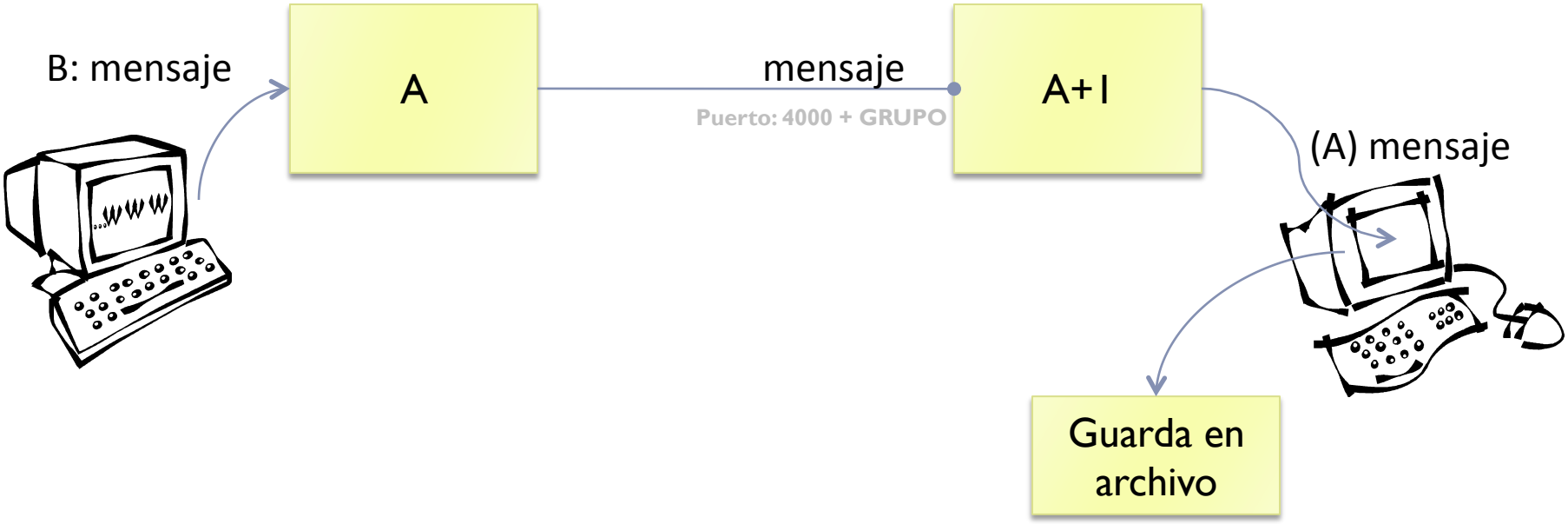
Práctica 1. Introducción a la programación con libring: almacenamiento remoto de mensajes

Laboratorio de Arquitectura de Redes de Comunicaciones

Enunciado: “almacenamiento remoto de msgs”

En esta práctica deberá implementar un servicio de almacenamiento remoto de mensajes desde un equipo al SIGUIENTE empleando para la librería libring. El funcionamiento básico se describe en la siguiente figura.



Enunciado: “almacenamiento remoto de msgs”

Requisitos:

- Cada equipo debe ser simultáneamente cliente (enviará mensajes) y servidor (recibirá y almacenará mensajes).
- **Como cliente**, el programa se ejecutará en un bucle infinito, donde el usuario escribirá órdenes con la siguiente sintaxis:

```
> /Redactor/Mensaje escrito por el Redactor. El mensaje es hasta el primer
retorno de carro (\r) introducido
> /Otro redactor/Otro mensaje de otro redactor
```

Enunciado: “almacenamiento remoto de msgs”

Requisitos:

- **Como servidor**, el programa se ejecutará en un bucle infinito, almacenando en un archivo (llamado “logs”) los mensajes.
- Cada mensaje recibido se almacenará siguiendo la sintaxis que se muestra en los ejemplos mostrados más abajo.
- El archivo “logs” debe tener un encabezado (creado por el servidor) tal y como se muestra en el ejemplo:
  - Día y hora de creación del archivo, Datos de cabecera (Origen fecha hora mensaje) y separador

```
Sun Oct 28 10:12:22
Archivo creado por 5
Fecha      hora    mensaje

_____________________________________________________________________
Mon Oct 29 13:23:12 mensaje de un equipo (Redactor)
Mon Oct 29 13:45:32 Otro mensaje de otro equipo (Otro redactor)
```

Desarrollo de la práctica

Procederemos de modo incremental. En la **primera versión** nos centraremos en desarrollar los dos bloques básicos (cliente y servidor) de modo totalmente independiente. En esta versión, el servidor tan sólo debe mostrar la información por pantalla. No debe almacenar en archivo.

Herramientas para el desarrollo:

- Lenguaje C (compilaremos con gcc + Makefile).
- Librería libring
- Librerías de I/O básica

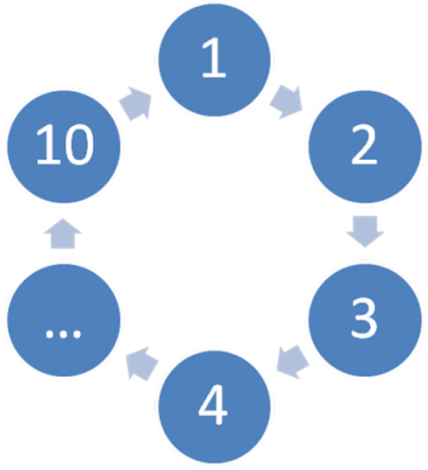
Librería libring

- Librería de comunicación a nivel de enlace no orientada a conexión

ANTES DE CONTINUAR DEBE LEER EL MANUAL DE LIBRING. SE ENCUENTRA EN AULA VIRTUAL.

Librería libring

- Soporta envío y recepción de tramas en una topología en una red en anillo formada por las PCs del laboratorio como la mostrada en la figura



TOPOLOGIA DE LA RED

- En esta red las tramas se enviarán en sentido horario, esto es, si 1 quiere enviar a 3 el datagrama será enviado a 2, y éste reenviará a 3. En el caso de que 3 quiera responder a 1 el datagrama viajará a través de 4, 5, …10, y finalmente entregado a 1.
- El envío/recepción proporcionados por la librería es NO CONFIABLE. Es decir, pueden producirse pérdidas y retardos en los mensajes!

Librería libring

- En la versión de libring (v1.0) proporcionada puede activarse una directiva de compilación LIBRING_CONFIABLE para evitar pérdidas y retrasos de tramas. Es decir, compilar libring, llamando al gcc:

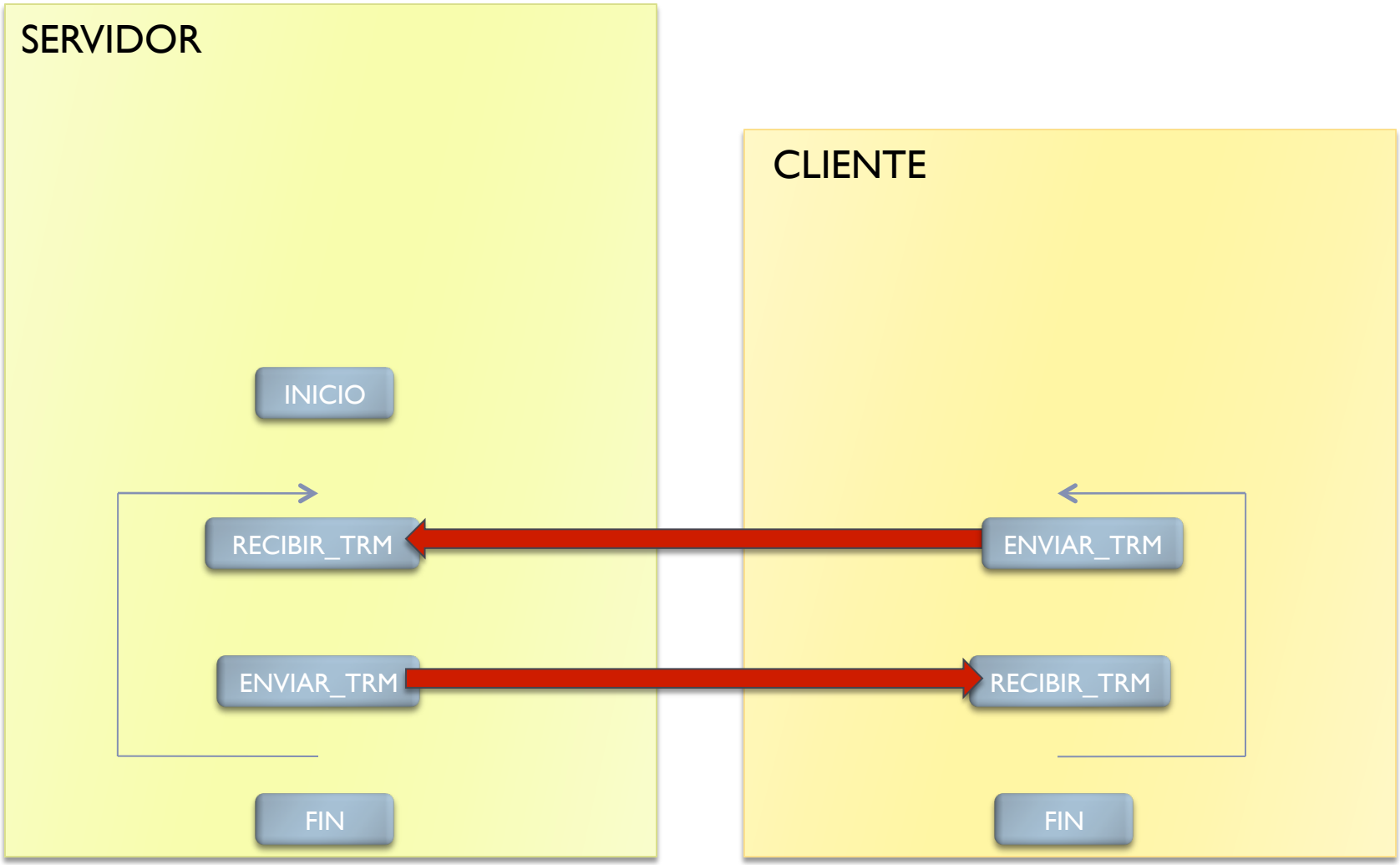
```
gcc -c libring.c -DLIBRING_CONFIABLE
```

- En esta práctica (y posteriores) puede hacer uso de dicha directiva para realizar pruebas del protocolo con comodidad. No obstante, las prácticas se probarán sin esta opción activa.

- Para esta práctica debe activarse otra directiva de compilación, LIBRING_SIMPLERING, que permite emplear una red en anillo creada por 2 PCs. Por ejemplo, para emplear sólo los PCs 3 y 4 compilar:

```
gcc -c libring.c -DLIBRING_SIMPLERING=3
```

Modelo básico de comunicación con libring



Ejercicio de mensajería (1ª versión)

EJERCICIO: Escriba el cliente de mensajería (cliente.c) con las características solicitadas en la primera versión haciendo uso de las librerías anteriores

EJERCICIO: Escriba el servidor (servidor.c) de mensajería con las características solicitadas en la primera versión haciendo uso de las librerías anteriores

EJERCICIO: Escriba un Makefile adecuado para la compilación simultánea de cliente y servidor **¡¡Hacer esto en primer lugar!!**

Desarrollo de la práctica (2ª Versión)

En la **segunda versión** implementaremos en el programa servidor un aviso de inactividad. Si transcurridos 30 segundos no se ha recibido ningún mensaje, debe indicarlo por pantalla.

Además de imprimir por pantalla los mensajes recibidos, el servidor deberá almacenarlos en el archivo "logs". Sólo debe almacenar los mensajes recibidos, no los avisos de inactividad.

Herramientas para el desarrollo:

- Lenguaje C (compilaremos con gcc + Makefile).
- Librería de libring (DEBE USAR LA OPCION SIMPLERING)
- Librerías de I/O básica

Desarrollo de la práctica (3ª Versión)

Ésta será la **versión final**. Para ello haremos que un mismo binario se comporte a la vez como cliente y servidor.

Herramientas para el desarrollo:

- Lenguaje C (compilaremos con gcc + Makefile).
- Librería de libring
- Librerías de I/O básica
- Librerías de gestión de procesos (funciones FORK y WAIT)

Funciones fork y wait

- Permite dividir la ejecución de un proceso en dos (padre e hijo) totalmente independientes (espacio de estado -variables- separado).
- Uso típico (ver ejemplo en la web de la asignatura):

```
if ( (pid = fork()) ) {
          /* Código del padre */

          /* Esperamos por la terminación del hijo */
          wait(NULL);
} else {
          /* Código del hijo */

}
```

CUESTIÓN: Ejecute un man de las funciones fork y wait, ¿qué uso tiene cada parámetro?

Ficheros a entregar

EJERCICIO: Unifique el cliente de almacenamiento en un único programa (almac.c) haciendo uso de las funciones fork y wait, y con las características solicitadas en la primera y segunda versión. Proporcione un Makefile adecuado para la compilación con las reglas tar y clean.

Copie el fichero almac.tgz correspondiente al directorio practica1 del directorio "entregas". Este fichero debe contener los ficheros fuentes (.c y .h) y el Makefile.

Recomendación: No use un bloque de código monolítico, es mejor utilizar funciones (por ejemplo, procesa_cliente() y procesa_servidor()). Las macros (ej: el PUERTO de trabajo) y los cuerpos de las funciones deben definirse en un fichero de cabecera (.h) aparte.