UNIVERSIDAD CARLOS III DE MADRID
AREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES
GRADO EN INGENIERÍA INFORMÁTICA. SISTEMAS DISTRIBUIDOS

Para la realización del presente examen se dispondrá de **2 horas y 30 minutos. NO** se podrán utilizar libros, apuntes **ni** calculadoras de ningún tipo. **Responda** a los ejercicios en el **espacio reservado**.

Alumno: _____ **Grupo:** _____

Pregunta 1 (2 puntos): Un cliente envía una petición de longitud L bytes a un servidor con un único procesador. El servidor procesa la petición en un tiempo t y devuelve una respuesta de longitud L/2 bytes. El tiempo t de procesamiento de una petición sólo supone tiempo de CPU. La red dispone de un ancho de banda de 1Mbit/s. Responda razonadamente a las siguientes preguntas:

a) ¿Cuál es el tiempo de respuesta percibido por el cliente?
b) Si el servidor es secuencial, ¿cuántas peticiones como máximo podría atender en el intervalo de tiempo [1,N]?
c) ¿Y si el servidor es concurrente?
d) Asuma que el servidor dispone de 4 procesadores, ¿cuál sería el tiempo percibido por el cliente?
e) Ventajas e inconvenientes de un servidor secuencial frente a un servidor concurrente.

Pregunta 2 (1 punto): Responda a las siguientes cuestiones:

a) En un sistema basado en RPC ¿qué quiere decir que el *binding* es persistente? Ventajas e inconvenientes de un *binding* persistente frente a uno no persistente.
b) ¿En qué consiste el paradigma de computación peer-to-peer? Indique tres ejemplos de aplicaciones basadas en este paradigma.

Pregunta 3 (1,5 puntos): Se disponen de 3 clientes A, B y C que usan tres sistemas de ficheros distribuidos SF1, SF2 y SF3, cada uno de ellos empleando un modelo de acceso diferente. Un cliente A usa el SF1 que emplea el modelo de carga/descarga, un cliente B usa el SF2 que emplea acceso remoto (sin caché en los clientes) y un cliente C que usa el SF3 que emplea un modelo híbrido. Los clientes A, B y C quieren acceder para lectura a un único bloque del fichero 'foo.txt'. Complete la siguiente tabla:

	Cliente A	Cliente B	Cliente C
Servicios del SFdel cliente que implican comunicación con S			
Unidad de transferencia C→S			
Dónde se realiza la operación			
Unidad de transferencia S→C			
Número de mensajes de petición/respuesta			

Notación: S: Servidor/C: Cliente

Pregunta 4 (1,5 puntos): Considere un sistema distribuido compuesto por N=5 nodos que mantienen réplicas de acuerdo al algoritmo de votación dinámica (quórum). Inicialmente:

	Nodo 1	Nodo 2	Nodo 3	Nodo 4	Nodo 5
SC	5	5	5	5	5
NV	8	8	8	8	8

La red se fragmenta en dos particiones, una partición 1 compuesta por {1,5} y una partición 2 compuesta por {2,3,4}. Se pide representar el valor de SC y NV después de cada uno de los siguientes eventos:

a) Actualización en la partición 2.

	Nodo 1	Nodo 2	Nodo 3	Nodo 4	Nodo 5
SC					
NV					

b) La partición 2 se fragmenta en dos particiones más, formadas por los nodos {2} y {3,4}. Actualización en la partición {3,4}

	Nodo 1	Nodo 2	Nodo 3	Nodo 4	Nodo 5
SC					
NV					

Pregunta 5 (4 puntos): Se desea diseñar un servicio de *broadcast* en el que un proceso envía un mensaje a todos los N procesos de un sistema distribuido excepto a sí mismo. En este sistema, el identificador de los procesos oscila entre 0 y N-1 y es conocido por todos los procesos. Los procesos ejecutan en la misma máquina pero en puertos distintos. Se asumirá que la dirección IP de la máquina es 168.222.12.12 y que los procesos usan puertos consecutivos a partir del puerto 10000 (el proceso con ID igual a 0 usa el puerto 10000, el proceso con ID igual a 1 usa el 10001, etc.).

El servicio debe proporcionar cierta forma de fiabilidad, por lo que se dispone de un servicio, denominado `timeout`, que permite gestionar un temporizador de retransmisiones. El servicio `timeout` bloquea al proceso t unidades de tiempo o hasta que un mensaje es recibido. Devuelve un 1 si el temporizador ha expirado y 0 en caso contrario. Su prototipo es el siguiente:

- `int timeout(int t);`

El servicio de `broadcast` tiene el siguiente prototipo:

- `int broadcast(int N, char *msg,int t);`

donde `N` es el número de procesos a los que se envía el mensaje, `msg` es el mensaje a enviar y `t` es el tiempo de espera (en segundos).

Se dispone además de la siguiente interfaz de paso de mensajes:

- `send(i,msg,size)` – envia el mensaje `msg` de tamaño `size` al proceso `i`
- `receive(i,&msg,size)` – recibe el mensaje `msg` de tamaño `size` del proceso `i`

Se pide:

a) Escribir el pseudocódigo que permite conectar a todos los procesos entre sí.
b) Escribir el pseudocódigo de la función broadcast asumiendo que todos los procesos están conectados entre sí.
c) Escribir el pseudocódigo de una función que implemente un broadcast fiable.
d) Se quiere implementar la función broadcast usando el API de sockets de C.
    1. Describir las cuestiones de diseño necesarias para realizar una implementación con sockets que cumpla los requisitos anteriores.
    2. Implementar el código que permite conectar a todos los procesos entre sí en C.
    3. Implementar la función `broadcast` en C.