UNIVERSIDAD CARLOS III DE MADRID
DEPARTAMENTO DE INFORMÁTICA

Para la realización del presente examen se dispondrá de **1:30 horas.** **NO** se podrán utilizar libros ni apuntes.

Pregunta 1 (4 puntos) Responda a las siguientes preguntas:

a) (1 puntos) Utilizando el lenguaje de definición de procedimientos remotos XDR, especifique la interfaz asociada a un procedimiento remoto que recibe tres argumentos de entrada (una cadena de caracteres, un número entero y número en coma flotante de simple precisión) y devuelve dos argumentos de salida (una cadena de caracteres y un número entero).

Solución:

```
struct respuesta {
      string      s<>;
      int         a;
};

struct respuesta Procedimiento (string s, int a, float f) = 1;
```

b) (1.5 puntos) Implemente el código de la siguiente función utilizada en las prácticas de la asignatura:

```
int Enviar (int sd, char *buf, int len);
```

Esta función permite enviar a través de un socket TCP (con descriptor `sd`) un buffer (`buf`) de una determinada longitud expresada en bytes (`len`). La función devuelve -1 en caso de error y 0 en caso de éxito cuando todos los datos han sido enviados correctamente a través del socket.

Solución:

```
int Enviar(int socket, char *mensaje, int longitud)
{
      int r;
      int l = longitud;

      do {
            r = write(socket, mensaje, l);
            l = l -r;
            mensaje = mensaje + r;
      } while ((l>0) && (r>=0));

      if (r < 0)
            return (-1);   /* fallo */
      else
            return(0);  /* se ha enviado longitud */
}
```

c) (0.5 puntos) Definir brevemente en qué consiste el multicast atómico.

Solución:

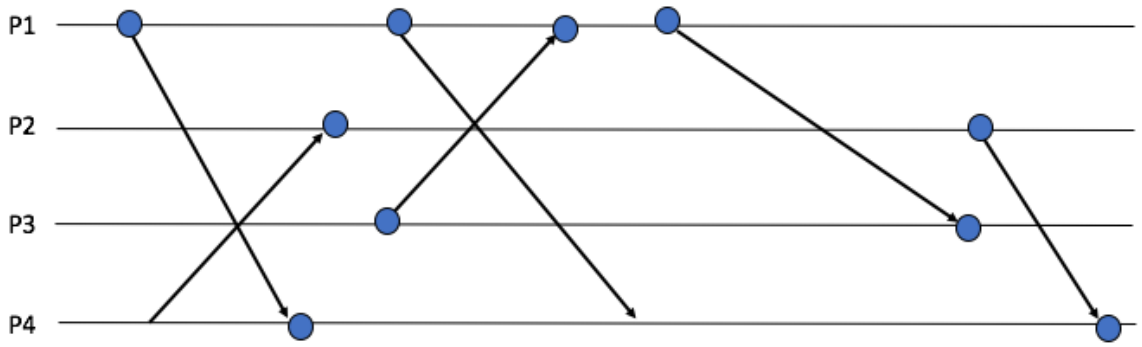
Es un multicast fiable (el mensaje es recibido por todos los nodos en funcionamiento) que asegura que todos los miembros del grupo reciban los mensajes de los diferentes nodos en el mismo orden.

d) (0.5 puntos) En una aplicación desarrollada utilizando llamadas a procedimientos remotos, ¿qué cambios hay que realizar en la aplicación completa cada vez que se actualiza la implementación del servidor?

Solución:

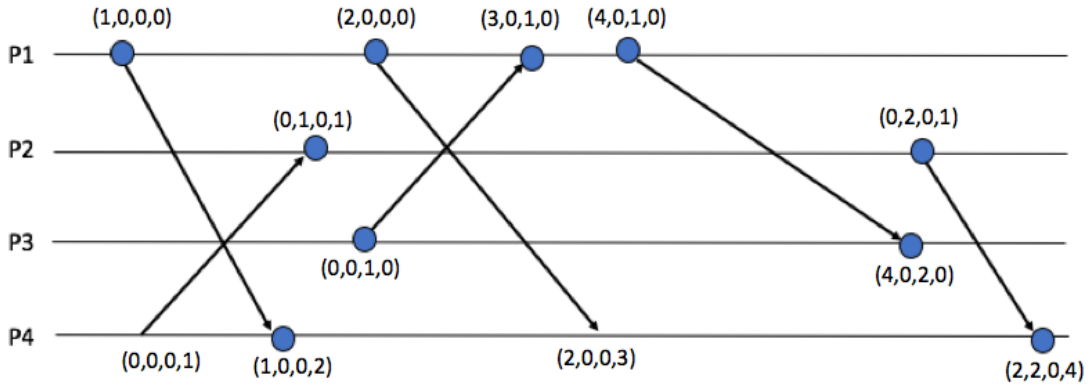
Puesto que la interfaz no cambia, no hay que realizar ningún cambio en el lado del cliente. En el lado del servidor no es necesario volver a generar los *stubs*, basta con modificar la implementación y volver a enlazar este código con los *stubs* del servidor para generar el nuevo ejecutable del servidor.

e) (0.5 puntos) Usando relojes vectoriales, defina las marcas de tiempo para los eventos de los siguientes procesos.



Dados dos eventos a y b cuyos relojes vectoriales asociados son (7, 6, 5) y (8, 7, 3) respectivamente ¿Cuál de los dos eventos precede al otro? Razone su respuesta.

Solución:



Las marcas de tiempo asociadas a los eventos a y b se caracterizan por (7,6,5) no es menor o igual que (8,7,3) y (8,7,3) tampoco es menor o igual que (7,6,5). Por tanto, ambos eventos son concurrentes.

Pregunta 2 (6 puntos) Se quiere desarrollar un sistema para la notificación de información en estaciones ferroviarias. El sistema consta de un servidor central con información de todas las circulaciones y una serie de estaciones cada una de la cuales recibe información de las circulaciones. La información que incluye una **circulación** es la siguiente:

- Hora de salida de la primera estación.
- Frecuencia de salida de los trenes desde la primera estación (número de minutos entre un tren y el siguiente).
- Lista de estaciones por las que pasa cada tren. Cada estación viene identificada por un nombre (por ejemplo: “Atocha”).
- Por cada estación se incluye el tiempo de parada del tren en esa estación, en minutos.

El sistema a diseñar sigue un modelo de suscripción. Es decir, cada vez que una estación desea recibir información sobre una determinada circulación, envía un mensaje de suscripción al servidor central. Cada vez que se produce una modificación en una circulación, el servidor central actualiza la información de esa circulación en todas las estaciones que se han suscrito a la misma. Una estación puede estar suscrita a varias circulaciones. Cada **circulación** viene identificada por un número entero.

Cada vez que una estación desea suscribirse a una circulación, lo solicita al servidor central y éste envía como respuesta a este mensaje de suscripción todos los datos asociados a esa circulación. Cada vez que se produce una modificación en una circulación, el servidor central envía los datos de la nueva circulación a todas las estaciones suscritas. El sistema incluye también un servicio para que las estaciones se puedan dar de baja en el sistema de una determinada circulación, de forma que dejará de recibir actualizaciones de dicha circulación. El servidor central también incluye un servicio que permite conocer si existe una determinada circulación y otro que permite conocer el número de estaciones por las que pasa una determinada circulación.

Considerando que se quiere desarrollar la aplicación utilizando sockets, se pide:

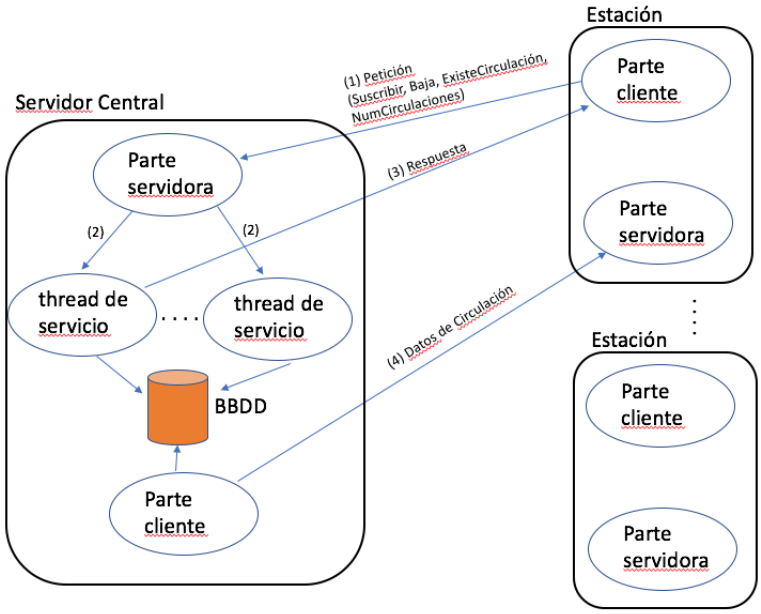
a) Haga un diseño detallado del sistema, haciendo todas las consideraciones que crea oportunas, teniendo en cuenta las características de cada uno de los componentes ofrecidos.
b) Especifique utilizando XDR el procedimiento remoto que permite suscribirse a una determinada circulación.

Solución:

a)

Como protocolo de transporte se va a utilizar TCP, que ofrece un esquema orientado a conexión que simplifica el desarrollo de la aplicación al ofrecer un canal orientado a flujos de bytes y mecanismos que permite conocer la existencia de fallos en la transmisión. Se asume que la dirección IP del servidor central es conocida por todos los equipos instalados en las estaciones. Asimismo, la estación central conoce las direcciones IP de todos los equipos instalados en las estaciones.

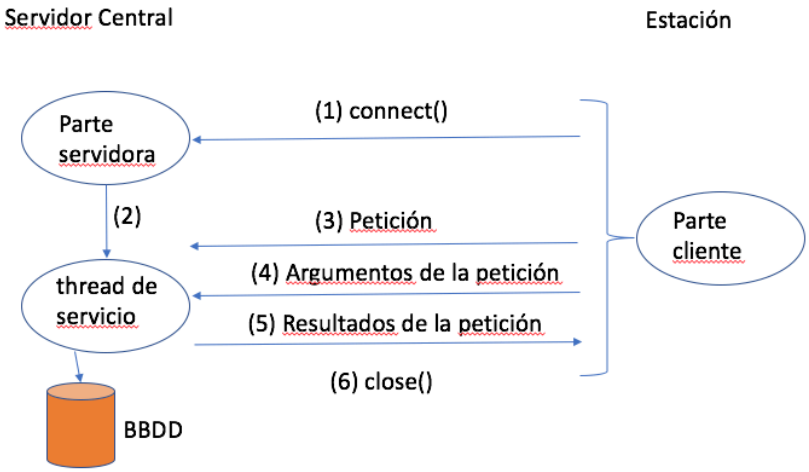
Un posible esquema del diseño es el que se muestra en la siguiente figura.



En cada estación va a haber una aplicación instalada con una parte cliente y otra servidora (dos *threads* de la misma aplicación). Se va a suponer que el puerto utilizado por la parte servidora es el mismo en todas las estaciones, por ejemplo, el puerto 2000. De igual forma, en el servidor central va a haber instalada una aplicación, con una parte servidora (cuyo puerto es conocido, por ejemplo, el 4000) y una parte cliente. La parte servidora de la estación central es la encargada de atender los cuatro servicios que ofrece la aplicación:

- Suscribirse a una nueva circulación
- Dar de baja la suscripción de una circulación
- Conocer si existe una circulación
- Conocer el número de estaciones que hay en una determinada circulación

La parte cliente de la estación es la que se va a encargar de solicitar alguno de estos cuatro servicios. Dado que se utiliza, TCP, el esquema de conexión entre ambas partes va a ser el siguiente, el cual utiliza un modelo de conexión por petición.



Cada vez que la parte cliente de la estación desea solicitar un servicio establece una conexión (1) con la parte servidora que ejecuta en el servidor central (IP del servidor, puerto 4000). A continuación, la parte servidora crea un thread (2) para atender la petición. La parte cliente envía (3) un mensaje identificando la

petición (Suscribirse a una nueva circulación, Dar de baja la suscripción de una circulación, Conocer si existe una circulación y Conocer el número de estaciones que hay en una determinada circulación). Una vez enviado este mensaje envía (4) en uno o varios mensajes los argumentos de la petición y obtiene un mensaje con los resultados de la petición (5). Finalmente, ambas partes cierran la conexión (6). Como puede verse la parte servidora de la estación central, se trata de un servidor concurrente que crea un thread para atender cada nueva petición. Este thread tiene acceso a una base de datos donde residirá toda la información que necesita la aplicación.

A continuación, se describe el formato de los mensajes utilizados en esta interacción:

- **Petición** (1). Puede identificarse con un byte:
  - 0 para suscribirse
  - 1 para darse de baja
  - 2 para conocer si existe una circulación
  - 3 para conocer el número de estaciones de una circulación.

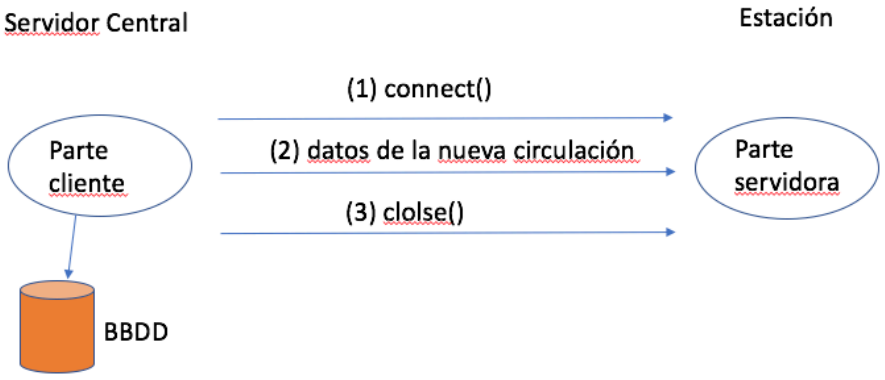
  También podría utilizarse una cadena de caracteres finalizada con el carácter `\0´para identificar cada petición:
  - "ALTA" para suscribirse
  - "BAJA" para darse de baja
  - "EXISTE" para conocer si existe una circulación
  - "NUMERO" para conocer el número de estaciones de una circulación
- **Argumentos para todas las peticiones (2)**. Todas las peticiones envían como argumento el número de circulación. Este puede venir identificado por una cadena de caracteres que identifica la circulación codificada en texto ("234"). La cadena finaliza con el código `\0´para identificar el fin de la cadena.
- **Resultado para el mensaje de suscripción** (3). En este caso en primer lugar se envía al cliente como respuesta, información sobre la existencia o no de la circulación. Esta información se devuelve codificada en un byte (0: existe, 1 no existe). En caso de recibir un 1, se cierra la conexión. En caso de que la circulación exista el cliente recibirá todos los datos asociados a la circulación. Los datos de una circulación incluyen:
  - Hora de salida de la primera estación.
  - Frecuencia de salida de los trenes desde la primera estación (número de minutos entre un tren y el siguiente).
  - Lista de estaciones por las que pasa cada tren. Cada estación viene identificada por un nombre (por ejemplo: "Atocha").
  - Por cada estación se incluye el tiempo de parada del tren en esa estación, en minutos.

  Como formato para todos estos datos se puede enviar una secuencia de cadenas de caracteres, cada una de las cuales finaliza con el código `\0´. Para identificar el fin de la secuencia se pueden utilizar dos códigos `\0´ seguidos. De esta forma, el resultado vendría codificado de la siguiente manera:
  - Una cadena de caracteres con la hora de salida de la primera estación en formato "HH:MM:SS". La cadena finaliza con `\0´.
  - Una cadena de caracteres que codifica la frecuencia en minutos ("240", por ejemplo). La cadena finaliza con `\0´.
  - Por cada estación se envía:
    - Una cadena con el nombre ("ATOCHA"). La cadena finaliza con `\0´.
    - Una cadena que codifica el tiempo de parada en dicha estación ("120", por ejemplo). La cadena finaliza con `\0´.
  - El fin de la secuencia se identifica con dos códigos `\0´seguidos.
- **Resultado para el mensaje de baja** (3). En este caso se puede enviar un simple byte que codifica el resultado de la operación (0: fallo, 1: éxito, etc.).
- **Resultado para la petición que permite conocer si existe una circulación** (3). Se puede devolver un simple byte con el resultado (0: existe, 1: no existe).

- **Resultado para la petición que permite conocer el número de estaciones de una circulación** (3). Se puede enviar una cadena de caracteres (“23”), que codifica este número. La cadena finaliza con el código ´\0´.

Cada vez que una estación se suscribe a una circulación, el servidor guardará el par <número de circulación, dirección IP), para identificar la estación que está suscrita a una determinada circulación. Toda la información que necesita la aplicación residirá en una base de datos accesible por la parte servidora y cliente del servidor central.

De forma desacoplada a estas dos partes, cada vez que hay un cambio en una circulación, la parte cliente del servidor central enviará los datos de la nueva circulación a todas aquellas estaciones suscritas. Para ello, se utilizará una modelo de conexión por cada nuevo envío que seguirá el siguiente esquema:



Cada vez que haya cambios en una circulación que haya que notificar. La parte cliente realizará por cada estación suscrita las siguientes acciones:

- Establecerá una conexión (1) con la parte servidora de la estación (que ejecuta en el puerto 2000 y en una determinada dirección IP).
- Enviará una cadena de caracteres que codifica el número de la circulación (“234”, por ejemplo). La cadena finaliza con `\0´.
- Para el resto de datos de la circulación se seguirá un esquema como el descrito anteriormente. Como formato para todos estos datos enviará una secuencia de cadenas de caracteres, cada una de las cuales finaliza con el código `\0´. Para identificar el fin de la secuencia se pueden utilizar dos códigos `\0´ seguidos. De esta forma, el resultado vendría codificado de la siguiente manera:
    - Una cadena de caracteres con la hora de salida de la primera estación en formato “HH:MM:SS”. La cadena finaliza con `\0´.
    - Una cadena de caracteres que codifica la frecuencia en minutos (“240”, por ejemplo). La cadena finaliza con `\0´.
    - Por cada estación se envía:
        - Una cadena con el nombre (“ATOCHA”). La cadena finaliza con `\0´.
        - Una cadena que codifica el tiempo de parada en dicha estación (“120”, por ejemplo). La cadena finaliza con `\0´.
    - El fin de la secuencia se identifica con dos códigos `\0´seguidos.

c) A continuación, se especifica un posible formato de definición para el servicio de suscripción:

```
struct listaEstaciones {
   string           estación <>;
```

```
   int               tiempoParada;
   listaEstaciones   *next;
};

typedef  listaEstaciones *t_lista;

struct restulado {
   string      horaSalida<>;
   int         frecuencia;
   lista       t_lista;
};

program CIRCULACIONES {
     version CIRCULACIONESVER {
         resultado Suscribir(int numCirculacion) = 1;
     } = 1;
 } = 99;
```