Programación Concurrente

Práctica 7: Mensajes

1. Se desea definir un servidor que recibe dos tipos de mensajes de un cliente: "sigue" y "cuenta".

   a) Cada vez que el servidor recibe un mensaje "cuenta" debe mostrar por pantalla la cantidad de mensajes "sigue" que recibió desde la última vez que recibió "cuenta".

   b) Modificar la solución para hacer que el servidor en lugar de mostrar por pantalla el número correspondiente, lo envíe al cliente y sea este último el que lo muestre por pantalla. Dar también el código para el cliente.

2. Se desea definir un servidor que provea el siguiente comportamiento: debe generar un número pseudoaleatorio entero en el rango [0,10]. El cliente debe intentar adivinar el número que ha generado el servidor. Para ello envía sucesivamente mensajes conteniendo un número. Cada mensaje es contestado por el servidor indicando si el cliente acertó o no. Cuando un cliente acierta, el servidor queda disponible para aceptar un nuevo cliente. Notar que el cliente no puede enviar un nuevo mensaje hasta tanto el servidor no le conteste el anterior.

   a) ¿Cómo modificaría la solución si el cliente puede enviar mensajes aún cuándo el servidor no le ha contestado el anterior?

   b) ¿Cómo modificaría la solución si el cliente puede tener a lo sumo tres mensajes no contestados por el servidor?

3. Se desea implementar un servidor que recibe mensajes de dos clientes distintos y muestra por pantalla la concatenación de un par de mensajes (uno de cada cliente). Cada cliente continuamente envía un mensaje y espera una cantidad aleatoria de tiempo. El servidor recibe los mensajes de los clientes y por cada par de mensajes recibidos muestra por pantalla la concatenación de los mensajes recibidos.

4. Se desea definir un servidor que funcione como una variable compartida a la que los clientes pueden acceder mediante oportunos mensajes. Proponer una solución.

5. Considerar el servidor de trimming que recibe un String y responde con el resultado de aplicar la función trim al valor recibido.

   a) Dar una solución para el caso en que el servidor recibe solicitudes concurrente de 3 clientes distintos.

   b) Extender la solución para soportar un número no acotado de clientes.

6. Se desea resolver el siguiente problema utilizando comunicación por intercambio asincrónico de mensajes

   a) Se desea modelar un proceso Timer que recibe al momento de su creación un valor indicando la frecuencia con la que deberá generar ticks y el conjunto de canales en los que deberá señalar tales ticks. Por ejemplo, $Timer(100, [1, 2, 3, 4])$ deberá iniciar un proceso timer que cada 100 milisegundos genere mensajes (con contenido vacío) en los canales 1,2,3 y 4.

   b) Modificar la solución anterior asumiendo que el Timer se crea inicialmente con una lista vacía de canales y que los clientes que necesiten recibir los ticks deberán registrarse primero. Para ello se comunicarán con Timer a través del canal reservado 1. Los pedidos de registración pueden arribar en cualquier momento.

   c) Se desea modelar una red de sensores de temperatura. Cada sensor mantiene un valor real que representa la temperatura relevada en su ubicación. Este valor se modifica de dos maneras distintas: O bien recibe una lectura directa (que actualiza el valor que mantiene el sensor), o bien se actualiza en base a las lecturas de sus vecinos, siguiendo el siguiente protocolo. Cada sensor conoce una lista de sensores vecinos (puede asumir que estos se fijan al momento de la creación de la red). Todos los sensores están sincronizados utilizando un proceso Timer. Cada vez que el timer señala un tick, cada sensor envía su valor a todos sus vecinos y luego actualiza su valor calculando el promedio de todos los valores leídos.

7. Se desea definir un servicio de codificación y transmisión de información llamado T. El servicio T recibe la información que un cliente C quiere enviar a un servicio remoto R. El servicio T es el encargado de codificar cada mensaje recibido desde C y de reenviarlo a R. Puede asumir que cuenta con la función

   ```
   String codificar(String mensaje)
   ```

   que se encarga de codificar un mensaje.

   Se solicita

   a) Dar el código para el servicio T.

   b) Suponer ahora que el servicio T debe permitir a un cliente reenviar la información a distintos servicios remotos. Para ello el cliente debe indicar al servicio T quién será el destinatario de los mensajes que se enviarán a continuación (los destinatarios estarán identificados por un canal). Por ejemplo, el cliente puede comenzar indicando al servicio T que los mensajes que se enviarán a continuación deberán ser reenviados al servicio remoto 1, luego todos los mensajes que envíe a continuación el cliente, y hasta tanto este no notifique a T que desea cambiar el destinatario, deberán ser codificados y reenviados al servicio remoto 1.

   Asumir que el cliente siempre comenzará indicando al servicio T cual es el primer servicio remoto al que se deberán reenviar los mensajes. Sin embargo, no puede realizar ninguna suposición sobre la cantidad de mensajes que se enviarán a cada servidor remoto.

   Dar una solución para el servicio T que exhiba el comportamiento descripto.

   c) Suponer ahora que para codificar cada mensaje el servicio T necesita contar con una clave nueva que es generada por otro servicio de codificación K. Esta clave será recibida por el servicio T a través de un canal dedicado. Luego, el servicio T deberá codificar cada mensaje del cliente utilizando una clave nueva que recibirá desde el servicio K. Para realizar la codificación con una clave puede utilizar la función

   ```
   String codificar(String mensaje, String clave)
   ```

   Dar una implementación para el servicio T descripto anteriormente.

8. Se cuenta con un servicio S que permite hacer cálculos numéricos computacionalmente costosos. S recibe cada pedido de cálculo a realizar en un mensaje sobre el canal 1. El servicio toma el requerimiento, lo procesa y envía la respuesta a través del canal 2. Es decir, un cliente C envía un mensaje sobre el canal 1 y espera la respuesta en el canal 2. Como consecuencia de un mal diseño resulta que el cliente fue implementado de manera tal que puede enviar varios pedidos de cálculo sin esperar a que los anteriores hayan sido procesados y por otro lado S no se puede garantizar el correcto funcionamiento cuando S recibe un nuevo mensaje sobre el canal 1 y aún está procesando un mensaje anterior. Por este motivo, se desea implementar otro servicio proxy P que intermedie la interacción entre C y S. La idea es que el cliente C no interactuará directamente con S (es decir, no usará los canales 1 y 2), en su lugar interactuará sólo con P sobre los canales 3 y 4. La idea es que P reenvíe a S los mensajes que recibe de C y viceversa pero asegurando que S nunca recibirá un nuevo mensaje sobre el canal 1 hasta que no haya concluido de procesar el pedido anterior.

   a) Dar una implementación para P. La solución propuesta debe funcionar asumiendo que S no cambia y en C sólo se renombran los canales 1 y 2 por 3 y 4.

   b) Suponer ahora que S puede manejar a lo sumo N mensajes concurrentes de manera correcta. Extender la solución propuesta anteriormente para P de manera tal de maximizar la concurrencia. Asumir que S produce los resultados respetando el orden en que llegan los pedidos siempre y cuando haya a lo sumo N pedidos concurrentes.

   c) Extender la implementación de P para aprovechar el hecho de que existen dos copias de S (una que usa los canales 1 y 2 y la otra que usa los canales 11 y 22). P debe aprovechar las dos copias de S a fin de maximizar la concurrencia. Debe considerar que al cliente se le deben retornar los resultados en el orden en que hizo los requerimientos.