Remote Method Invocation

Sistemas Distribuidos

Aldo Amezcua – Vidal Ortiz

Problemas encontrados y sus propuestas de solución

- **Problema:**
  Falta de conocimiento de la arquitectura de RMI y del funcionamiento del plugin de Eclipse.
  Solución:
  Investigación y seguimiento de un tutorial de RMI y uno de Genady, así como con consulta de foros.

- **Problema:**
  Problemas al intentar correr el cliente y el servidor sobre distintas maquinas en la misma red.
  Solución:
  Apagar firewalls y configurar una red ad hoc entre las maquinas.

- **Problema:**
  Errores de comunicación cuando se lanzaban múltiples registries (para simular multiples servidores).
  Solución:
  Arrancar los registries desde el código del servidor.

- **Problema:**
  Sincronizar acceso a la memoria de los objetos del servidor.
  Solución:
  Añadir bloques de sincronización para las condiciones de carrera entre cliente y servidor, así como entre servidores, diseñándolos para evitar deadlocks, utilizando la estrategia de ordenamiento de recursos (los locks siempre se consiguen en el mismo orden).

- **Problema:**
  Manejar las posibles excepciones de la arquitectura RMI. Por ejemplo, si un server o un cliente mueren.
  Solución:
  En el caso del server, si la comunicación con un cliente falla, el cliente se remueve de la lista de usuarios. En el caso del cliente, si la comunicación con el server falla, el cliente se suicida. En el caso de comunicación entre servers, si la comunicación con el servidor maestro falla, se asigna un servidor maestro nuevo. Si la comunicación con un servidor esclavo falla, el servidor esclavo se remueve de la lista de servidores.

- **Problema:**
  Sincronización del estado global (número y secuencia de mensajes y lista de usuarios).
  Solución:
  Serialización de los cambios al estado global desde un servidor maestro (Token Master).
- **Problema:**
  Elegir un servidor maestro por primera vez y cuando el actual muere.

Remote Method Invocation

Sistemas Distribuidos

Aldo Amezcua – Vidal Ortiz

Solución:
Cada servidor tiene asignado un identificador único, los cuales tienen un orden total entre ellos. El líder será el servidor con el identificador más bajo dentro de los servidores que se encuentran corriendo en ese momento. Se implementó un algoritmo para descubrir los servidores que se encuentran activos en un momento dado, que lidia con los problemas de concurrencia.

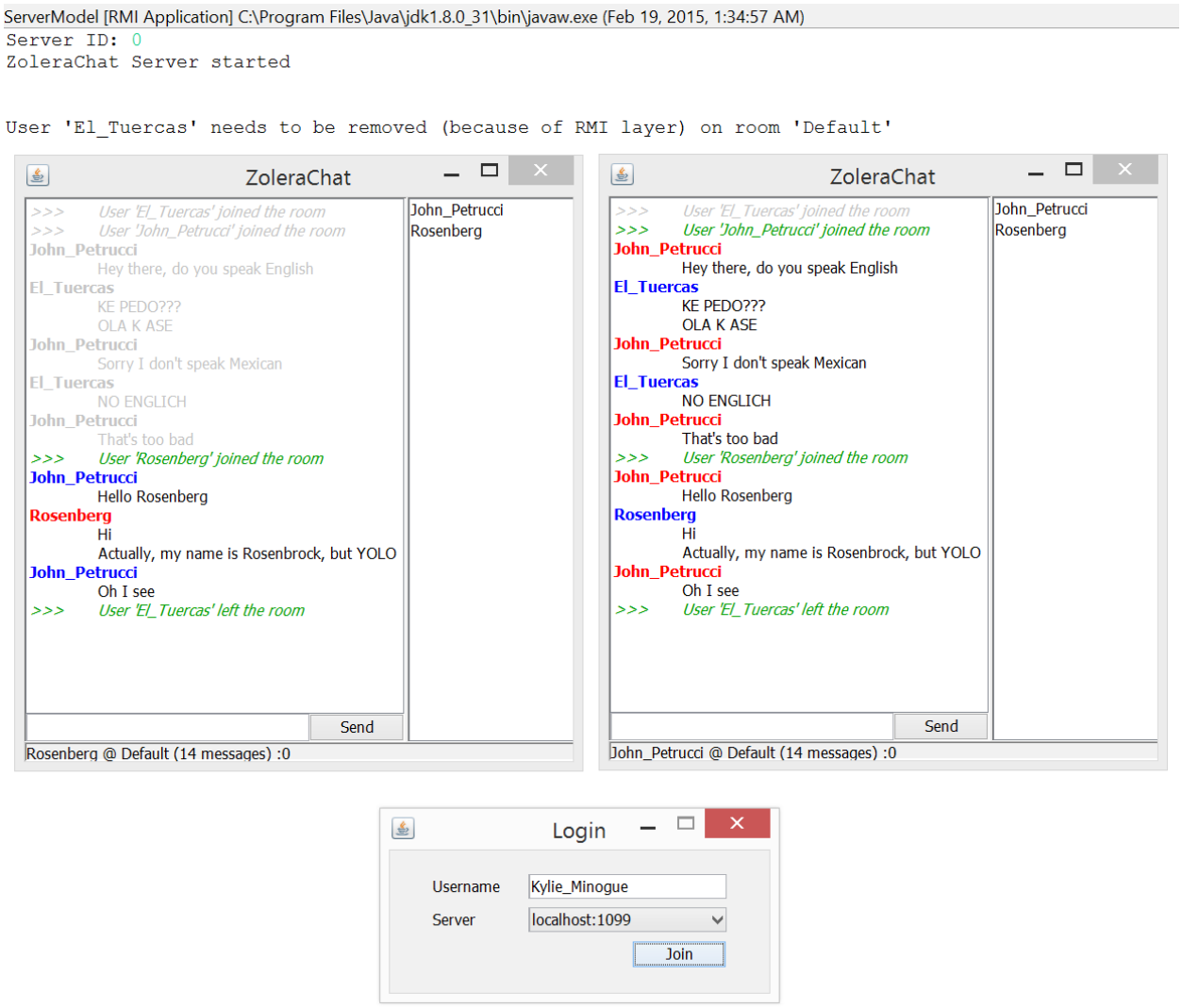
- **Problema:**
  Descubrir nuevos servidores que se incorporen al cluster.
  Solución:
  Se creó una lista fija de potenciales servidores. Cuando un server se une recorre dicha lista y se registra con cada servidor activo. Cuando otro servidor que no estaba activo en el momento inicial se une, este se registra con el resto.

Remote Method Invocation

Sistemas Distribuidos

Aldo Amezcua – Vidal Ortiz

Captura de pantalla de funcionamiento (1 servidor, n clientes)



Repositorio del código de las aplicaciones

https://github.com/adavzlr/ZoleraChat