Nome: _____ **Número:** _____

MESTRADO INTEGRADO EM ENGENHARIA INFORMÁTICA
UNIVERSIDADE DO MINHO

Sistemas Distribuídos

Exame[1]

1 de fevereiro de 2019 Duração: 2h00m

I

1 Distinga comunicação síncrona de assíncrona em sistemas distribuídos. Dê exemplos de middleware para cada uma delas.

Comunicação Síncrona : O emissor e o recetor das mensagens estão sincronizados, ou seja, +para haver comunicação o cliente e o servidor têm de estar sincronizados. O envio e a recepção de mensagens são ambas ações bloqueantes, isto é, quando um cliente/emissor emite uma mensagem este fica bloqueado até obter uma resposta e enquanto o recetor/servidor não recebe uma mensagem este fica também bloquado.

Comunicação Assíncrona : Neste caso, o envio não é uma ação bloquante, ou seja, o emissor pode enviar uma mensagem e continuar a sua execução logo após o encio da mesma, isto porque a mensagem é copiada para um buffer/queue de mensagens. A transmissão destas mensagens ocorre em paralelo com a execução do emissor. No caso da receção de mensagens, a recepção tanto pode ser uma ação bloqueante ou não.

Exemplos de MiddleWare para a comunicação síncrona são: Message passing em MOM (MiddleWare Orienteado a Mensagens)CORRIGIR
Exemplos de MiddleWare para a comunicação assíncrona são: Message Queuing em MOM (MiddleWare Orienteado a Mensagens) CORRIGIR

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 Identifique uma aplicação e descreva succintamente o funcionamento de um *relógio lógico de Lamport* num sistema distribuído.

Para sincronizar relógios lógicos, Lamport definiu uma relação Happened-Before(->):
- Se A e B são eventos do mesmo processo e A foi executado antes de B, então A->B.
- Se A é o evento de envio de uma mensagem por um processo e B o evento de recepção dessa mensagem por outro processo, então A->B.
- Se A->B e B->C, então A->C.

Para a realização destas relações, é associado uma etiqueta temporal a cada evento do sistema, de forma a que se A->B então a etiqueta de A é menor que a etiqueta de B, isto é:
- Cada processo tem um relógio lógico associado. O relógio é um contado que é incrementado entre cada dois eventos sucessivos do processo.
- Cada mensagem enviada transporta o instante lógico em que foi enviada.
- Ao receber uma mensagem, o processo acerta o seu relógio com o instante da mensagem se este último for mais recente.

Um exemplo de aplicação da utilização dos relógios de Lamport é uma Base de Dados replicada em várias cidades.

[1] Cotação — 10+10

II

Considere a gestão de uma ponte móvel sobre um canal navegável (talvez conheça a ponte móvel do porto de leixões). Quando o tabuleiro da ponte se encontra em cima podem passar barcos, mas apenas um de cada vez. Quando se encontra em baixo podem passar múltiplos carros. Sempre que chega um barco que pretende passar, este ainda tem de aguardar 5 minutos, após os quais deixam de ser admitidos novos carros no tabuleiro da ponte. Quando o tabuleiro da ponte fica vazio de carros, este pode levantar para admitir a passagem de barcos, voltando a admitir a passagem de carros quando não há nenhum barco a passar.

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 Control {
  void entra_carro();
  void sai_carro();
  void entra_barco();
  void sai_barco();
}
```

O método `entra_carro` deve bloquear enquanto houver algum barco a passar ou a pedir permissão pelo uso da ponte, sendo o retorno do método indicação que o carro pode entrar na ponte. Já o método `entra_barco` deve aguardar 5 minutos e tentar obter permissão de entrada, apenas retornando quando o acesso for possível, sendo que só é admitido um barco de cada vez. Os métodos `sai_carro` e `sai_barco` indicam que cada um destes tipos de transporte já abandonaram a ponte após efectuarem a travessia.

2 Implemente o programa servidor usando threads, sockets TCP, e a classe desenvolvida na pergunta anterior. Este programa deve receber do cliente indicação do tipo de veiculo (carro ou barco) e quantos minutos este demora a passar a ponte, invocando nas alturas apropriadas os métodos indicados no interface.