INE5645 – Programação Paralela e Distribuída – Turma B - Prof. Bosco - 05/07/2011

Nome _____BOSCO - GABARITO DA PROVA 2 _____

Indique Verdade (V) ou Falso. Quando for falso corrija a definição ou explicação dada, colocando a frase que exprime a correção conceitual.

Excetuando as questões de Verdade/Falso, existem questões que contém conceitos que precisam ser explicados entre os parênteses colocados logo após o conceito. Onde não existir parênteses, apenas complete a frase explicativa.

Cada questão rigorosamente correta vale 1 ponto.

1. (V) RPC – Remote Procedure Call - serve para desenvolver aplicações distribuídas cliente-servidor com processos locais e remotos.
   (F) RMI - Remote Method Invocation – serve para desenvolver aplicações distribuídas cliente-servidor com threads executadas assincronamente.

2. (V) RPC utiliza sockets para a comunicação entre um cliente-servidor.
   (F) RMI utiliza sockets entre os módulos de comunicação cliente-servidor com objetos.

   RMI utiliza sockets entre os módulos de comunicação cliente-servidor com objetos. RMI utiliza stubs e skeletons associados a cada dos objetos-servants da lógica da aplicação, os quais implementam transparentemente toda a comunicação entre cliente e servidor.

3. (F) Sockects TCP stream utiliza o protocolo UDP com datagramas.
   Sockects TCP stream utiliza o protocolo TCP.

   (V) Multicast Socket utiliza um endereço IP único para um grupo de membros participantes no multicast.

4. (V) Importância de transparência em RMI - Todas as etapas necessárias para empacotamento e troca de mensagens, mais a tarefa de localizar e contactar um objeto remoto, são ocultadas do programador que faz uma chamada remota.

   (F) Em um sistema distribuído com transparência de acesso, o cliente não precisa saber da localização do servidor. Em RMI, o objeto que está fazendo a invocação não pode identificar se o objeto é local ou não.

Em um sistema distribuído com transparência de localização, o cliente não precisa saber da localização do servidor. Em RMI, o objeto que está fazendo a invocação não pode identificar se o objeto é local ou não.

(V) Nos programas distribuídos baseados em eventos, os objetos que geram eventos e os objetos que recebem notificações desses eventos, não precisam conhecer a localização um dos outros. Porque, existindo um servidor entre os objetos que geram eventos e os objetos que se inscrevem em eventos e recebem notificações, as comunicações entre os primeiros e o servidor se passam independemente das comunicações do servidor com os segundos.

5. Middleware é uma camada de software que fornece um modelo de programação (apropriado para aplicativos distribuídos, ou seja, para implementação de aplicações com objetos distribuídos compostos de programas que estão em cooperação, excutados em várias máquinas), e que é situado entre camadas de software (no nosso caso, o Java RMI, entre as camadas de Aplicação e de Transporte que usa um protocolo baseado em mensagens entre objetos (locais e remotos), no sentido de fornecer abstrações de mais alto nível (o que é de baixo nível fica oculto pela transparência de localização), como as invocações e eventos remotos, proporcionando transparência de localização nas invocações remotas.

6. **Protocolos de comunicação em middleware**: Os protocolos que suportam as abstrações/funcionalidades de um *middleware* são independentes dos protocolos de transporte subjacentes. Esses protocolos geram mensagens em código binário. Assim, são protocolos que devem utilizar mecanismos de segurança contra ataques situados nas mensagens em binário.

7. (F) Interfaces remotas e referências de objetos remotos caracterizam parcialmente o modelo de objetos distribuídos.

   Interfaces remotas e referências de objetos remotos caracterizam totalmente o modelo de objetos distribuídos.

8. A noção de referência de objeto é estendida para permitir qualquer objeto que possa receber uma invocação remota, tenha uma referência de objeto remoto. Essa referência é um identificador que pode ser usado por todo um sistema distribuído para se referir a um objeto único em particular. Sua representação contém um **endereço IP**, **número de porta**, **hora**, **número do objeto local** e **interface do objeto remoto**. Juntos, o **número de porta** e **hora,** produzem um identificador do objeto. O objeto remoto que vai receber uma invocação num método remoto, é especificado pelo invocador como uma referência de objeto remoto . As referências de objeto remoto podem ser passadas como argumentos e resultados de invocações a métodos remotos. Em Java RMI, um objeto-cliente precisa, para invocar um método remoto, conhecer uma referência de objeto remoto, a qual é usada no instante da invocação.

9. Uma interface de *Eleição* provê duas operações:

   vote: com dois parâmetros através dos quais o eleitor fornece o nome de um candidato (uma string) e o número do eleitor (um inteiro uado para garantir que cada cliente vota somente 1 vez). Os números dos eleitores são alocados, espaçadamente, dentro de um domínio de inteiros para torná-los difícil de adivinhar.

   result: com dois parâmetros através dos quais o servidor fornece ao apurador o nome de um candidato e o número de votos obtidos para aquele candidato.

   Discutir a semântica de invocação (indicar e explicar o porquê, qual a melhor semântica de invocação que pode ser utilizada quando o protocolo *request-reply* é **implementado em Java RMI**, sobre uma conexão TCP/IP, que garante que os dados são entregues na ordem enviada, sem perda ou duplicação, quando a conexão não falha. **Leve em consideração condições causando uma conexão falhar.**

   Escolha uma **Semântica de Invocação RMI**, apropriada para a aplicação *Eleição* acima:

   Semântica de Invocação Talvez: o método remoto pode ser executado uma vez ou não ser executado.

   Semântica de Invocação pelo menos uma vez: o invocador recebe um resultado quando o método remoto foi executado pelo menos uma vez, ou recebe uma exceção, informando-o que nenhum resultado foi obtido. Pode ser obtida pela retransmissão das mensagens de requisição.

   (Correção) **Semântica de Invocação no máximo uma vez**: o invocador recebe um resultado quando o método remoto foi executado exatamente uma vez, ou em caso contrário, uma exceção. Esta é o caso implementado em Java RMI.

10. (F) Um objeto *Factory*, usado no contexto de Java RMI serve para criar objetos localmente em objetos-clientes.
    (Correção) Um objeto *Factory*, usado no contexto de Java RMI serve para criar objetos remotamente em objetos-clientes.

    (F) Uma implementação de *Callback* em Java RMI, serve para fornecer respostas em tempo real, para objetos clientes que invocam respostas imediatas de alguma programação implementada em algum *objeto-servant* instanciado pelo objeto-servidor.
    (Correção) Uma implementação de *Callback* em Java RMI, serve para fornecer respostas a posteriori, para objetos clientes que invocam respostas de alguma programação implementada em algum *objeto-servant* instanciado pelo objeto-servidor. A resposta desta programação aparece tempos depois no objeto-cliente.