FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE NOVA DE LISBOA

Departamento de Informática

Licenciatura / Mestrado Integrado em Engenharia Informática
Sistemas Distribuídos – 1º teste, 14 de Abril de 2014
1º Semestre, 2013/2014

NOTAS: Leia as questões atentamente antes de responder. **O teste é sem consulta. A duração do teste é 1h30min**. O teste contém **4** páginas.

Nome: _____ **Número**: _____

1) Considere o contexto do trabalho prático, em que um conjunto de servidores fornecem acesso a ficheiro através de RMI e Web Services (SOAP). Complete o código do anexo A, um excerto do código do sistema com o código adequado face aos comentários no código.
2) Indique se cada afirmação é **[V]erdadeira** ou **[F]alsa** (**nota: respostas incorretas descontam**):

___ Num sistema distribuído é possível garantir que os relógios de várias máquinas não diferem mais do que um valor tão pequeno quanto se pretender.

___ Um sistema fechado (em oposição a um sistema aberto) permite uma mais fácil integração de novas funcionalidades.

___ O UDP implementa um modelo de comunicação bi-direcional.

___ O TCP implementa um modelo de comunicação síncrona.

___ Um sistema de *message queuing* implementa um modelo de comunicação volátil.

___ Usar particionamento dos dados nos servidores é uma forma de aumentar a escalabilidade dum sistema cliente/servidor.

___ Num modelo cliente/servidor replicado, a falha dum servidor deve ser tratada de forma cuidadosa para garantir que o sistema não fica inconsistente.

___ Numa rede peer-to-peer não estruturada é impossível encontrar a informação pretendida de forma eficiente.

___ Numa rede peer-to-peer estruturada (DHT) é impossível replicar dados em mais do que um nó de forma a que os mesmos sejam pesquisáveis.

___ No BitTorrent, o facto de se saber o hash de todos os blocos é fundamental para que o sistema funcione corretamente.

___ A única vantagem dum proxy próximo do cliente é melhorar a latência da resposta nos clientes.

___ Um sistema que falha raramente pode ter baixa fiabilidade.

___ A segurança dum sistema distribuído está apenas relacionada com manter a integridade e confidencialidade dos dados manipulados pelo sistema.

No contexto da invocação remota de métodos/procedimentos:

___ Enviar mensagens codificadas no formato do emissor é tipicamente mais eficiente do que enviar mensagens num formato intermédio.

___ O Java RMI implementa uma semântica de invocação "at least once".

___ No Java RMI os tipos primitivos são passados por valor, mas os objetos são passados por referências.

___ Nos web services (SOAP) os tipos primitivos e os objetos são passados por valor.

___ O servidor *rmiregistry* a executar na máquina X só pode registar servidores a executar na máquina X.

___ Nos web services, é possível criar uma referência remota que permita invocar um método dum servidor a correr em qualquer máquina (desde que se conheça o seu endereço).

3) Considere que pretende implementar um sistema semelhante ao DropBox e tem à sua disposição três centros de dados, um em cada um dos seguintes continentes: Europa, Ásia, América.

   a) Proponha uma arquitetura para implementar este serviço. Justifique a sua opção, indicando como distribuiria os ficheiros de cada utilizador pelos três centros de dados e quais as vantagens da sua solução face a outras soluções.

   b) Seria interessante estender a sua arquitetura usando servidores presentes nos ISPs (e.g. rede Akamai)? Justifique explicando a razão da sua resposta e indicando o que assume sobre a utilização do sistema por parte dos utilizadores.

Sim/ Não , porque…

4) Compare o modelo de transmissão dos dados usado no Java RMI e nos web services SOAP.

5) Apresente um protocolo de invocação remota com semântica exatamente uma vez. Explique porque funciona em caso de falhas, enumerando os casos para as diferentes falhas que podem ocorrer. NOTA: deve indicar claramente o que faz o cliente e o servidor.

ANNEX A

```
/** Exceção indicando que ficheiro/directório não existe */
public class NotFoundException extends Exception { ... }

/** Interface do servidor RMI */
public interface IFileServer extends                    {
/** Devolve um array de bytes com o conteúdo do ficheiro. Em cas de erro lança NotFoundException */
   public byte[] getFileContents( String path) throws NotFoundException,
}
/** Classe do servidor RMI */
public class FileServer extends                  implements                  { ...
   public void main(String[] args) {
      try { ...
         FileServer server = new FileServer();
```

```
      } catch(Exception th) { ... }
} }
/** Class for WS server */
```

```
public class FileServerWS {
   /** Escreve no ficheiro parh o conteúdo de contents. Em caso de erro lança NotFoundException */
```

```
   public void putFileContents( String path, byte[] contents) throws
   {...}
   public void main( String[] args) {
```

```
} }
/** Classe principal do cliente */
class FileClient {
   public IFileServer getRMIRef ( String url) { ... }  // Devolve referência para servidor RMI
   public ws.FileServerWS getWSRef ( String url) { ... }  // Devolve referência para servidor web service
   /** Copia o ficheiro frompath do servidor RMI fromURL para o servidor de web services
    * toURL, com a path toPath. Devolve false em cado de erro no acesso a algum dos servidores */
   protected boolean cp( String fromUrl, String frompath, String toURL, String toPath) {
```

```
} }
```