LEIC/LETI – 2014/15, 1º Teste de Sistemas Distribuídos, 27 de Março de 2015

Responda no enunciado, apenas no espaço fornecido. Identifique todas as folhas. Duração: 1h30m

Grupo I [7v]

Considere um servidor que implementa o jogo do galo distribuído em SUN RPC, oferecendo a seguinte interface remota:

```
struct play_args {
    int row;
    int column;
    int player;
};

program TTT {
    version V1 {
        string CURRENTBOARD(void)=1;
        int PLAY(play_args)=2;
        int CHECKWINNER(void)=3;
    } = 1;
} = 0x31423456;
```

1. [0,7v] Depois de executado o *rpcgen*, o ficheiro *ttt.h* contém o seguinte extrato:

```
#define CURRENTBOARD 1
extern  char ** currentboard_1(void *, CLIENT *);
extern  char ** currentboard_1_svc(void *, struct svc_req *);
#define PLAY 2
extern  int * play_1(play_args *, CLIENT *);
extern  int * play_1_svc(play_args *, struct svc_req *);
#define CHECKWINNER 3
extern  int * checkwinner_1(void *, CLIENT *);
extern  int * checkwinner_1_svc(void *, struct svc_req *);
```

Entre as seis funções definidas acima no ficheiro *.h*, indique:

a) Quais se tratam dos *stubs* do lado do cliente.

b) Quais deverão ser implementadas pelo programador.

2. [1,4v] Programe de seguida um cliente que: i. estabelece sessão por UDP com um servidor a correr na máquina sigma.tecnico.ulisboa.pt; ii. chama a função remota "*checkwinner*" e imprime o resultado no ecrã. Caso ocorra falha originada no RPC, o programa cliente deve imprimir a mensagem "erro de RPC".

Caso necessite de chamar a função *clnt_create*, apresenta-se de seguida a respetiva descrição das *man pages*:

> *CLIENT *clnt_create(char *host, unsigned long prog,unsigned long vers, char *proto);*
> *Generic client creation routine. host identifies the name of the remote host where the server is located. proto indicates which kind of transport protocol to use. The currently supported values for this field are "udp" and "tcp".*

```
int main (int argc, char *argv[])
{









}
```

3. Considere a seguinte implementação da função play no servidor:

```
int *play_1_svc(play_args *argp, struct svc_req *rqstp)
{
  static int  result;
  pthread_mutex_lock(&mutex);
  board[argp->row][argp->column] = (argp->player == 1) ? 'X' : 'O';
  nextPlayer = (nextPlayer + 1) % 2;
  numPlays ++;
  pthread_mutex_unlock(&mutex);
  result=0;
  return &result;
}
```

a) [0,7v] Porque razão é usado um trinco lógico (pthread_mutex_lock e pthread_mutex_unlock) na implementação desta função?

b) [0,8v] A função remota *play* é idempotente? Justifique ilustrando com excertos do código acima.

c) O argumento do tipo *play_args* é enviado na mensagem de pedido que é enviada pelo cliente.

i) [0,5v] É passado por valor ou por referência?

ii) [0,8v] "O XDR, protocolo de apresentação do SUN RPC, usa uma política de conversão canónica dos parâmetros, o que prejudica o desempenho quando cliente e servidor são homogéneos." Concorda ou discorda? Justifique.

4. Considere o seguinte extrato de um programa cliente que invoca uma função remota f usando um determinado RPC (não necessariamente SUN RPC).

```
result = f_4(&arg, clnt);
```

O retorno será diferente de NULL caso o cliente tenha recebido resposta do servidor; será NULL caso o *run-time* do RPC do lado cliente detete que houve um erro na chamada remota.

Nas alíneas seguintes, para cada semântica de invocação, indique quantas vezes a função f se executa no servidor na sequência da linha de código cliente acima, consoante o retorno recebido pelo programa cliente. Exemplo: "pode ter sido executado 1 ou 2 vezes".

Assuma que o canal de comunicação pode perder mensagens e que o servidor pode falhar silenciosamente, mas nunca durante a execução de uma função remota.

Caso exista mais que uma resposta possível para determinada alínea, indique todas as respostas possíveis.

a) [0,7v] Semântica pelo-menos-1-vez:

i. Programa cliente recebe result != NULL.

ii. Programa cliente recebe result == NULL.

Justifique:

b) [0,7v] Semântica no-max-1-vez:

i. Programa cliente recebe result != NULL.

ii. Programa cliente recebe result == NULL.

Justifique:

c) [0,7v] Semântica exatamente-1-vez:

i. Programa cliente recebe result != NULL.

ii. Programa cliente recebe result == NULL.

Justifique:

Grupo II [6v]

Considere a aplicação móvel de um serviço de *car-sharing*, baseada em Java RMI. Essa aplicação permite aos utilizadores do serviço consultarem e reservarem automóveis disponíveis numa determinada zona da cidade. Considere os seguintes excertos de duas interfaces Java deste sistema.

```
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 one
       //is returned
       ICar findNearestFreeCar(float LattitudeCoordinates, float LongitudeCoordinates, int
maxRadiusMeters)
              throws RemoteException, NoAvailableCar;
       …
}

public interface ICar extends Remote {
       //Reserves the car for the specified user; in case of success, the car becomes
       //locked for other users for 30 minutes, and can only be opened by the user holding
       //the credentials in the returned CarTicket
       CarTicket reserve(int userId)
              throws RemoteException, CarNotAvailable;
       …
}
```

Existe uma instância de ICarManager a correr num servidor e registada num RMI Registry com o nome "//rmi.carsharing.pt/carSharingManager".

1. [1,4v] Programe um método Java que tenta reservar o carro mais próximo da zona definida nos argumentos. O método deve retornar um objeto do tipo CarTicket caso a reserva tenha tido sucesso ou null caso contrário. Na sua resposta, pode omitir a configuração do *SecurityManager*. Não se esqueça de tratar eventuais erros da invocação remota.

```
public CarTicket reserveCar
(float LattitudeCoordinates, float LongitudeCoordinates, int maxRadiusMetersfloat , int userId) {




}
```

2. Indique, quando executa a reserva de um carro com a sua solução da alínea anterior:
   a) [0,7v] Quantas referências remotas tem o cliente? Justifique.

   b) [0,7v] Quantas classes proxy existem carregadas no cliente? Justifique.

   c) [0,8v] Sabendo que o Java RMI usa o método de contagem de referências para gerir a recolha automática de memória (*garbage collection*) dos objetos remotos, indique quantas vezes as operações addRef e removeRef são chamadas na execução da reserva com o seu programa. Justifique.

3. O tipo CarTicket inclui os campos: *int pinCode; string streetAddress; int carId.* Deve ser passado por valor.

a) [0,8v] Apresente em Java a definição da classe CarTicket. Omita métodos da classe.

b) [0,8v] O que acontecerá caso o programa cliente, que chama o método reserveCar da alínea 1, não tenha a classe CarTicket disponível na sua máquina virtual Java?

4. [0,8v] "Em Java RMI, qualquer objeto remoto tem de ter estar registado num *RMI Registry* para ser referenciado por outros clientes." A frase é verdadeira ou falsa? Justifique ilustrando com o sistema considerado acima.

Grupo III [7v]

1) Considere os seguintes extratos de um documento WSDL

A

```
<wsdl:message name="createUser">
    <wsdl:part name="parameters" element="tns:createUser"/>
</wsdl:message>
<wsdl:message name="createUserResponse" />
<wsdl:message name="UserAlreadyExists">
    <wsdl:part name="fault" element="tns:UsernameProblem" />
</wsdl:message>
```

B

```
<xs:complexType name="createUser">
   <xs:sequence>
       <xs:element name="emailAddress" type="xs:string"/>
       <xs:element name="userId" type="xs:string"/>
    </xs:sequence>
</xs:complexType>

<xs:element name="UsernameProblem" type="tns:UsernameProblem" />
<xs:complexType name="UsernameProblem">
    <xs:sequence>
        <xs:element name="message" type="xs:string" minOccurs="0" />
        <xs:element name="userId" type="xs:string" minOccurs="0" />
     </xs:sequence>
</xs:complexType>
```

C

```
<wsdl:operation name="createUser">
   <wsdl:input message="tns:createUser" name="createUser"/>
   <wsdl:output message="tns:createUserResponse" name="createUserResponse"/>
   <wsdl:fault message="tns:InvalidEmail" name="InvalidEmail" />
   <wsdl:fault message="tns:EmailAlreadyExists" name="EmailAlreadyExists" />
   <wsdl:fault message="tns:UserAlreadyExists" name="UserAlreadyExists" />
</wsdl:operation>
```

a) [0,6] Indique a que secção de um documento WSDL corresponde cada excerto.

A	
B	
C	

b) A partir do excerto pode identificar uma operação executada por este serviço.

i. [0,2] Indique o nome da operação.

ii. [0,2] Indique os parâmetros de entrada.

iii. [0,2] Indique as exceções.

iv. [0,7] Escreva o protótipo desta operação em Java sem os elementos específicos de XML presentes nos excertos apresentados.

c) [0,7] Suponha que evoca a operação numa situação em que já existe registado um utilizador com o mesmo userID por exemplo IST12000. Descreva o conteúdo do envelope do pacote SOAP de resposta.

d) No documento aparece a tag *tns* definida como xmlns: tns = "urn:pt:sdid:ws".

i. [0,5] Qual a função deste tipo de tags em documentos XML? Justifique explicando o excerto seguinte:

```
<wsdl:message name="createUser">
  <wsdl:part name="parameters" element="tns:createUser"/>
</wsdl:message>
```

ii. [0,5] Seria possível ter xmlns:tns = "http://disciplinas.tecnico.ulisboa.pt/leic-sod/2014-2015/labs/"? Justifique.

e) [0,5] Considere a invocação da operação descrita acima, das secções reproduzidas posso assumir que o protocolo de transporte utilizado é HTTP? Justifique.

f) [0,5] Num projeto desenvolvido no modelo *implementation-first* existe alguma forma de poder definir múltiplos protocolos de transporte para invocar a operação? Ou fico limitado a HTTP e HTTPS? Justifique.

2. Considere o seguinte programa cliente:

```
1.  String endpointAddress = uddiNaming.lookup(name);
2.  if (endpointAddress == null) {System.out.println("Not found!"); return;}
3.  HelloImplService service = new HelloImplService();
4.  Hello port = service.getHelloImplPort();
5.  BindingProvider bindingProvider = (BindingProvider) port;
6.  Map<String, Object> requestContext = bindingProvider.getRequestContext();
7.  requestContext.put(ENDPOINT_ADDRESS_PROPERTY, endpointAddress);
```

a) [0,6] O uso inicial do UDDI na linha 1 é para o cliente poder aceder ao WSDL que necessita para executar o proxy. Concorda ou discorda? Justifique.

b) [0,6] Neste excerto, o programa usa o URL definido no WSDL para localizar o serviço. Concorda ou discorda? Justifique com base no programa indicando as respetivas linhas.

c) [0,6] O objeto proxy para contactar o serviço é instanciado automaticamente ou necessita de uma ação de criação específica? Justifique com base no programa indicando as respetivas linhas.

d) [0,6] Na linha 4 o JAX-WS vai buscar o código do proxy ao servidor onde este estiver armazenado e carrega a respetiva classe dinamicamente ligando-a ao código do cliente. Concorda ou discorda? Justifique.
