MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
UNIVERSIDADE DO MINHO

Sistemas Distribuídos

Época Especial[1]

15 de setembro de 2020 Duração: 2h00m

I

Responda a cada um dos 2 grupos em folhas de teste separadas.

Não ultrapasse 1/2 página (15 linhas) em cada resposta neste grupo.

1 Identifique os componentes principais de um sistema de *invocação remota* em sistemas distribuídos, descrevendo o objetivo de cada um deles.

2 Quais os principais problemas que se resolvem com *migração de código* em sistemas distribuídos? Justifique.

3 Diga o que entende por *transparência de localização* em sistemas distribuídos e explique qual é o método genérico para a atingir.

II

Responda a cada um dos 2 grupos em folhas de teste separadas.

Considere um sistema que gere um conjunto de pessoas à espera de ser atendidas, identificadas pelo seu nome. Uma vez que se admite que a espera pode ser demorada, a desistência de uma pessoa pode ser assinalada. Assuma que a ordem de atendimento não é relevante.

1 Pretende-se que escreva em Java, fazendo uso de primitivas baseadas em monitores, uma classe que implemente a seguinte interface para ser usada no servidor por *threads* que atendem pedidos de clientes remotos:

```
interface SalaDeEspera {
    boolean espera(String nome);
    void desiste(String nome);
    String atende();
}
```

A chegada de uma pessoa é assinalada com a invocação do método *espera*, que espera até ser atendida (resultado `true`) ou até ter desistido (resultado `false`). A desistência de uma pessoa que está à espera é assinalada com a invocação do método *desiste*. Para saber qual a próxima pessoa a ser atendida, retirando-a da sala de espera, invoca-se o método *atende* que devolve `null` no caso de não haver ninguém à espera.

Valorização: *i)* Garanta que as pessoas são atendidas estritamente por ordem de chegada. *ii)* Permita que possam ser atendidas até n pessoas de cada vez acrescentando um método `List<String> atende(int n)`, que espera por pelo menos uma pessoa. Indique na sua resposta quais destas garantias oferece e minimize a necessidade de acordar *threads* bloqueados.

2 Implemente o programa servidor usando *threads*, *sockets* TCP, e a classe desenvolvida na pergunta anterior.

Valorização: Implementa também o suporte no servidor para o método `List<String> atende(int n)`. Minimize o número de operações nos *sockets*.

[1] Cotação — I: $3 + 4 + 3$; II: $7 + 3$