Disciplina: Sistemas Distribuídos

Trabalho: RMI – Servidor de Tempo

Alunos:
Raul Vieira Cioldin – 379468
Marcelo Takao Goto – 379522

Professor:
Luis Carlos Trevelin

Implementação

Linguagem: Java JDK 1.7

Sistema Operacional: Arch Linux (Kernel 3.9)

Principais componentes:

- java.rmi

Arquivos:

- RelogioRMI.java
- RelogioRMIImpl.java
- RelogioRMIClientInterface.java
- RelogioRMIClient.java

Implementação:

Para assegurar a assincrocidade da aplicação utilizamos a tecnologia de callback em RMI, o que evita ter que registrar um novo nome no rmiregistry para cada cliente (uma vez que devem operar também como "servidor" na hora que a aplicação desejar saber o tempo do cliente pra fazer a sincronização). Isso é feito por uma estruturação de objetos remotos tanto do lado "cliente" como "servidor" (note que aqui é uma separação semântica, uma vez que os dois lados atuam servindo e requisitando objetos), ambos extendendo a classe **java.rmi.server.UnicastRemoteObject** (isso permite que os objetos sejam serializáveis para os dois lados) e implementando uma interface própria, definida em **RelogioRMI.java** para o servidor e **RelogioRMIClienteInterface.java** para o cliente.

Os clientes se registram adquirindo uma instância do servidor pelo serviço de lookup do **RMIRegistry**, e invocando a função **register**().

A sincronização é feita por um laço infinito da função **doSync()** no servidor, que acessa regularmente a lista de clientes registrados (contida em **listaClientes**, um java.util.Vector), itera sob essa lista fazendo cast pra objeto de cliente e requisitando o tempo pela função remota **getTimeMillis()** (retorna um objeto Long que representa tempo) e tirando a média com o tempo do servidor. A partir disso, envia de volta o tempo médio para os clientes (de maneira serializada) através da função **setNewTime()** dos objetos remotos dos clientes. Os clientes serão atualizados apenas se o tempo recebido for superior ao tempo atual (a partir do recebimento do novo tempo e não do envio para cálculo da média), portanto, em ambiente local, quase nunca essa condição será real.

Instruções:

- Primeiro compile as classes do programa, acesse o diretório em que os arquivos estão e digite:

  javac *.java

- O RMIRegistry deve ser iniciado, você deve ter conhecimento da porta em que o serviço está rodando (por padrão, 1099)

- Inicie o servidor especificando o arquivo de policy e a porta do serviço de RMIRegistry, por exemplo:

  java -Djava.security.policy=java.policy -classpath . RelogioRMIImpl 1099

- Inicie instâncias de cliente especificando o arquivo de policy e <hostname>:<porta> como parâmetro, por exemplo:

  java -Djava.security.policy=java.policy -classpath . RelogioRMIClient localhost:1099

Observações:

- É necessario permissão de administrador para mudar o tempo do sistema operacional se assim você deseja.