Proyecto 2

Capa de Enlace +

Detección y Corrección de Errores

Para esta 2da etapa del proyecto de redes incorporaremos nuevos dispositivos y conceptos relacionados con la capa de enlace.

Elementos a implementar

1. **Cable Duplex**. A partir de ahora, los cables tienen la capacidad de transmitir información en dos sentidos. Los *hosts* pueden estar leyendo y transmitiendo a la misma vez. Piénsenlo de la siguiente forma: ahora el cable físico, tiene internamente varios hilos/pelos por los que viaja información, similar a los cables de corriente que tienen un vivo y un neutro. En este caso hay uno que transmite "hacia una dirección" y el otro que transmite "en la otra dirección".
    - Puede seguir habiendo colisiones si varios *hosts* transmiten simultáneamente.
    - Se debe garantizar que si hay solamente 2 *hosts* transmitiendo en el mismo *cable/hub* que no haya colisiones porque cada uno de los *hosts* transmitirá lo suyo y leerá lo del otro. Pero en el momento que aparezca un 3ro transmitiendo deben empezar a ocurrir las colisiones.
2. **Dirección *MAC***. Cada computadora a partir de ahora, tiene una dirección *MAC* única. Esta dirección *MAC* está compuesta por 2 *bytes* (16 *bits*)
3. **Trama** (***frame***). Cuando una computadora quiera transmitir información, la transmitirá en forma de una trama que tiene la siguiente estructura:
    - Primeros 16 *bits*: dirección *MAC* a la que está dirigida la trama. Si el valor de la dirección *MAC* tiene 1 en todos sus *bits* (valor entero = `65535`) significa que está dirigida a todo el mundo (*broadcast*).
    - Próximos 16 *bits*: dirección *MAC* del *host* que transmite.
    - Próximos 8 *bits*: tamaño de los datos (en *bytes*). O sea si el *byte* contiene un `00001010` lo cual es un `10` en decimal, eso significa que a continuación vienen `10` *bytes* (`80` *bits*) de información.
    - Próximos 8 *bits*: tamaño de los datos de verificación.
    - A continuación vienen los datos (el tamaño especificado en el 3er campo).
    - A continuación vienen los datos de verificación (el tamaño especificado en el 4to campo)
4. ***Switch*** (**conmutador**). El *switch* es un dispositivo de conexión de redes, que tiene varios puertos (similar al *hub*) pero con una importante diferencia. El *switch* contiene una tabla de

las direcciones *MAC* de los dispositivos conectados a él y utiliza esta información para solo transmitir por el puerto al que está conectado el *host* receptor. De forma que un *switch* de 4 puertos puede tener 4 *hosts* conectados, por ejemplo: `pc1` y `pc2` comunicándose entre ellos y `pc3` y `pc4` comunicándose entre ellos, sin que ocurran colisiones.

  - Al iniciar la red el *switch* no tiene información sobre las direcciónes *MAC* de los dispositivos conectados. Pero va "aprendiendo" en la medida que se van transmitiendo datos por la red.

Transmisión de información.

- A partir de ahora hablaremos de transmitir *bytes* enteros, pero la transmisión sigue ocurriendo un *bit* cada vez. **Estos *bytes* se representarán en formato hexadecimal**.
- Si hay varios dispositivos conectados al mismo *switch* y uno transmite. Solo debe recibir información aquel dispositivo al que esté dirigido el mensaje. A menos que esté diridigo a todo el mundo (cuando la direccion *MAC* de destino es `1111111111111111` o `FFFF`).
- Puede darse el caso, que existan 3 *hosts* en la red y que 2 de ellos quiern transmitirle al 3ro simultáneamente. **Este es uno de los temas que ustedes tienen que aportar ideas o variantes de como enfrentar esa situación**.

Detalles de los algoritmos de detección de errores

- Cada *host* agregará al final de cada trama, información que servirá al *host* de destino para verificar y comprobar si los datos llegaron correctamente.
- Existen muchas técnicas y algoritmos posibles de los cuales ustedes escogerán (o inventarán) una o varias y las implementarán en el sistema.
- Estas técnicas deberían ser capaces de detectar si hubo algun cambio en los datos que se transmitieron.
- Una ténica simple y bastante común que pueden implementar es crear una especie de *hash* de los datos y enviarlos a continuación de los datos. Y el receptor, lee los datos, aplica la misma función *hash* y comprueba el resultado.
- Por ejemplo, **el hash más simple posible** es una suma de cada uno de los valores. Supongamos que la *data* es `AABBCCDD` (tamaño 4 *bytes* recuerden que se especifica en hexadecimal y un 1 *byte* tiene 2 dígitos hexadecimales):
  - 1er *byte* = `AA` (170 en decimal)
  - 2do *byte* = `BB` (187 en decimal)
  - 3er *byte* = `CC` (204 en decimal)
  - 4to *byte* = `DD` (221 en decimal)
  - suma total = 782
  - necesitas 2 *bytes* para transmitir eso y por lo tanto
    - el campo extra tendría valor 2 (`00000010`) lo cual indica que cuando se terminen de transmitir los datos, vienen 2 *bytes* más
    - al final de los datos transmites 2 *bytes* con valor 782 (`00000011 00001110`)
  - el receptor lee los datos, lee el campo extra, suma los valores de los datos y le tiene que dar 782.
- Cada algorítmo que ustedes implementen tienen que quedar bien documentado en el directorio docs/ del proyecto.
- **En caso de que decidan implementar uno solo:**
  - Cada *host* transmite los frames y rellena esa información de verificación utilizando ese algoritmo.
  - Y cada *host* recibe y analiza los datos usando ese mismo algoritmo.

- **En caso de que decidan implementar más de un algoritmo**:
  - En el fichero `config.txt` de su proyecto tienen que incluir un elemento configurable con *keyword*: **`error_detection`** cuyo valor corresponda con alguna de las diferentes técnicas implementas por ustedes:
    - Por ejemplo supongamos ustedes implementan un algoritmo llamado *CRC*.
    - E implementan además otro algoritmo llamado *Hamming*.
    - Ustedes tienen que documentar ambos algoritmos en el directorio `docs/`
    - Además tienen que poner el el fichero `config.txt` una entrada que sea `error_detection = crc`.
    - En ese caso, los *hosts* a la hora de transmitir o recibir utilizarán el algoritmo *CRC*.
    - Y si yo voy al fichero `config.txt` y donde dice "`crc`" pongo "`hamming`" entonces los *hosts* a la hora de transmitir y recibir información utilizarán esta otra técnica.

Instrucciones

A continución de muetran las posibles instrucciones adicionales que se incluyen en este 2do proyecto que puede tener el `script.txt`

- `<time> **create switch** <name> <cantidad de puertos>`
  - **ejemplo**: `0 create switch sw1 4`

- `<time> **mac** <host> <address>`
  - Este comando permite que asignarle una dirección MAC a una computadora.
  - La dirección *MAC* se especifica en hexadecimal
  - **ejemplo**: `0 mac pc_1 A4B5`

- `<time> **send_frame** <host> <mac destino> <data>`
  - con este comando se crea una trama y se transmite
  - la trama a enviar debe contener todos los campos especificados:
    - *MAC* destino
    - *MAC* origen
    - tamaño
    - campo extra
    - datos
  - la *MAC* de origen, el tamaño, etc. deben calcularse.
  - Una trama nunca podrá tener más de `255` *bytes* de datos.
  - Los datos se especifican en formato hexadecimal al igual que la *MAC*
  - **ejemplo**: `20 send_frame pc A5B4 A6F43400FFB34E…`

Salida

- Además de todo lo que se escribe actualmente (del proyecto anterior) se deben crear ahora unos ficheros nuevos por cada computadora de la red. Este fichero tendrá el nombre de la compudora seguido de `_data.txt`
- En este fichero se escribirán solamente los datos recibidos por esa computadora y quién se los envió. Aquí hay que tener en cuenta los datos que fueron enviados directamente a esa computadora, y también los datos que fueron enviados a todo el mundo.
- En el fichero `_data.txt` asociado a cada *host* se pondrá un **`ERROR`** al final de cada línea en los casos en los que los datos llegaron corruptos y ustedes no pudieron recuperarlos.
  - Si llegan los datos correctamente ustedes escriben en el fichero lo mismo que antes.
  - Si llegan datos corruptos, pero los algorítmos que ustedes implementan les permiten recuperar los datos, ustedes escriben lo mismo que antes en el fichero.

    - Si ustedes no son capaces de reconstruir la data original de ninguna forma, solo pueden saber que los datos están mal, entonces en la línea correspondiente ponen **`ERROR`** al final.
- El formato de salida debe tener en cada línea (separadas por espacio)
    - tiempo final de recepción de los datos
    - *MAC* que envió los datos
    - datos
- **Ejemplo**: fichero `pc_data.txt`
    - `100 FFA4 A6F43400FFB34E…`

Objetivo del proyecto

Como mencionamos anteriormente, la situación a resolver en este 2do proyecto es el caso en el que 2 o más computadoras decidan transmitir información simultáneamente hacia un mismo destino incluso con los cables duplex y los *switch*, hay desiciones que tomar. **Lo que se quiere es que ustedes piensen e implementen variantes de como resolver eso**. Y además implementar una o varias técnicas de detección y corrección de errores en el sistema.

Hasta el Momento

Elementos Físicos

- [1] Cable
- [1] Computadora / *Host*
- [1] Concentrador / *Hub*

- [2] Cable *Duplex*
- [2] Conmutador / *Switch*

Conceptos

- [2] Dirección *MAC*
- [2] Trama / *Frame*

Instrucciones

- [1] `<time> **create host** <name>`
- [1] `<time> **create hub** <name> <cantidad-de-puertos>`
- [1] `<time> **connect** <port1> <port2>`
- [1] `<time> **disconnect** <port>`
- [1] `<time> **send** <host> <data>`

- [2] `<time> **create switch** <name> <cantida-de-puertos>`
- [2] `<time> **mac** <host> <address>`
- [2] `<time> **send_frame** <host> <mac-destino> <data>`