logoISEL.png

Licenciatura em Engenharia Informática e de Computadores

SISTEMAS DISTRibuídos, verão 2016/2017

Relatório do 1º trabalho prático de avaliação

Grupo 3

41839 – André Carvalho

41482 – gonçalo veloso

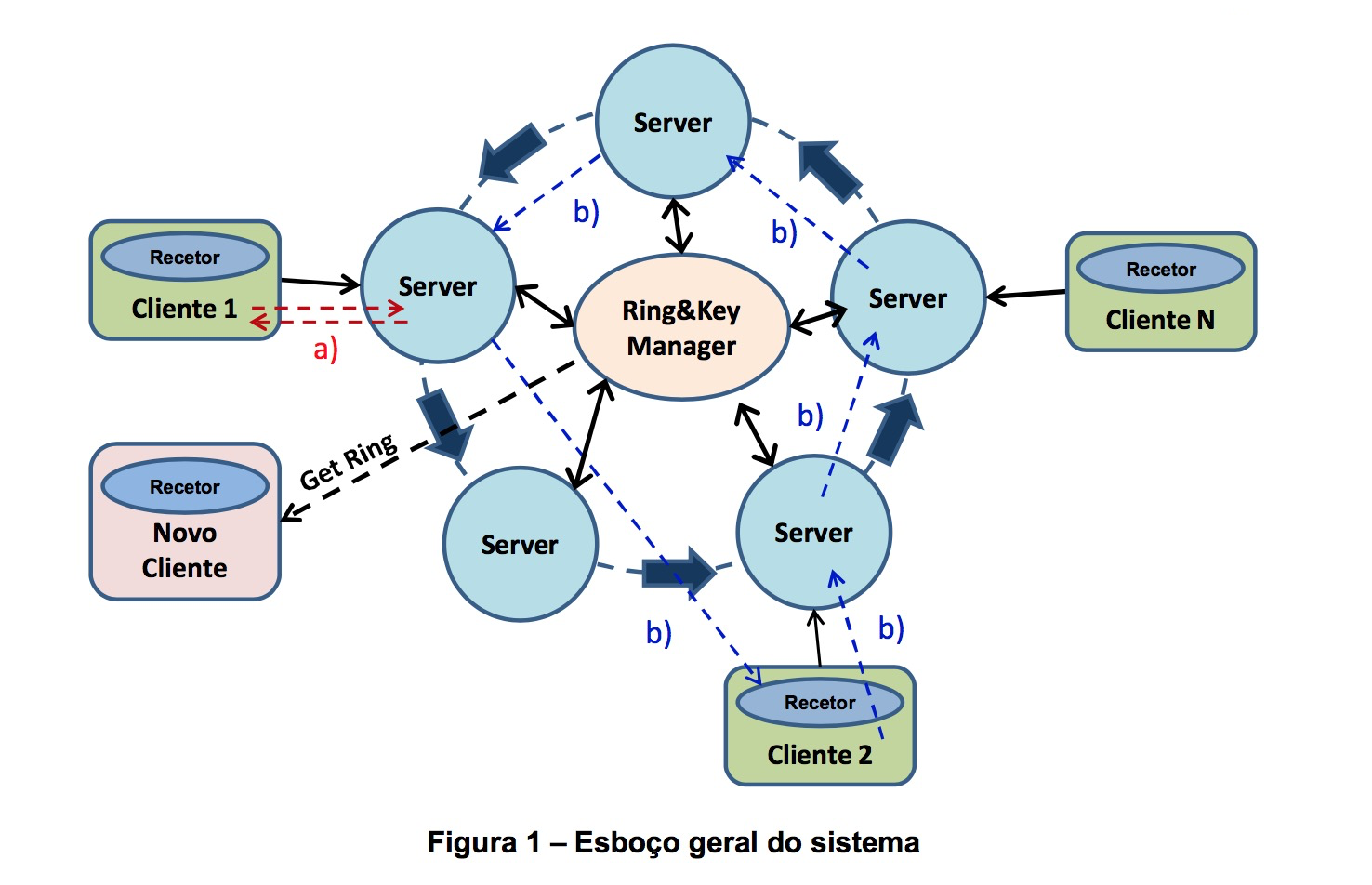
39134 – rúben taborda

índice

introdução

No âmbito da unidade curricular de Sistemas Distribuídos foi proposto o desenvolvimento de sistemas distribuídos usando objetos distribuídos na plataforma .NET.

Neste documento é descrita como foi concebida a solução para o problema proposto.

O cenário apresentado na Figura 1 é caracterizado por uma rede de múltiplos servidores, onde poderão ser guardados pares (Chave, Valor) emitidos pelos diversos clientes.Os servidores (Server) estão ligados em anel a um servidor central.

arquitectura do sistema

Apesar de ser explícito no enunciado que a solução pretendia que existisse 5 servidores, graças a ficheiros de configuração é possível ter N servidores.

Contudo existem algumas restrições de implementação:

* Cada servidor pode ser conectado a mais do que um cliente, caso o número de clientes seja superior ao número de servidores disponíveis.
* Cada Servidor tem uma ligação indirecta com os restantes Servidores, formando assim um anel.

Registo de uma par chave-valor

Quando um cliente efectua a operação de push de um par-valor para o servidor associado, este irá efectuar as seguintes operações:

1. Colocar um pedido ao Manager, a perguntar se a chave já existe nalgum servidor.
2. Caso a chave não exista, este irá guardar a chave nele mesmo.Depois disso irá colocar um pedido ao Manager a pedir que ele ordene os servidores adjacentes a realizarem uma copia.

Eliminação de uma par chave-valor

Quando um cliente efectua a operação de delete de um par-valor para o servidor associado, este irá efectuar as seguintes operações:

1. Apagar a informação local, se esta existe.
2. Enviar um pedido ao servidor central para que este apague a informação correspondente de outros servidores.

Ao receber o pedido, o anel efectua as seguintes operações:

1. Verifica na lista local de chaves quais os servidores que possuem informação para essa chave.
2. Pedir a esses servidores que apaguem essa informação.
3. Apagar essa chave da sua lista de chaves.
4. Retornar com sucesso ao servidor que efectuou o pedido ou lançar exceção em caso de falha. O servidor por sua vez faz o mesmo ao cliente que efectuou o pedido.

Estrutura e descrição da solução

Nesta capítulo descrevemos a estrutura adoptada pelo grupo. Temos a plena noção que existem outras soluções para o problema, mas falaremos apenas da nossa solução, sabendo que em engenharia informática não existem soluções perfeitas.

A estrutura da nossa solução está organizada em sete projetos embebidos numa solução. Tentámos separar a parte lógica de negocio com a parte da user interface. Temos um projeto de nome, Interfaces, que corresponde à uma biblioteca com todas as interfaces necessárias para a solução. O assembly desta biblioteca é partilhado pelo manager, pelo servidor como pelo cliente. Os projetos com o nome ServerClass, ManagerClass e ClientClass tratam da lógica de negócio da solução, implementando as interfaces da biblioteca referida anteriormente. Os projetos Cliente, Server e Ring são referentes à user interface respetivamente do cliente, do server e do RingManager.

De seguida é apresentado uma descrição mais pormenorizada de cada projeto e as relações existentes em cada um deles.

Server

Os servidores, após estabelecerem ligação com o Ring e com um ou mais clientes, estão encarregados de receber pedidos do cliente, processa-los localmente e envia-los para o Ring. Todos os pedidos possíveis correspondem a alterações locais e alterações remotas, excepto o método **deleteKeylocally** que apenas é chamado pelo Ring para apagar a chave localmente e não propagar essa informação, visto que o Ring já se está a encarregar disso quando chama esse método.

Ring&Key Manager

O Ring&Key Manager, estabelece uma conexão entre ele e um servidor contém a informação sobre as chaves atribuídas e a que servidores estas estão associadas. Contém métodos de verificação, obtenção e tratamento de falhas por parte dos servidores.

O método **checkIfKeyExists** trata de verificar se para o servidor desejado existe a chave.

O método **ReplicateInformationBetweenServers** trata de verificar se é possível a replicação no servidor +1, +2 e trata as falhas destes porque se um destes falhar replica para o seguinte, mantendo sempre 2 réplicas para além do original. Para tal este método exige o uso do método **storePairLocally** para guardar os pares pelos seguintes servidores.

O método **deleteInformation** como o nome indica serve para eliminar os pares chave/valor para os servidores que contiverem a chave a ser eliminada, este usa o método **deletePairLocally** para cada servidor que contenha tal chave.

O método **searchServersForObject** procura pelo servidor que contenha a chave desejada e encarrega-se por obter o valor desta.

Conclusão

Optamos por uma solução sem muitos requisitos funcionais para concentrar o nosso trabalho maioritariamente nos requisitos não funcionais.

Mesmo assim tivemos partes que não conseguimos implementar como o dinamismo do anel e não conseguimos garantir todos as falhas que possam ocorrer estão tratadas na nossa solução.