Nome: _____ **Número:** _____

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
UNIVERSIDADE DO MINHO

Sistemas Distribuídos

Teste[1]

11 de Janeiro de 2019 Duração: 2h00m

I

1 Distinga *escala geográfica* de *escala numérica* em sistemas distribuídos e identifique técnicas usadas para as atingir.

Escala Geográfica : Dentro de um sistema onde os recursos e os utilizadores se encontrem afastados, a escala geográfica faz com que esta distancia não cause delays significativos na comunicação entre ambos. Ou seja, não importa a distancia a que a sua localização fisica fica, a comunicação não é gravemente afetada.
Uma das Tecnicas para atingir a escala geográfica é a replicação de dados ou a criação de caches para os locais mais proximos dos recursos/utilizadores., assim como os dados se encontram mais proximos, os problemas relativos a latencia é diminuida.

Escala Númerica: Dentro de um sistema, a quantidade de clientes/pedidos mesmo que aumente em quantidade não alterou diminui a performance. Isto é um sistema que se diga escalado numericamente, é possivel inserir Utilizadores e recursos sem a diminuição significativa de performance. Uma Tecnica para atingir este tipo de escalabilidade é aumentar a quantidade de Hardware/Servidores para poderem corresponder auma maior quantidade de utilizadores.

2 Defina *transparência de acesso* e explique em que medida é que a *invocação remota* (RPC) contribui para a obter.

Transparência de Acesso:É ocultado do usario/programador que determinados recusos no Sistema distribuído pode ser acessado e é também ocultado as diferenças de representação de dados.O usuario não deve saber se o recurso acedido é local ou remoto.

A transparencia de acesso faz com que o sistema não tenha que fornecer a localização dos recursos, ou seja, os programas devem executar os processos de leitura e escrita de arquivos remotos da mesma maneira que operam sobre os arquivos locais,sem qualquer modificação no programa.É desta maneira que o RPC ajuda a cumprir a transparência de acesso, pois como o RPC encapsula as rotinas de acesso e consulta como também efectua o controlo de concorrência do SD(Comunicação entre entidades).

3 Explique uma forma de mitigar a incerteza quanto ao tempo de transmissão de mensagens para conseguir sincronizar relógios em sistemas distribuídos.

Um metodo é através do protocolo dos relógios de Lamport.
Para implementar o algoritmo dos relógios de Lmaport, cada processo A mantem um contador C(i). Estes contadores são atualizados conforme as seguintes etapas:
- Antes de executar um evento(enviar uma mensagem para uma network, entregar mensagem a uma app ou quaque outro evento interno), 'A' incrementa C(i) : C(i) <- C(i) +1;
- Quando o processo A envia uma mensagem M para um processo Pj, define um timestamp ts(m) em M igual a C(i) após ter executado a ultima ação;
- Na receção de uma mensagem M, o processo Pj ajusta o seu counter local para C(j)- max {C(j), ts(m)}, o qual depois executa a primeira etapa e envia a mensagem para a aplicação.

[1] Cotação — 10+10

II

Considere um serviço de transferência de passageiros. Assuma 5 terminais, um percurso circular (ou seja, 1–2–3–4–5–1 repetidamente), e um shuttle com capacidade para 30 passageiros. O shuttle pára em cada terminal para permitir saída e entrada de passageiros. Por questões de rentabilidade, o shuttle só viaja com pelo menos 10 passageiros, esperando por mais se necessário. A viagem entre terminais demora 5 minutos. Cada passageiro utiliza uma aplicação (cliente) que interage com o servidor que controla o sistema, devendo permitir: o passageiro requisitar ao servidor uma viagem entre o terminal onde está (origem) e o terminal de destino; o servidor informar o passageiro que pode entrar no shuttle; o servidor informar o passageiro que chegou ao seu destino.

1 Apresente uma classe (para ser usada no servidor) que implemente a interface abaixo, tendo em conta que os seus métodos serão invocados num ambiente multi-threaded.

```
interface Controlador {
  void requisita_viagem(int origem, int destino);
  void espera(int destino);
}
```

O método `requisita_viagem` deve bloquear até o passageiro poder ser informado que pode entrar no shuttle (o shuttle chegou à origem e há lugar disponível). O método `espera` deve bloquear até o shuttle ter chegado ao terminal `destino`. Caso haja mais passageiros num terminal do que lugares disponíveis, os passageiros devem entrar por ordem de requisição. Nota: esta interface deve considerar o uso dos seus métodos apenas por threads que representam passageiros; considere a possibilidade de criar threads auxiliares na sua implementação.

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