Proyecto 1 Capa Física

Se quiere modelar el comportamiendo de varios dispositivos de red conectados a un mismo medio de comunicación. Vamos a simular una serie de dispositivos conectados a una misma red por cable.

Elementos a implementar

1. El **Cable** conecta 2 dispositivos. En un mismo cable pueden ocurrir colisiones si varios dispositivos intentan escribir a la misma vez.
2. La **Computadora** es un dispositivo que envia y transmite información a la red. Tiene un solo puerto de comunicación al que se conecta un cable.
3. El ***Hub*** (concentrador) es un dispositivo que permite centralizar el cableado de una red de computadoras. Este dispositivo recibe una señal por uno de sus puertos y transmite la misma señal por todos los demás puertos.
    - El *hub* tiene una cantidad de puertos variable, normalmente 4 u 8. (se especifica en el script de ejecución)
    - A cada uno de esos puertos se puede conectar un cable hacia otro dispositivo.

Transmisión de información.

- Cuando se hable de “transmitir información” esta se realizará un *bit* cada vez.
- Cada dispositivo tiene como mínimo un puerto. Las computadoras tienen 1 solo puerto, los *hubs* tienen un cantidad de puertos variable.
- Un puerto de un dispositivo se conecta con un puerto de otro dispositivo a través de un cable. **Ejemplos**:
    - una computadora con otra
    - una computadora con un *hub*
    - un *hub* con otro *hub*
- El dispositivo que transmite, escribe el valor del *bit* que quiere transmitir (0 o 1) por el cable (o por cada uno de los puertos en el caso de los *hubs*) durante un tiempo (configurable, por defecto 10 ms, *keyword*: **signal_time**)
- Durante ese tiempo, cualquier lectura que se haga en ese cable/puerto el valor debe ser el mismo. Ejemplo: si el dispositivo A escribe un 1 en el canal en el momento t = 0ms, y otro dispositivo consulta la información que hay a los 3ms va a obtener un 1, y si pregunta a los 7ms va a obtener de nuevo un 1.
- Si hay varios dispositivos conectados al mismo *hub* y solo uno está transmitiendo, todos los demás serán capaces de recibir la información simultáneamente y obtendrán la misma información.
- Si un dispositivo quiere transmitir un *byte*, tiene que trasmitir 1 *bit* a la vez. Ejemplo. Si quiere escribir 10101010. primero escribe un 1, espera 10 ms, luego escribe un 0, espera 10ms, y asi sucesivamente. Es importante recalcar que durante 10ms cualquier lectura que se haga de la misma respuesta.
- O sea, no es que haya un delay de 10ms entre *bit* y *bit*, sino que cada bit se mantiene “en transmisión” durante ese tiempo. Durante 10ms se está transmitiendo el primer bit, y durante los proximos 10ms se está transmitiendo el 2do bit.
- Un dispositivo “sabe” que la información que está transmitiendo es correcta si luego de transmitir, realiza una lectura y el valor que obtiene es el mismo que transmitió.
- Si dos dispositivos escriben en el mismo cable a la misma vez, el cable tomará un XOR entre ambos *bits*. **Ejemplo**:
    - t = 0ms el dispositivo A transmite un 1. realiza una lectura un obtiene un 1. todo ok

    - t = 10ms el dispositivo A transmite un 0. realiza una lectura y obtiene un 1. error, alguien más está escrbiendo en el canal.
    - Si el dispotivo A emite un 1 y el dispositivo B tambien emite un 1. el cable hace un XOR entre ambas señales por lo que ambos dispositivos reciben un 0 y se dan cuenta que hay alguien mas transmitiendo.
- Si ningún dispositivo está transmitiendo en el canal, el canal tiene un valor **nulo**/vacío.
    - No es lo mismo que haya un cero en el canal a que el canal esté vacío.
    - Una PC cuando haga una lectura en el canal si detecta que el canal está vacío, no escribe nada en el `.txt` correspondiente.

Instrucciones

A continución de muetran las posibles instrucciones que puede tener el `script.txt`

- `<time> create hub <name> <cantidad de puertos>`
    - **ejemplo**: `0 create hub h 4`
    - los puertos de los *hubs* se identifican con el nombre del *hub*, un guión bajo y un numero entre 1 y la cantidad de puertos. En el ejemplo anterior el *hub* se llama **h** y los puertos se llaman **h_1**, **h_2**, **h_3** y **h_4**

- `<time> create host <name>`
    - esta instruccion permite adicionar una nueva computadora al sistema
    - el nombre del único puerto que tiene la computadora es igual al nombre de la computadora concatenado con un "_1". en el ejemplo siguiente la computadora se llama **pc**, por lo tanto el puerto se llama **pc_1**
    - **ejemplo**: `0 create host pc`

- `<time> connect <port1> <port2>`
    - conecta 2 puertos de 2 dispositivos
    - **ejemplo**: `0 connect h_1 pc_1`

- `<time> send <host> <data>`
    - este comando permite que 1 computadora decida empezar a transmitir una información.
    - Esta información será siempre múltiplos de un *byte*, y se especificarán todos los *bits*
    - **ejemplo**: `0 send pc_1 0101010111001100`

- `<time> disconnect <port>`
    - con este comando se desconeta un cable de uno de los puertos
    - puede ser en medio de una transmisión
    - **ejemplo**: `20 disconnect pc_1`

NOTA: no existe ninguna instrucción read porque todos los dispositivos se encuentran siempre leyendo lo que hay en el canal.

Salida

- Un fichero `.txt` por cada computadora y dispositivo de la red.
- En cada fichero debe mostrarse línea por línea, cada intervalo de tiempo la información que ese dispositivo recibió o transmitió, especificando siempre el puerto.
- Además debe especificar si la transmisión fue satisfactoria o si hubo colisión en la transmisión debido a que otro dispositivo estaba transmitiendo en el mismo momento.
- Esta ultima especificación solo es necesaria para las computadoras.
- **Ejemplo 1**: fichero `pc.txt`

    - `0 pc1 send 1 ok`
    - `10 pc1 send 0 collision`
    - `20 pc1 send 1 ok`
    - `30 pc1 send 0 ok`
- **Ejemplo 2**: fichero `h.txt`
    - `0 h_1 receive 1`
    - `0 h_2 send 1`
    - `0 h_3 send 1`
    - `0 h_4 send 1`

Objetivo del proyecto

Como dijimos anteriormente, si dos computadoras deciden enviar información a la misma vez se producirán colisiones. El objetivo de este proyecto es que **ustedes decidan como resolver ese problema, busquenle solución a eso**. Invéntense algun protocolo, como las computadoras se pondrán de acuerdo para decidir de quien es el turno de escribir.

Tenga en cuenta que estamos trabajando en el nivel de comunicación más básico que existe. Es decir no hay número IP, ni dirección MAC ni nada similar. Toda información que emita una computadora llega a todas las demás.

El programa debe continuar hasta que todas las computadoras hayan terminado de transmitir su información. Es posible que una computadora envie el mismo *bit* varias veces si las veces anteriores colisionó con otro.