Virtual Beamer

Distributed Systems Project - A.A. 2015-1016

Alessandro Gattolin - Daniele Marangoni - Daniele Moro

Requirements

Implement in Java a virtual beamer to share a set of slides and show them in a synchronous way under control of a single node

- Create sessions
- Available sessions discovery
- Join one session before start / after start
- Share presentation slides
- Change session leader (runtime)
- Manage crashes (leader, session creator, normal users), leader election

Create Session

- Insert session name
- Select presentation slides (ordered jpg files)
- Wait for users to join, opening a server socket

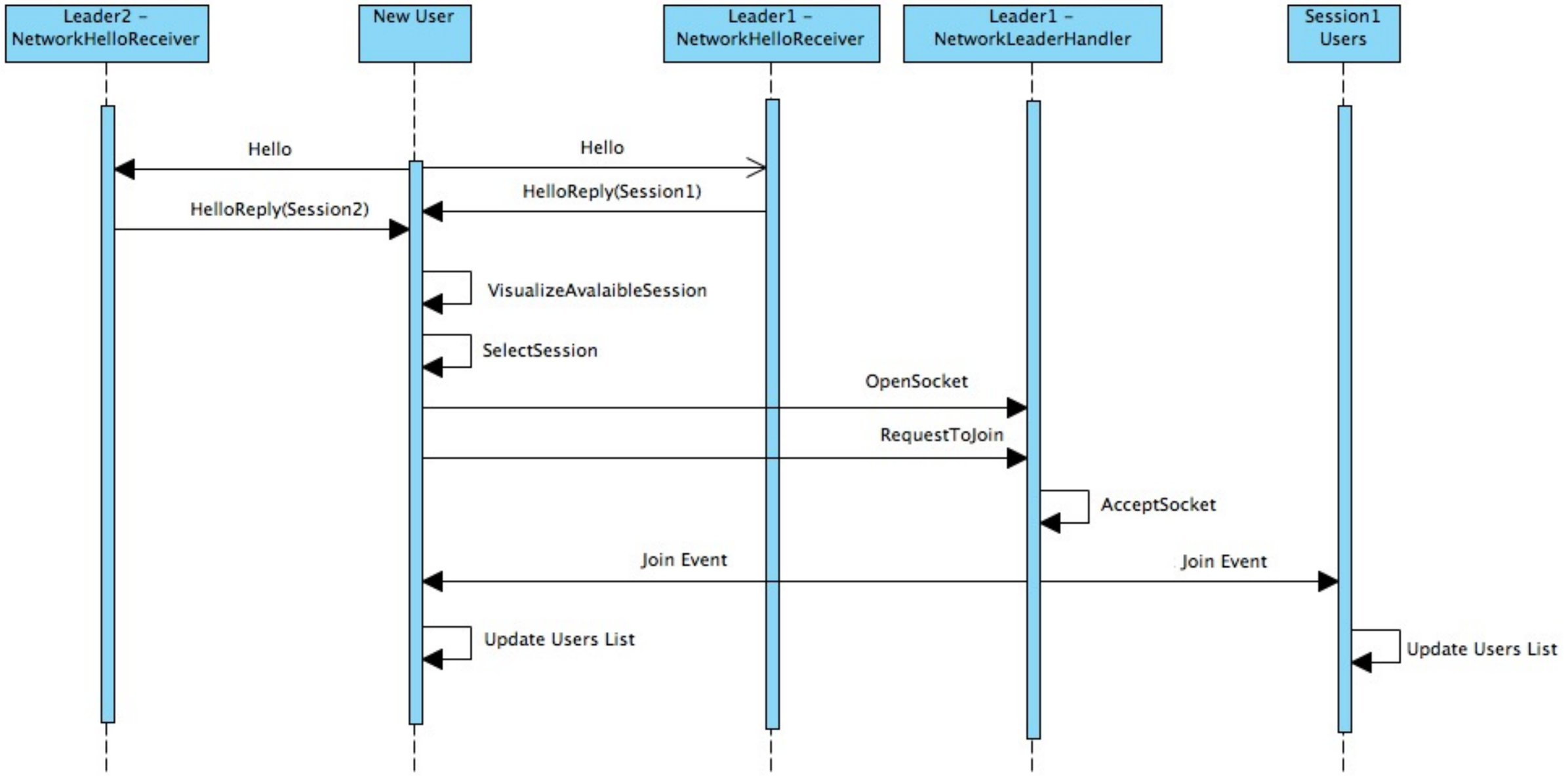
Available sessions discovery

- Global multicast group
- New user requests the available sessions on the global multicast group, sending an “hello” packet
- Each session leader replies with the necessary information to join the session

Join

- Selection of an available session
- Request to join to the session leader opening a connection to the server socket
- The leader notifies all the already connected users about the new user

Join


Leader2 - NetworkHelloReceiver
New User
Leader1 - NetworkHelloReceiver
Leader1 - NetworkLeaderHandler
Session1 Users
Hello
Hello
HelloReply(Session1)
HelloReply(Session2)
VisualizeAvalaibleSession
SelectSession
OpenSocket
RequestToJoin
AcceptSocket
Join Event
Join Event
Update Users List
Update Users List

Share presentation slides

- Slides sender creates the packets from the slides (UDP payload dim 65507) and sends them all
- Each receiver stores the packets in a data structure; if it receives a packet out of order, it sends a NACK for the previous packet
- If the user doesn’t receive any packet, after the timeout expires, it sends a NACK for each packet it misses
- When the user has all the slides, it sends an ACK
- Before session start: download from session creator
- After session start: download from a single user

Change session leader

- Old leader selects one of the other users to become the new leader and notifies everyone about it
- All users close the socket to the old leader
- New leader opens a server socket
- Other users connects to the new leader’s server socket

Manage Crashes

- All users send periodically an ALIVE message in a specific multicast group
- If a user doesn't receive an ALIVE message from another user for a predefined amount of time, it considers that user as crashed and decides what to do
- If the crashed user is a normal one it simply removes it from the list of session users
- If the crashed user is the leader and the session creator is active, the session creator becomes the new leader
- If the crashed user is the leader and the session creator is crashed too, it starts the leader election algorithm

Leader Election

Bully Election

- When a user starts the Bully Election, it sends an “Elect” message to each other user

- When a user receives an Elect message, it checks if its own id is higher than the one of the sender: in the positive case, it sends a “Stop” message to that user and starts a new election

- If a user has started the election and, after a timeout, hasn’t received any Stop message, it is the new leader, so he sends a Coordinate message to everyone

Pattern MVC

- Model: stores the main information on the status of the presentation (slides, connected users, session leader, session creator)

- View: displays the presentation on the screen

- Controller: modifies the information in the model and what is displayed by the view according to the leader's actions and the users' status.

Runtime Architecture - Leader

- **Network Leader Handler**: it manages the communication between the leader and all other users. It opens a server socket the other users have to connect to.
  Thread:
  - Connection_Server: it manages the server socket, creating a Network Handler to handle the communication to specific users.

- **Network Hello Receiver**: his aim is to receive the “Hello” messages from new users, in order to reply indicating the availability of the session. It performs its work exploiting a thread.

Runtime Architecture - Client

- **Network Handler**: its main tasks are:
  - open a socket to the leader;
  - receive messages from the leader and reply to it;
  - notify the controller about the received events.

  The last two tasks are performed by a thread.
  NB: communications between server and clients are mainly through serialized objects, in particular “events”, containing all the necessary information.

Runtime Architecture - Leader and Client

- **Crash detector**: it is used to detect if a user is crashed and to manage the leader election. It exploits a specific multicast group reserved for the session.
  Timers:
  - Timer_Alive: it is used to send periodically an “Alive” message to other users;
  - Timer_Increment: periodically increments the counters related to the users.
  - Check_For_Election_Confirm: after a fixed amount of time, if the user that has start the election doesn’t receive any “Stop” message, it becomes the new leader.

  Thread:
  - Receiver_Alive: it listens for all the messages sent in the multicast group and, according to the particular message, performs some operations.