

Simulado nº 1

1) Assinale a alternativa correta (V ou F)

- () Abertura, transparência, concorrência, e compartilhamento são características de um SD.
- () Abertura e compartilhamento são características que indicam que um SD pode ser estendido para suportar mais usuários/máquinas.
- () Sistemas Multicomputadores e Multiprocessadores são sistemas com memória distribuída.
- () A principal diferença entre um NOS e um DOS é que o NOS esconde a multiplicidade do hardware dos elementos de software enquanto o DOS se preocupa em compartilhar os recursos entre os elementos de software.
- () Na migração de processos, a política iniciada pelo transmissor possibilita que uma máquina ociosa inicie a busca por processos em máquinas que estão sobrecarregadas.
- () Duas threads podem atribuir valores a uma variável de maneira a deixá-la inconsistente.
- () A adoção de middleware, como CORBA, possibilita maior transparência de localização do que a utilização de *sockets*.
- () Ao adotar um serviço de nomes (SN), toda requisição de um objeto cliente para um objeto servidor deve, obrigatoriamente, passar por esse serviço já que é ele (o SN) o responsável pela contacto entre cliente e servidor.
- () O algoritmo bully consegue eleger um líder mesmo quando um líder eleito “cai” antes de enviar a mensagem de líder.
- () O algoritmo do anel necessita de $2(N - 1)$ mensagens, $N = \#$ de processos, no pior caso, para eleger um líder.
- () Uma falha no algoritmo distribuído de exclusão mútua acontece quando qualquer um dos processos não respondem às requisições. Essa falha acontece no algoritmo *token ring* quando um processo deseja entrar na região crítica mas o *token* está no próximo processo, na organização do anel.
- () De maneira geral, a construção de um servidor utilizando *sockets* adota as seguintes primitivas (nessa ordem): *socket*, *bind*, *listen*, *read*, *accept*, *write*, *close*.
- () *Middlewares*, como CORBA, possibilitam desenvolver aplicações distribuídas em um nível mais próximo aos protocolos da camada de transporte que a biblioteca *Sockets*.

2) Diferencie as arquiteturas de *threads*: *thread* por requisição, *thread* por conexão e *thread* por objeto.

3) Considerando o algoritmo *Bully*, mostre que, no pior caso, N^2 mensagens, tal que $N = \#$ de processos do SD, são necessárias para que um líder seja eleito.

- 4) Mostre que o algoritmo centralizado utiliza menos mensagens para entrar na região crítica que o algoritmo distribuído.
- 5) Faça um diagrama para cada modelo de desenvolvimento de aplicações distribuídas (*sockets* e CORBA), indicando o processo de desenvolvimento. Além disso, diferencie essas duas abordagens indicando a possibilidade de utilização de serviços de nomes e nível de abstração para os protocolos de rede.
- 6) Queremos desenvolver uma aplicação distribuída usando *Sockets* que, dado um valor na linha de comando (`args[0]`), envia esse valor para um servidor. O servidor deve receber esse valor e enviar uma mensagem do tipo “OK” como resposta. Considere os códigos do cliente e servidor a seguir que devem implementar essa aplicação distribuída. Indique quais linhas, no código, contém erros e o que é necessário para corrigi-los.

– **Cliente**

```
1) Socket s = null;
2) int serverPort = 7896;
3) int send=10;
4) s = new Socket(InetAddress.getByName(args[0]), serverPort);
5) DataInputStream in = new DataInputStream( s.getInputStream());
6) DataOutputStream out = new DataOutputStream(s.getOutputStream());
7) out.writeUTF(args[0]);
8) out.writeUTF(Integer.toString(send));
9) String data = in.readUTF();
```

– **Servidor**

```
1) int serverPort = 7896;
2) DataOutputStream in;
3) DataInputStream out;
4) ServerSocket listenSocket = new ServerSocket(serverPort);
5) while(true) {
6)     Socket clientSocket = listenSocket.accept();
7)     in = new DataInputStream( clientSocket.getInputStream());
8)     out =new DataOutputStream( clientSocket.getOutputStream());
9)     out.writeUTF("OK");
10)}
```