taggingsystem/test/bin/unix$
```

Figure 1: Agent ...490b is initialized with IT status

Step 2: ***Agent …490b*** with IT status arrives in BailiffRoom3, successfully Tags ***Agent…3629*** *and changes status back to TAGGABLE.*

```
TagGame: f0ec85cb-423e-4f56-aaa8-e2b043f3490b : Status: STATUS_IS_IT    Looking for Target in: room_3
TagGame: f0ec85cb-423e-4f56-aaa8-e2b043f3490b : Status: STATUS_IS_IT    TAGGING A TARGET IS IN PROGRESS...
TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    is an IT-PLAYER!!!
TagGame: f0ec85cb-423e-4f56-aaa8-e2b043f3490b : Status: STATUS_TAGGABLE    Successfully Tagged another Agent
TagGame: f0ec85cb-423e-4f56-aaa8-e2b043f3490b : Status: STATUS_TAGGABLE    Arrived in: room_3
TagGame: f0ec85cb-423e-4f56-aaa8-e2b043f3490b : Status: STATUS_TAGGABLE    Entering restraint sleep.
```

Figure 2: Agent …490b with IT status arrives in BailiffRoom3

Step 3: ***Agent…3629*** with IT status wakes up from sleep in BailiffRoom3 and Migrates to BailiffRoom2 to search for targets.

```
TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Leaving restraint sleep.
TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Found 3 Bailiffs.
TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Pings Bailiff-room_2
TagGame: Ping echo from Bailiff of Room  [room_2]  user=username_2.
ing Accepted.
TagGame: Roomroom_2 : has: 1    ACTIVE agents
TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Found Remote Bailiff-room_2 with 1 agents,Trying to Migrate
TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Successfully Migrated to Bailiff-room_2 to TAG a victim
TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Removed from room_3
```

Figure 3: Agent …3629 with IT status wakes up from sleep

Step 4: ***Agent…3629*** with IT status arrives at BailiffRoom2 looking for targets. Finds ***Agent…cfd7***, successfully tags it and changes its own status back to TAGGABLE.

Step 5: ***Agent… cfd7*** with IT status wakes up from sleep in BailiffRoom2, finds ***Agent…3629*** still around and tags it back. ***Agent…3629*** with IT status wakes up from sleep and Migrates to BailiffRoom_**1** to search for targets. At this stage, all the agents have once been IT and successfully traverse all the Bailiffs and transferred the IT property.

```
 TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Looking for Target in: room_2
 TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    TAGGING A TARGET IS IN PROGRESS...
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_IS_IT    is an IT-PLAYER!!!
 TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_TAGGABLE    Successfully Tagged another Agent
 TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_TAGGABLE    Arrived in: room_2
 TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_TAGGABLE    Entering restraint sleep.
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_IS_IT    Leaving restraint sleep.
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_IS_IT    Found 3 Bailiffs.
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_IS_IT    Pings Bailiff-room_3
 TagGame: Ping echo from Bailiff of Room  [room_3]  user=username_3.
Ping Accepted.
 TagGame: Roomroom_3 : has: 0    ACTIVE agents
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_IS_IT    Pings Bailiff-room_2
 TagGame: Ping echo from Bailiff of Room  [room_2]  user=username_2.
Ping Accepted.
 TagGame: Roomroom_2 : has: 2    ACTIVE agents
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_IS_IT    Found Local  Bailiff-room_2 with 2 agents,Trying to Tag
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_IS_IT    TAGGING A TARGET IS IN PROGRESS...
 TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    is an IT-PLAYER!!!
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_TAGGABLE    Successfully Tagged another Agent
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_TAGGABLE    Tagging-Attempt on Bailiff-room_2 finished, trying another Bailiff
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_TAGGABLE    Pings Bailiff-room_1
 TagGame: Ping echo from Bailiff of Room  [room_1]  user=username_1.
Ping Accepted.
 TagGame: Roomroom_1 : has: 1    ACTIVE agents
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_TAGGABLE    Trying to Migrate to Bailiff-room_1
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_TAGGABLE    Successfully Migrated to Bailiff-room_1
 TagGame: bf11e3aa-6404-4a92-9897-6ac6a1a3cfd7 : Status: STATUS_TAGGABLE    Removed from room_2
 TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Leaving restraint sleep.
 TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Found 3 Bailiffs.
 TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Pings Bailiff-room_1
 TagGame: Ping echo from Bailiff of Room  [room_1]  user=username_1.
Ping Accepted.
 TagGame: Roomroom_1 : has: 2    ACTIVE agents
 TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Found Remote Bailiff-room_1 with 2 agents,Trying to Migrate
 TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Successfully Migrated to Bailiff-room_1 to TAG a victim
 TagGame: 4de5aa98-71b6-4899-a51d-6d2f38193629 : Status: STATUS_IS_IT    Removed from room_2
```

Figure 4: Tagging between agent…cfd7 and Agent…3629

III. *The tag (the 'it' property) must be passed reliably from one player to another. It must not be lost or duplicated during the transaction.*

This task was challenging to implement because the program is multithreaded. Initial implementation had a problem where the different threads modifying agent's status at different time and that would make the status of the system be inconsistent, at one point all the agents had the IT property. This problem was rectified by synchronizing the threads in a Bailiff to change agent's status one at a time. In addition to synchronization of threads, additional enum class was created to contain agent's status, the status class is instantiated when the agent starts, and default status assigned. The status is later changed at runtime during the tagging process.

IV. *Unlike seeking to be lonely or seek company, the design choice for this assignment is to implement agents* ***acting randomly****.*

This requirement was implemented for agents who are Non-IT-Players. Before migrating to a new Bailiff, they will check if an IT-PLAYER is in it and avoid that Bailiff. The following simulation shows this requirement but was not recorded at the same time as the previous experiments.

```
TagGame: Roomroom_3 : has: 2    ACTIVE agents
TagGame: 4892908e-8d46-4ecb-ab2d-bf49abe0e827 : Status: STATUS_TAGGABLE    Trying to Migrate to Bailiff-room_3
TagGame: 4892908e-8d46-4ecb-ab2d-bf49abe0e827 : Status: STATUS_TAGGABLE    Avoided Migration to Bailiff-room_3 IT-PLAYER PRESENT!!!
TagGame: 4892908e-8d46-4ecb-ab2d-bf49abe0e827 : Status: STATUS_TAGGABLE    Pings Bailiff-room_1
TagGame: Ping echo from Bailiff of Room  [room_1]  user=username_1.
```

ANALYSIS AND CONCLUSION

The task was challenging to implement, especially when working with multiple threads, most of the problems experienced revolved around thread handling. The other challenging problem was to have the UI animation working, a lot of time was spent trying to get animation working than handling the real task at hand.

This implementation greatly portrayed autonomous property of agents using RMI middleware and Jini infrastructure.

At first, this assignment seemed like **synchronous model** where the client blocks after making a remote procedure call as it waits for the server's reply. The real scenario is that the client(agent) invokes server's(Bailiff) ***migrate()*** method, once a copy has been sent the agent's thread returns and the object is terminated. **Asynchronous model** is portrayed when the new object/copy object is handled by a worker thread(Agitator) which is unique for each new agent object. ***Asynchronous model*** of the chat assignment mainly involved message queues where client main thread delivers a message to the server, and the server's listening thread empties the queues upon receipt of new message.

Link of Source code on GitHub