LARC

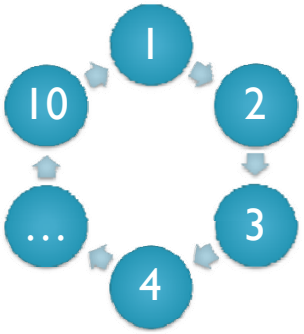
Anteproyecto – Curso 2010/2011

Protocolo de comunicación fiable en una red de anillo

Objetivo

- Desarrollar un **protocolo de nivel de red para comunicar equipos** entre sí de modo fiable.
- Conocer y solventar los problemas asociados a las capas de red inferiores.
- Adquirir soltura en el desarrollo de aplicaciones C.

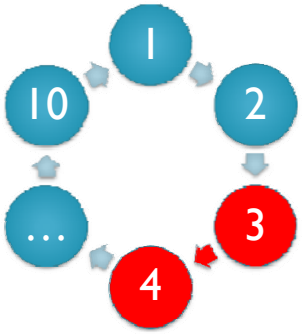
Visión general de nuestro escenario



Topología de red

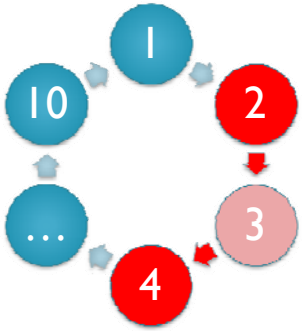
El sentido de la comunicación es único

Visión general de nuestro escenario



- Función **ENVIAR**
- Función **RECIBIR**

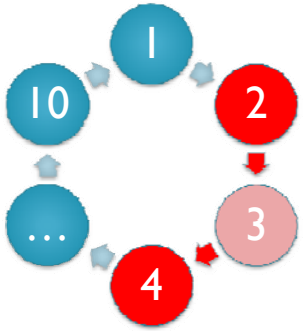
Visión general de nuestro escenario



Se desea implementar un protocolo de comunicación confiable, entre dos host cualquiera del anillo.

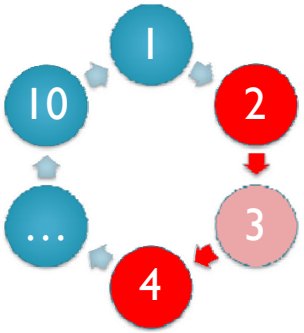
Los nodos intermedios actuarán como puentes en la comunicación

Visión general de nuestro escenario



¿Qué sucede si se pierde una transmisión?

Visión general de nuestro escenario

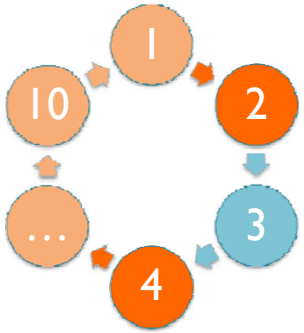


¿Qué sucede si se pierde una transmisión?

- Debe retransmitirse!

¿Pero cómo se sabe que no ha llegado?

Visión general de nuestro escenario



Necesario el uso de ACK ➔ Es necesario circular todo el anillo para una transmisión correcta

Visión general de nuestro escenario

- Asimismo, los ACKs pueden perderse, ¿qué consecuencias produce?
  - DUPLICADOS del paquete en el otro extremo!
- Podemos tener paquetes fuera de orden
  - No si no se transmite el siguiente hasta recibir el ACK correspondiente → PARADA y ESPERA

Requisitos

- Es necesario definir distintos tipos de paquetes: DATOS y ACK
- Es necesario incluir numeración en los paquetes (para corregir los duplicados)

Desarrollo

- El programa funcionará siguiendo un esquema síncrono. Cada T segundos (tiempo de *timeslot*) se enviará el contenido de la variable *buffer* como mensaje i-ésimo.
- El nodo destino estará indicado en una variable llamada *destino*. Si un equipo no recibe confirmación de este mensaje i-ésimo, retransmitirá el mensaje en el siguiente timeslot, así hasta recibir el ACK correspondiente.
- En ese momento, 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*, y en el siguiente *timeslot* enviará el mensaje (i+1)-ésimo.

Desarrollo

- Asimismo, el programa debe atender todos los mensajes entrantes:
  - Si son para el propio nodo debe almacenar su contenido en una lista enlazada.
  - 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)
  - 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

¿Cómo ejecutar acciones específicas cada T segundos?

```
repetir
   deltaT = T
   repetir
        t1= tiempo
        recibir(..., deltaT,...) → tiempo máximo T segundos
        t2 = tiempo
        // procesar el paquete recibido, si lo hay
        deltaT -= (t2-t1) → El tiempo que falta hasta T
        // si deltaT <= 0 → Ejecutar acciones periódicas
   fin
fin
```

Otras consideraciones

- Cada grupo debe entregar un anteproyecto rellenando el documento que aparece en aula virtual
- En el enunciado específico de la práctica 3 se indicarán otros detalles de funcionamiento (ej. los parámetros Nmax, T, etc.) y los formatos precisos requeridos para la práctica 3.