LETI/LEIC 2015-2016, Repescagem do 1º Teste de Sistemas Distribuídos

28 de junho de 2016

Responda no enunciado, usando apenas o espaço fornecido. Identifique todas as folhas.
Uma resposta errada numa escolha múltipla com N opções desconta 1/(N-1) do valor da pergunta.
Duração da prova: 1h30m

Grupo I [6,3]

Considere o seguinte programa em Java:

```
public class Car implements Serializable {
	private String make;
	private int mileage;
	// ...
}

public interface ICarManager extends Remote {

	// Looks for an available car that is parked within a specified radius of the
	// coordinates supplied as arguments;
	// if more than one car is available, the nearest is returned
	Car findNearestFreeCar(float latitudeCoordinates, float longitudeCoordinates,
		int maxRadiusMeters) throws RemoteException, NoAvailableCar;
...
```

1) Considere a interface acima, procure escrevê-la na IDL do SUN-RPC.
   a) [0,8] Em primeiro lugar, programe as estruturas de dados necessárias para os parâmetros dos RPC do SUN-RPC de modo a obter **exatamente o mesmo funcionamento** desta interface em Java.

```
struct                          struct
```

   b) [0,8] Complete agora a IDL Sun-RPC do serviço `findNearestFreeCar`. Inclua todos os elementos que achar necessários, propondo os que não conseguir obter diretamente da interface acima.

2) Considere que a semântica do RPC utilizada é "pelo-menos-uma-vez" (*at-least-once*). O cliente efetua a chamada a `findNearestFreeCar`. Indique no campo "Resultado para o cliente" se o RPC retorna um resultado válido ou inválido (`RPC_SUCCESS` ou `RPC_ERROR`, respetivamente) e indique no segundo campo o número de vezes que a função é executada no servidor.

a) [0,5] A mensagem de invocação inicial é perdida

Resultado para o cliente	Número de vezes executada no servidor

b) [0,5] A mensagem de resposta do servidor é perdida pela rede 2 vezes

Resultado para o cliente	Número de vezes executada no servidor

c) [0,5] As mensagens de resposta do servidor são perdidas pela rede excedendo o *timeout* do cliente. O *timeout* do cliente é 1s e a repetição demora 100 ms.

Resultado para o cliente	Número de vezes executada no servidor

3) [0,8] Se a semântica fosse "no-máximo-uma-vez" (*at-most-once*) qual (ou quais) dos três quadros anteriores seria diferente? Identifique claramente qual ou quais e justifique.

4) Considere que o RPC tinha uma política de "receptor-converte" (*receiver makes it right*) que não é a do Sun-RPC. Suponha que o servidor tem uma arquitetura de dados TIPO I e tem um cliente A com a mesma arquitetura (TIPO I) e um cliente B com uma arquitetura diferente TIPO II.

a) [0,5] Represente o que teria de enviar numa invocação do serviço `findNearestFreeCar` quando o cliente A invoca o serviço (arbitre os valores dos parâmetros).

b) [0,5] Represente o mesmo quando o cliente B invoca o serviço (arbitre os valores dos parâmetros).

c) [0,6] Quais seriam as principais diferenças se a invocação fosse em XDR? Justifique

d) [0,8] A utilização de XDR teria algum impacto sobre o desempenho do RPC? Justifique.

Grupo II [6,7]

Considere novamente **o programa em Java do Grupo I**. Mantendo o serviço descrito pretende-se agora que o sistema use RMI e as funcionalidades deste sistema de objetos distribuídos.

1) [0,8] Defina a interface `ICar` com os seguintes requisitos:
   - Os objetos da classe correspondente ficam residentes nos sistemas informáticos dos automóveis, mantendo-se ativos e enviando periodicamente a localização GPS do veículo.
   - O objeto tem, entre outros, um método:

     ```
     CarReservation reserve(int userId) throws CarNotAvailable
     ```

   Este método deverá poder ser executado remotamente pelo cliente para reservar um carro. O parâmetro de resposta é uma reserva (ignore, por agora, a informação que a compõe) para reservar um carro durante 1 hora que, quando apresentado ao sistema do carro, o permite usar.

```
public interface ICar
```

2) [0,8] Modifique agora a interface que centraliza a informação sobre os carros: `ICarManager`.

As principais diferenças a incluir são:
- O método `findNearestFreeCar` deve retornar o objeto que representa o carro com o tipo programado na alínea anterior
- Deve haver um método para o carro se registar no sistema indicando os elementos fabricante e quilometragem.
- Ignore outros métodos que porventura seriam necessários para o sistema ter funcionalidade completa

3) Suponha que o `CarReservation` deve conter `int pinCode, DigSign pinDigSign`
   a) [0,6] Na lógica desta aplicação, este objeto deveria ser passado por valor ou por referência? Justifique.

   b) [0,6] Que diferença existe a nível da programação para especificar a decisão da alínea anterior?

   c) [0,8] Em qualquer das situações o cliente necessita da classe do objeto para o poder utilizar. Como resolve o Java RMI esta situação quando o cliente ainda não possui a classe? Justifique.

4) O cliente do sistema obtém de acordo com as alíneas anteriores uma referência remota para um objeto, podemos supor neste caso o `car1234`.
   a) [0,6] Comente a afirmação: "Para obter esta referência remota o objeto car1234 tem de registar-se no *RMI Registry* de outra forma não será possível encontrar a sua referência remota".

   b) [0,5] **Quando** é criada a referência remota inicial para este objeto? Justifique.

   c) [0,5] **Quem** é responsável por criá-la? Justifique.

   d) [0,5] **Quando** o cliente recebe a referência remota para um veículo **quem lha transmite?**

e) Suponha que o sistema implementa um *garbage collector* baseado na contagem de referências. Na situação descrita nesta alínea procure explicar:

i) [0,4] Que valor terá o contador de referências na máquina virtual onde se executa o objeto `car1234`?

ii) [0,6] Justifique a sua resposta detalhando os passos que permitem calcular esse valor.

Grupo III [7]

Considere um sistema de encomendas eletrónicas baseado na tecnologia de **Web Services**.

1) O Servidor recebe pedidos de encomenda através da operação `placeOrder`.
Capturou-se na rede a seguinte mensagem SOAP a caminho do Servidor:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:placeOrder xmlns="http://tempuri.org/PurchaseOrderSchema.xsd"
xmlns:ns2="http://ws.example/">
            <ns2:order OrderDate="2016-06-12+01:00">
                <ShipTo>
                    <name>Warehouse</name>
                    <street>22nd st</street>
                    <city>Springfield</city>
                    <state>MA</state>
                    <zip>1105</zip>
                </ShipTo>
                <BillTo>
                    <name>Headquarters</name>
                    <street>Vassar St</street>
                    <city>Cambridge</city>
                    <state>MA</state>
                    <zip>2139</zip>
                </BillTo>
            </ns2:order>
        </ns2:placeOrder>
    </S:Body>
</S:Envelope>
```

a) [0,5] Consegue inferir a linguagem de programação em que o Servidor está programado? Justifique.

b) [0,4] O que define o valor "`http://tempuri.org/PurchaseOrderSchema.xsd`" na mensagem?

i) O endereço de destino da mensagem SOAP.

ii) A localização do *schema*.

iii) O espaço de nomes dos elementos `ShipTo` e `BillTo`.

iv) O espaço de nomes do elemento `order`.

c) [0,4] Qual é a representação do valor "`2139`" que passa na rede?
   i) São os valores binários dos caracteres '2' '1' '3' '9'
   `00110010 00110001 00110011 00111001`
   ii) É 2139 em binário, inteiro de 32 bits, *little endian*
   `11011010 00010000 00000000 00000000`
   iii) É 2139 em binário, inteiro de 32 bits, *big endian*
   `00000000 00000000 00001000 01011011`
   iv) Não é nenhuma das anteriores.

d) [0,5] A data "`2016-06-12+01:00`" indicada na mensagem corresponde ao dia 12 de Junho de 2016. Como foi resolvida a ambiguidade com o dia 6 de Dezembro de 2016?

2) Considere agora que o Cliente e o Servidor foram desenvolvidos em **Java** com a biblioteca JAX-WS.
   a) [0,3] Qual é a ferramenta responsável pela geração de código para invocação do Web Service?

   b) [0,4] Qual é o argumento principal que deve fornecer à ferramenta de geração? Exemplifique.

   c) Escreva em pseudo-código as classes Java geradas pela ferramenta para transporte de dados:
      i) [0,6] Classe que representa a Encomenda (`order`).

      ii) [0,6] Classe que representa a Morada. Pode assumir que `BillTo` e `ShipTo` são do mesmo tipo.

3) Entretanto obteve acesso ao contrato do serviço que descreve a operação `placeOrder` e encontrou a seguinte informação: `<fault message="tns:orderFault" name="orderFault" />`

   a) [0,7] Como representaria o método Java correspondente à operação `placeOrder` na interface gerada, tendo em conta esta definição? Assuma que a operação não retorna resultados.

```
void placeOrder(
```

   b) [0,5] Dê exemplo de uma situação concreta em que faça sentido o Servidor devolver uma `orderFault`?

4) Considere os seguintes nomes de etiquetas XML usadas por normas de Web Services.

   a) [0,5] Assinale com ● os elementos abaixo que são usados num documento WSDL.

   `handlerChain`

   `message`

   `header`

   `types`

   `portType`

   `service`

   `envelope`

   `binding`

   b) [0,5] Que elementos escolhidos definem a interface abstrata do serviço no WSDL?

   c) [0,6] Porque é necessária uma interface concreta no WSDL além da interface abstrata?

   d) [0,5] Indique um item de informação que faça parte apenas da interface concreta do serviço no WSDL e explique sucintamente a utilidade dessa informação.