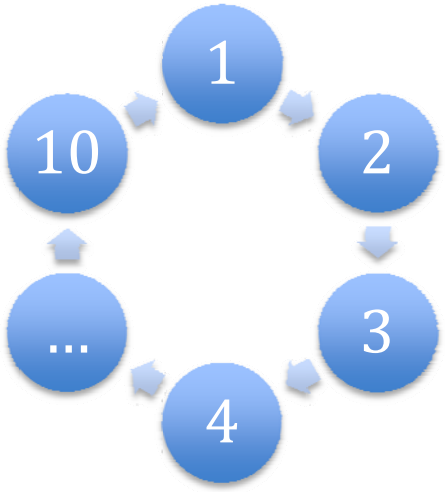
Descripción del Protocolo de comunicación fiable en una red de anillo

En el contexto del laboratorio los equipos se han configurado con la siguiente topología en anillo:



Es decir, que el equipo N sólo puede transmitir paquetes al equipo (N+1), si N<10, o desde el equipo 10 al 1.

En esta configuración se considera el desarrollo de un protocolo de envío/recepción de mensajes que se construirá sobre libring. Es decir, el protocolo a construir hará uso de las funciones enviar(), recibir(), localhost() y éstas exhibirán el mismo comportamiento "no fiable" (pérdidas, retardos) que en el desarrollo de la primera práctica de la asignatura.

El protocolo a desarrollar debe permitir:

- El envío de mensajes entre dos pares cualesquiera de la red (origen, destino), teniendo en cuenta las restricciones topológicas. Es decir, si por ejemplo 1 desea enviar un mensaje a 3, éste debe dar un salto a través de 2.

- Entrega confiable de mensajes. Es decir, los paquetes deberán ser asentidos mediante un mensaje de ACK. En el ejemplo anterior, el ACK deberá viajar de 3 a 1 a través de los nodos 4, 5, 6, 7, 8, 9 y 10.

Asimismo, es necesario solucionar los problemas de duplicados que se pueden producir por la posible pérdida de paquetes de ACK. Es decir, deberá incluirse una numeración de paquetes y guardar en destino información con el último número de paquete asentido según el nodo de origen.

El sistema funcionará siguiendo un esquema síncrono. Cada **T** segundos (tiempo de **timeslot**) se enviará el contenido de una variable buffer como mensaje i-ésimo. El nodo destino estará indicado en una variable global llamada ***destino***, el contenido del mensaje en ***buffer*** y el tamaño del mismo en ***bufferlen*** (también variables globales). Para implementar la entrega confiable de mensajes, si un equipo no recibe confirmación de este mensaje i-ésimo durante el tiempo de timeslot, retransmitirá el mensaje en el siguiente timeslot, así hasta recibir el ACK correspondiente. En caso de recepción del ACK, llamará a la función actualizarbuffer() (debe asumir que esta función ya existe) que actualizará *buffer* y *bufferlen* con el siguiente mensaje y la variable *destino*. En el siguiente timeslot su programa deberá continuar enviando el mensaje (i+1)-ésimo.

Asimismo, durante un timeslot el sistema debe atender los mensajes entrantes:

- Si son para el propio nodo debe almacenarlos en una lista enlazada que use las funciones creadas en la práctica 2
- Si son para otro nodo debe enviarlo al siguiente nodo del anillo
- Si es un ACK para el propio nodo debe procesarlo
- Si es un ACK para otro nodo debe enviarlo al siguiente nodo del anillo
- Cuando haya recibido **Nmax** mensajes, debe concatenar su contenido en un nuevo buffer y entregarlo llamando a la función entregarbuffer(char *nuevobuffer) (debe asumir que esta función ya existe)

El objetivo así es crear un protocolo de comunicación fiable entre equipos en el que cada equipo ejecutará un programa que procesará los mensajes recibidos y se ocupará de transmitir los nuevos mensajes enviados por la aplicación.

Debe consultar como información adicional los apuntes de la práctica 3 en aula virtual.

En el guión final de la práctica 3 se detallará el resto de los aspectos como: nombre de cada ejecutable, parámetros que deben aceptar y formato de estos, librerías y funciones a emplear, etc.