UNIVERSIDAD AUTONOMA DE MADRID

Asignatura SISTEMAS INFORMÁTICOS II Grupo 236 y 240

Apellidos ..... Nombre .....

Ejercicio del día 15 de marzo de 2018. Examen parcial.

Teoría 1 (1)	Teoría 2 (1)	Teoría 3 (1)	Teoría 4 (1)	Teoría 5 (1)	Total Teoría (10)

1.- TEORÍA (10 puntos). Contesta de modo claro y conciso a las siguientes cuestiones.

1. Nombra la secuencia de llamadas a realizar en el servidor para establecer una conexión TCP usando la API de los sockets.

2. Indica para qué se usa el stub del cliente en las RPCs.

3. Indica la utilidad de portmapper en la implementación de RPC de SUN.

4. Nombra los protocolos y componentes del estándar de directorios X.500.

5. Indica las ventajas y desventajas de usar modelos matemáticos simples frente a un “experto” para estimar el rendimiento en un sistema distribuido.

UNIVERSIDAD AUTONOMA DE MADRID

Asignatura SISTEMAS INFORMÁTICOS II Grupo 236 y 240

Apellidos Nombre

Ejercicio del día 15 de marzo de 2018. Examen parcial.

Teoría 6 (1)	Teoría 7 (1)	Teoría 8 (1)	Teoría 9 (1)	Teoría 10 (1)

6. Nombra las componentes de CORBA en la parte del cliente.

7. Indica en qué consiste la transparencia de acceso.

8. Nombra las ventajas de un bus de servicios de empresa “Enterprise Service Bus” (ESB).

9. Indicar la función del Service Broker como componente de los Web Services basados en SOAP.

10. Explicar la función del Internet Inter ORB Protocol (IIOP) en CORBA.

UNIVERSIDAD AUTONOMA DE MADRID

Asignatura SISTEMAS INFORMÁTICOS II Grupo 236 y 240

Apellidos ..... Nombre .....

Ejercicio del día 15 de marzo de 2018. Examen parcial.

2.1 (5)	3.1 (2)	4.1 (3)	Total Problemas (10)

2. PROBLEMA. Considerar los siguientes casos particulares de sistemas distribuidos:

1. Se quiere construir una aplicación distribuida que permita compartir ficheros a través de la red. Utilizando esta aplicación, tras autentificarse, los clientes se conectarán al servidor y podrán ver qué usuarios hay conectados, los ficheros que comparten, y podrán descargar los mismos. También se podrá crear grupos de usuarios y se podrá especificar con qué usuarios en particular se quiere compartir los ficheros exportados, pudiendo bloquear a ciertos usuarios. La aplicación se usará a través de internet, y no se espera que el tráfico sea filtrado. Sin embargo, se desearía que el tamaño de los mensajes intercambiados entre clientes y servidores no fuese muy grande. Los clientes serán heterogéneos, aunque se ha establecido que se usará un único lenguaje de programación para la codificación de la aplicación. Por sus ventajas se ha decido diseñar la aplicación usando técnicas de orientación a objetos. También se desearía que la complejidad del proceso de desarrollo de la aplicación permaneciese lo más sencillo posible.

2. Una empresa quiere prestar un servicio de envío de video en tiempo real. Los usuarios que acceden al servicio de forma simultánea se encuentran conectados a la misma red local que el servidor. Se desearía que el envío de mensajes fuese lo más eficiente posible, para permitir el envío de vídeo de alta definición, pues la capacidad de la red local es limitada. Los clientes que accedan al servicio usarán la misma arquitectura que el servidor.

2.1. (4 puntos) Para cada uno de ellos se pide elegir razonadamente el mecanismo de comunicación más adecuado entre los vistos en la parte de teoría de la asignatura (Java RMI, CORBA, WS-soap, WS-rest, RPC, UDP, TCP o Colas de Mensajes). Indicar así mismo si será necesario implementar algún **mecanismo adicional de traducción de datos**.

No se tendrán en cuenta respuestas sin justificación (cuantas más justificaciones, mayor la puntuación).

1. Hay que implementar funcionalidades síncronas, por lo que descartaríamos las colas de mensajes. Este mecanismo también quedaría descartado, ya que los clientes son heterogéneos, y no incorpora nada para la traducción de datos. Descartaríamos UDP y TCP ya que la funcionalidad del sistema es bastante complicada y los sockets son de muy bajo nivel. Como los mensajes han de ser pequeños, descartamos WS-soap. Hemos de usar técnicas orientadas a objetos, por lo que descartaríamos RPCs frente a JAVA RMI o CORBA. Los datos no serán filtrados, por lo que WS-rest no es tan atractivo frente a estos mecanismos. Como hemos de usar un único lenguaje de programación y el proceso de desarrollo ha de ser sencillo, elegiríamos Java RMI y descartaríamos CORBA. No habría que añadir nada para traducir datos, ya que Java RMI se encargaría de ello.

2. Hay que implementar funcionalidades síncronas, por lo que descartaríamos las colas de mensajes. No será necesario implementar ningún mecanismo adicional de traducción ya que los clientes y el servidor tienen la misma arquitectura. La funcionalidad es sencilla (solo hay

UNIVERSIDAD AUTONOMA DE MADRID

Asignatura SISTEMAS INFORMÁTICOS II Grupo 236 y 240

Apellidos Nombre

Ejercicio del día 15 de marzo de 2018. Examen parcial.

una) y deber de garantizarse el envío eficiente de los mensajes. Esto descarta WS-soap y en general todos los mecanismos que son de alto nivel como RPC, Java RMI o CORBA pues son muy complicados para implementar algo tan sencillo. Por su eficiencia y simplicidad elegiríamos protocolos de bajo nivel como TCP o UDP, y entre ellos nos quedaríamos con UDP, pues es aún más eficiente que TCP (si se pierde algún mensaje no tiene sentido volverlo a pedir, ya que es video en tiempo real) y además UDP permite direccionamiento multi-cast que permite llegar a múltiples destinatarios poniendo un único mensaje en la red.

3. PROBLEMA Un servidor B recibe un mensaje de otro servidor A, a las 10:32:11.55 (las dos últimas cifras son centésimas de segundo). El mensaje lleva una marca de tiempo igual a las 10:32:05.32. Tras esto el servidor A recibe un mensaje del servidor B a las 10:32:11.92. La marca de tiempo del mensaje es igual a las 10:32:16.85.

3.1. (2 puntos) Determina la deriva entre los relojes de ambos servidores e indica la precisión de la estimación realizada.

T_i-3=10:32:05.32
T_i-2=10:32:11.55
T_i-1=10:32:16.85
T_i=10:32:11.92

o_i = ((T_i-2 – T_i-3) – (T_i – T_i-1)) / 2 = ((11.55 – 05.32) – (11.92 - 16.85)) / 2 = (6.23 + 4.93) / 2 = = 5.58 segundos

d_i / 2 = ((T_i-2 – T_i-3) + (T_i – T_i-1)) / 2 = ((11.55 – 05.32) + (11.92 – 16.85)) / 2 = (6.23 – 4.93) / 2 = 1.3 / 2 = 0.65 segundos.

La deriva entre los relojes es de 5.85+-0.65 segundos.

4. PROBLEMA. Considerar el siguiente fichero WSDL de un servicio web SOAP:

```
<?xml version="1.0"?>
<definitions name="MyService" targetNamespace=http://www.examples.com/wsdl/MyService.wsdl
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap=”http://schemas.xmlsoap.org/wsdl/soap/”
xmlns:tns=”http://www.examples.com/wsdl/MyService.wsdl”
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<message name="info1"><part name="login" type="xsd:string"/></message>
<message name="info2"><part name="email" type="xsd:string"/></message>

<message name="userUpdate1"><part name="login" type="xsd:string"/>
<part name="email" type="xsd:string"/></message>
<message name="userUpdate2"><part name="status" type="xsd:integer"/></message>
```

UNIVERSIDAD AUTONOMA DE MADRID

Asignatura SISTEMAS INFORMÁTICOS II Grupo 236 y 240

Apellidos Nombre

Ejercicio del día 15 de marzo de 2018. Examen parcial.


```
<portType name="Service_PortType"><operation name="getInfo">
<input message="tns:info1"/><output message="tns:info2"/>
</operation><operation name="updateUser">
<input message="tns:userUpdate1"/><output message="tns:userUpdate2"/>
</operation></portType>

<binding name="Service_Binding" type="tns:Service_PortType">
<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
<operation name="getInfo"><soap:operation soapAction="getInfo"/>
<input><soap:body encoding="literal"/></input><output>
<soap:body encoding="literal"/></output></operation>
<operation name="updateUser"><soap:operation soapAction="updateUser"/>
<input><soap:body encoding="literal"/></input><output>
<soap:body encoding="literal"/></output></operation>
</binding>

<service name="My_Service"><port binding="tns:Service_Binding"
name="Service_Port"><soap:address location="http://www.si2.com/MiSitioWeb/" />
</port> </service></definitions>
```

4.1 (1 punto) Indica las operaciones soportadas por el servcio web.

Son dos operaciones: getInfo y updateUser.

4.2 (1.5 puntos) Indica qué mensajes se enviará para cada operación y el contenido de dichos mensajes.

La operación getInfo recibe un mensaje de petición llamado info1 que contiene un string llamado login. Devuelve un mensaje llamado info2 que contiene otro string llamado email.

La operación userUpdate recibe un mensaje de petición llamado userUpdate1 que contiene un string llamado login y otro string llamado email. Devuelve un mensaje llamado userUpdate2 que contiene un entero llamado status.

4.3 (0.5 puntos) Indica cual es la URL que habrá que utilizar para acceder al servicio.

http://www.si2.com/MiSitioWeb