

# Proyecto 3 Capa de Red

Para esta 3ra etapa del proyecto de redes incorporaremos nuevos dispositivos y conceptos relacionados con la capa de red.

## Elementos a implementar

1. **Dirección IP**. La dirección *IP* está compuesta por 4 bytes (32 *bits*). Varios dispositivos de la red, en particular los *hosts* y los *routers*, tendrán a partir de ahora uno o varios *IP* asociados.
  - Para los humanos, la dirección *IP* son 4 números enteros (0..255) separados por '.' por ejemplo 10.10.10.240 o 192.168.20.45. Para los dispositivos de red el *IP* son 32 *bits* consecutivos.
2. **Subred**. Una *subred* es un conjunto de *hosts* cuyos *IP* tienen todos un prefijo común.
  - La dirección **IP de una subred** se define como el prefijo común de todos los dispositivos de la *subred* y el resto de los *bits* son **ceros**.
  - La **dirección de broadcast** de una *subred* (no confundir con la dirección *MAC* de *broadcast*) se define como el prefijo común de todos los dispositivos de la *subred* y el resto de los *bits* son **unos**
  - **Por ejemplo:** en la *subred* con *IP* 10.6.122.0
    - La dirección de la *subred* ocupa 24 *bits* (los primeros 3 *bytes*)
    - Esa *subred* puede tener hasta 256 (el último *byte*) posibles direcciones *IP* (desde 10.6.122.0 hasta 10.6.122.255)
    - El prefijo común de todas las direcciones de esa *subred* es 10.6.122
    - La dirección 10.6.122.0 (contiene el prefijo y todos los demás *bits* son ceros) está reservada como dirección de la *subred* (no se le puede asignar a ningún *host* ni *router*)
    - La dirección 10.6.122.255 (contiene el prefijo y todos los demás *bits* son unos) está reservada como dirección de *broadcast* (no se puede asignar a ningún *host* ni *router*)
    - Lo que deja 254 posibles *IP* que se pueden usar en esa *subred* para los diferentes dispositivos (desde 10.6.122.1 hasta 10.6.122.254)
    - O lo que es lo mismo (los primeros 3 octetos coinciden con 10, 6, 122):
      - desde: 00001010 00000110 01111010 **00000001**
      - hasta: 00001010 00000110 01111010 **11111110**
  - **Otro ejemplo:** en la *subred* con *IP* 10.8.0.0
    - La dirección de la *subred* ocupa 16 *bits* (los primeros 2 *bytes*)
    - Esa *subred* puede tener hasta 65536 (los últimos 2 *bytes*) posibles direcciones *IP* (desde 10.8.0.0 hasta 10.8.255.255)

- El prefijo común de todas las direcciones de esa **subred** es 10.8
- La dirección 10.8.0.0 (contiene el prefijo y el resto de los *bits* son ceros) está reservada como dirección de la **subred** (no se le puede asignar a ningún *host* ni *router*)
- La dirección 10.8.255.255 (contiene el prefijo y el resto de los *bits* son unos) está reservada como dirección de **broadcast** (no se puede asignar a ningún *host* ni *router*)
- Lo que deja 65534 posibles **IP** que se pueden usar en esa **subred** para los diferentes dispositivos (desde 10.8.0.1 hasta 10.8.255.254)
- En este caso sería:
  - desde: 00001010 00001000 **00000000 00000001**
  - hasta: 00001010 00001000 **11111111 11111110**

3. **Máscara de subred**. La **máscara** es un número entero de 32 *bits* que siempre va a contener varios 1 en la parte más significativa y el resto serán ceros.

- Ejemplos de **máscaras** comunes:
  - 255.255.255.0 => 11111111 11111111 11111111 00000000
  - 255.255.0.0 => 11111111 11111111 00000000 00000000
  - 255.255.192.0 => 11111111 11111111 11000000 00000000
- Las **máscaras** se usan para poder determinar 2 cosas:
  1. si un *host* pertenece a una **subred**
    - para identificar un *host* a que **subred** pertenece se realiza una operación **AND** entre el **IP** y la **máscara**.
    - Por ejemplo el **IP** 10.6.122.44 con **máscara** 255.255.255.0. si uno aplica un **AND** entre ambos números queda 10.6.122.0
  2. la dirección de **broadcast** de un *host* (y de una **subred**).
    - Se hace una operación similar a la anterior, pero luego se rellenan con 1 toda la parte del número resultante que es cero en la **máscara**.
    - O sea, usando el mismo ejemplo anterior, si se quiere saber cual es la dirección de **broadcast** quedaría 10.6.122.255.
      - 10.6.122.44 => 00001010 00000110 01111010 00101100
      - 255.255.255.0 => 11111111 11111111 11111111 00000000
      - un **AND** entre ambos números daría
      - 10.6.122.0 => 00001010 00000110 01111010 00000000
      - La **máscara** tiene ceros en los últimos 8 *bits* por tanto, los últimos 8 *bits* puestos en 1 serían la dirección de **broadcast**, o sea:
        - 10.6.122.55 => 00001010 00000110 01111010 11111111

4. **Protocolo ARP.** (*Address Resolution Protocol*) Este protocolo permite obtener una *MAC* a partir de un número *IP*. Funciona de la siguiente forma:
- El *host* origen desea enviar una información a otro *host* del cual conoce su *IP* pero no conoce su *MAC*.
  - Crea un *frame* “especial” que tiene las siguientes características:
    - la *MAC* destino es FFFF (dirección de *broadcast*)
    - la *MAC* origen es la *MAC* del *host* que envía el *frame* (como siempre) que en este caso es el *host* que está preguntando por la *MAC* de cierto *IP*
    - la *data* solo contiene 8 *bytes*
      - los primeros 4 *bytes*: exáctamente “ARPQ” (*ARP Query*)
      - los próximos 4 *bytes*: el *IP* de destino del cual no se conoce la *MAC*)
    - no hay *info* extra
  - Todos los dispositivos de la subred reciben el mensaje (los *hubs* envían la *trama* por todos sus puertos, como siempre, los *switches* también envían la *trama* por todos)
  - Solo el *host* que tenga el *IP* anteriormente mencionado (en caso de que esté presente en la red, porque puede no haber nadie con ese *IP* nadie responde) debe responder el mensaje. Y lo hace con otro *frame* “especial” de respuesta, que tiene las siguientes características:
    - la *MAC* destino es la *MAC* del host que preguntó (en el mensaje de preguntar esta dirección venía en el campo *MAC* origen, ahora es el *MAC* destino, porque es una respuesta)
    - la *MAC* origen es ahora la *MAC* del host que tiene el *IP* por el cual se estaba preguntando. (**esta es la respuesta**)
    - la *data* contiene solamente contiene 8 *bytes*
      - los primeros 4 *bytes*: exáctamente “ARPR” (*ARP Response*)
      - los próximos 4 *bytes*: el mismo *IP* que se envió anteriormente
    - no hay datos extras
5. **IP Packet.** Un *paquete IP* se utiliza para comunicar 2 *hosts* que “hablan” entre ellos a nivel de capa de Red. En esta capa la comunicación entre *hosts* incluye los números *IP* de ambos, y se desconocen las *MAC* (la *MAC* es un concepto de la capa de enlace). Los paquetes *IP* tienen las siguientes características:
- **NOTA:** normalmente un *IP packet* viaja “dentro” de un *frame*.
  - Los primeros 4 *bytes*: *IP* destino
  - Próximos 4 *bytes*: *IP* origen
  - El próximo *byte* (1 solo *byte*): *TTL* (*time to live*) que por el momento dejaremos en cero.
  - El próximo *byte* (1 solo *byte*): *Protocolo*. Que también será cero por el momento.

- El próximo *byte* (1 solo *byte*): *Payload size* (cantidad de *bytes* a enviar)
  - A continuación vienen una cantidad de *bytes* que coincide con el valor del campo anterior y son los datos a enviar. (similar al campo *data* de los *frames*)
6. **Router (enrutador)**: Es un dispositivo de capa de red que viene varios puertos (similar al *hub* y al *switch*). El *router* tiene internamente una *tabla de rutas* que le permite enrutar los *paquetes* entre diferentes *subredes*.
- **NOTA:** todos los broadcasts que lleguen a un *router* por alguno de sus puertos, “mueren” ahí. O sea, los *routers* nunca propagan un broadcast de ningún tipo.
7. **Tabla de rutas**. Cada *router* (o *host*) tiene un tabla (que se le configura a través de comandos que veremos en la próxima sección) que permite decidir la ruta correcta para cada *paquete*, basado en el *IP de destino*.
- Cada *ruta* contiene los siguientes campos:
    - *Destination*: un número *IP* que puede ser de un *host* o de una *subred*
    - *Mask*: la *máscara* que se aplicará para comparar con el *destination*
    - *Gateway*: un número *IP* de otro *router* al cual se enviará el *paquete*. Este valor será cero (0.0.0.0), en el caso en el que la comunicación entre el *host* y el destino sea “directa”. (**Comunicación directa** significa que no pasa por ningún otro *router*, que están en la misma *subred*)
    - *Interface*: un número entre 1 y N (cantidad de puertos del *router* o *host*).
  - Las *rutas* se ejecutan en orden de **prioridad**. La prioridad la da la cantidad de unos de la *máscara*.
    - La *máscara* 255.255.255.255 (todo en 1) es la más prioritaria.
    - La *máscara* 0.0.0.0 (todo en 0) es la última en ejecutarse.
  - Cuando llega un paquete a un *router*, se analiza el *IP* de destino de ese *paquete* y se analizan cada una de las *rutas* (en orden de prioridad) hasta que una de ellas sea capaz de **enrutar** ese *paquete* hacia su destino.
  - Para determinar cual *ruta* es la que puede **enrutar** ese *paquete* hacia su *IP* de destino:
    1. Se coge una *ruta* R
    2. Se le hace un AND al *IP* de destino con el campo *máscara* de la *ruta* R
    3. Se compara el resultado del AND con el campo *destination* de la *ruta* R
    4. Si el resultado es igual, el *paquete* se envía por la interfaz correspondiente (especificada en el campo *inteface* de la *ruta*) hacia el *IP* especificado en el campo *Gateway*.
    5. Si el resultado del AND no coincide con el *destination* se pasa a evaluar la próxima *ruta*.
    6. Si ninguna *ruta* puede **enrutar** dicho *paquete* se le envía el *host* de origen un *paquete ICMP* con “*destination host unreachable*”.

8. **Tabla de rutas de un host:**

- Además de los *routers*, los *hosts* también van a tener una *tabla de rutas*, aunque tengan una sola interfaz de red.
- La *tabla de rutas* más simple que puede tener un *host* contiene solamente 2 *rutas*:
  - una *ruta* para especificar la *subred* en la que se encuentra el *host*
  - y la *ruta por defecto*
- **Ejemplo:**
  - un *host* con *IP* 10.6.122.40 y *máscara* 255.255.255.0 es un *host* que se encuentra en la *subred* 10.6.122.0 y que en esa *subred* debe haber un *gateway* que por lo general es 10.6.122.1.
  - 10.6.122.0 255.255.255.0 0.0.0.0 1
  - 0.0.0.0 0.0.0.0 10.6.122.1 1
  - esto se puede leer de la siguiente forma:
  - todos los *paquetes* que vayan a un *IP* de la *subred* 10.6.122.0 se envían directo
  - *todos los paquetes que no sean para la subred 10.6.122.0 se envían a 10.6.122.1 que el debe saber como llegar a ese otro IP.*

9. **Protocolo ICMP:** es un protocolo de capa de red, que usa la misma estructura de los *paquetes IP* pero con un *payload* de 1 byte. (*ICMP* viene de las siglas en inglés *Internet Control Message Protocol*)

- En el *paquete IP* hay un campo Protocolo que habíamos dejado en cero. A partir de ahora si el campo protocolo vale 1 (00000001) significa que nos encontramos con un *paquete IP* que contiene un *mensaje ICMP*.
- El campo *payload size* en el caso de los mensajes *ICMP* siempre va a ser 1.
- El *payload* va a ser un byte, y va a ser una de las siguientes opciones:
  - 0 → *echo reply* (la respuesta a un *Ping*, también conocido como *Pong*)
  - 3 → *destination host unreachable*
  - 8 → *echo request* (lo que conocemos como *Ping*)
  - 11 → *time exceeded* (tiene que ver con el campo *TTL* que todavía no estamos usando)

## Instrucciones

A continuación se muestran las posibles instrucciones adicionales que se incluyen en este 4to proyecto que puede tener el *script.txt*

- `<time> mac <host>[:<interface>] <mac address>`

- Esto es una **modificación** al comando.
  - La `<interface>` es opcional, si no se especifica, se asume por defecto que tiene como **valor uno**.
  - En caso de aparecer el `<interface>` será un número entre 1 y N (N = cantidad de puertos de red del dispositivo)
  - esto no se va a usar en el caso de las **PC**, pero se usará en los **routers**. las **PC** continuarán teniendo una sola interfaz de red y una sola **MAC**.
  - **ejemplo:** `10 mac pc1 A4B5`
  - **ejemplo:** `20 mac pc1:1 A4B5 # este caso es equivalente al anterior`
  - **ejemplo:** `30 mac pc1:2 D4F7 # significa que pc1 tiene 2 interfaces de red`
- `<time> ip <host>[:<interface>] <ip address> <mask>`
    - La `<interface>` es opcional, si no se especifica, se asume como valor cero.
    - En caso de aparecer el `<interface>` será un número entre 1 y N (N = cantidad de puertos de red del dispositivo).
    - La `<mask>` tiene un formato similar al **IP**
    - Este comando le asigna un par **IP/mask** a un dispositivo (**PC** o **router**)
    - **ejemplo:** `10 ip pc1 10.6.122.44 255.255.255.0`
    - **ejemplo:** `20 ip router1:1 10.6.122.1 255.255.0.0`
    - **ejemplo:** `30 ip router1:2 10.6.145.1 255.255.128.0`
  - `<time> send_packet <host> <ip destino> <data>`
    - con este comando se crea un **paquete IP** y se transmite
    - el **paquete** a enviar debe contener todos los campos especificados:
      - **IP** destino
      - **IP** origen
      - **TTL**
      - **Protocolo**
      - **tamaño**
      - **datos**
    - la **IP** de origen, el tamaño, etc. se rellenan a partir de la información que se tiene.
    - Los datos se especifican en formato hexadecimal
    - **NOTA:** el **packet** debe “meterse” dentro de un **frame** y transmitir el **frame**
    - **ejemplo:** `20 send_packet pc1 10.6.122.50 AAAABBBBCCCCDDDEEEFF1111...`
  - `<time> create router <name> <number-of-ports>`
    - Crea un **router** con esa cantidad de puertos
    - **Nota:** El **IP** de cada interfaz y las **MAC** se asignan con los comandos **ip** y **mac** respectivamente.
    - **ejemplo:** `10 create router rt 4`
  - `<time> ping <host> <ip-address>`
    - El **host** correspondiente envía **4 mensajes ICMP (ping)** al **IP** especificado.
    - Cada **ping** se envía **100 ms** después del anterior.
    - **ejemplo:** `30 ping pc1 10.10.10.10`
  - `<time> route reset <name>`
    - borra la **tabla de rutas** completa de un **router**.
  - `<time> route add <name> <destination> <mask> <gateway> <interface>`
    - adiciona una nueva **ruta** a la **tabla** del **router** especificado
    - **ejemplo:** `10 route add rt 10.6.122.0 255.255.255.0 10.6.122.1 1`

- `<time> route delete <name> <destination> <mask> <gateway> <interface>`
  - elimina una **ruta** a la **tabla** del **router** especificado.
  - **nota:** el comando `route delete` tiene que tener exactamente los mismos campos especificados en el comando `route add` de lo contrario el comando no hace nada.
  - **ejemplos:**
    - `10 route add rt 10.6.1.0 255.255.255.0 10.6.1.1 1 # se crea la ruta`
    - `10 route delete rt 10.6.1.0 255.255.255.0 # este no hace nada`
    - `10 route delete rt 10.6.1.0 255.255.255.0 10.6.1.1 3 # no hace nada`
    - `10 route delete rt 10.6.1.0 255.255.255.0 10.6.1.1 1 # este funciona`

## Transmisión de información.

- Cuando un **host** quiera transmitir un **packet** hacia otro **host** teniendo el **IP** de antemano. Si El **host** de origen no tiene la **MAC** del **host** de destino, debe realizar un **request ARP** (preguntarle a todo el mundo quien es el que tiene el **IP**) para obtener la **MAC** correspondiente y luego enviar el mensaje correspondiente.
- En la medida que la que sigamos subiendo de capas, cada capa inferior **encapsula** la información que llega de capas superiores.
- Aunque ya tengamos números **IP**, la comunicación entre 2 **computadoras** tiene que mantenerse como hasta ahora, tiene que conocerse las direcciones **MAC**, tiene que seguir habiendo colisiones (donde las haya), etc.
- Cuando un **host** reciba un mensaje **ping** (**paquete IP** con protocolo **ICMP** y **payload** = 8) debe responder un **pong** (**paquete IP** con protocolo **ICMP** y **payload** 0)
- Cuando un **host** recibe un **mensaje ICMP** con **payload** = 3 (*destination host unreachable*) debe detener el envío de paquetes hacia ese **IP** debido a que ese **IP** no se encuentra en la red y/o no es accesible por las rutas configuradas.
- A partir de ahora, cuando se haga un **send packet** ese **paquete** puede pasar por varios **routers**, en cada uno de ellos se evaluará la **tabla de rutas** correspondiente para determinar el próximo paso del **paquete**.

## Salida

- Además de todo lo que se escribe actualmente se deben crear ahora unos ficheros nuevos por cada computadora de la red. Este fichero tendrá el nombre de la computadora seguido de **\_payload.txt**
- En este fichero se escribirán solamente los datos recibidos por esa computadora (en la capa de red) 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 (todavía no hemos visto broadcast en la capa de red pero lo veremos en el próximo proyecto).
- El formato de salida debe tener en cada línea (separadas por espacio)
  - tiempo final de recepción de los datos
  - **IP** que envió los datos
  - datos
- En el fichero **\_payload.txt** se escribirán, en caso de que un **host** reciba un **mensaje ICMP**, escribirá como hasta ahora el **payload** (que es 1 byte) y a continuación el significado del **payload**.
  - Si recibe un 0 escribe todo lo de siempre y además un “echo reply”
  - Si recibe un 8 escribe “echo request”
  - ... y así con todos los demás **mensajes ICMP**
- **Ejemplo:** fichero **pc\_data.txt**
  - `100 10.6.122.45 A6F43400FFB34E...`

# Hasta el Momento

## Entregables

Archivo `steve_rodgers_311_natasha_romanov_312.zip` subido al EVEA que contenga:

- `README.txt` con información para compilar y ejecutar el programa
- [1] archivo `script.txt`
- [2] archivo `config.txt`
- [1] directorio `docs/`
  - [1] decisiones tomadas en la capa física
  - [2] decisiones tomadas en la capa de enlace
  - [2] algoritmos de detección y corrección de errores
  - [3] decisiones tomadas en la capa de red
  - [1] cualquier otra cosa que nos ayude a entender mejor el proyecto
- directorio `output/`
  - [1] un archivo `.txt` por cada *dispositivo*
  - [2] un archivo `_data.txt` por cada *host*
  - [3] un archivo `_payload.txt` por cada *host*

## Elementos Configurables

- [1] `signal_time` = <time in ms>
- [2] `error_detection` = <algorithm name>

## Elementos Físicos

- [1] *Cable*
- [1] *Computadora / Host*
- [1] *Concentrador / Hub*
- [2] *Cable Duplex*
- [2] *Conmutador / Switch*
- [3] *Enrutador / Router*

## Conceptos

- [2] Dirección *MAC* (Dirección *MAC* de *broadcast*)
- [2] *Trama / Frame*
- [2] Detección y Corrección de *Errores*
- [3] Dirección *IP* y *Máscara* (Dirección de *subred*, Dirección de *broadcast*)
- [3] Protocolo *ARP*
- [3] *Paquete IP / IP Packet*
- [3] *Tabla de rutas*
- [3] *Ruta*
- [3] Protocolo *ICMP*

## Instrucciones

- [1] <time> **create host** <name>
- [1] <time> **create hub** <name> <number-of-ports>
- [1] <time> **connect** <port1> <port2>
- [1] <time> **disconnect** <port>
- [1] <time> **send** <host-name> <data>
- [2] <time> **create switch** <name> <number-of-ports>
- [2] ~~<time> mac <host-name> <mac-address>~~
- [2] <time> **send\_frame** <host-name> <mac-address> <data>
- [3] <time> **mac** <host-name>[:<interface>] <mac-address>
- [3] <time> **ip** <host-name>[:<interface>] <ip-address> <mask>



- [3] <time> **send\_packet** <host-name> <ip-address> <data>
- [3] <time> **create router** <name> <number-of-ports>
- [3] <time> **ping** <host-name> <ip-address>
- [3] <time> **route reset** <name>
- [3] <time> **route add** <name> <destination> <mask> <gateway> <interface>
- [3] <time> **route delete** <name> <destination> <mask> <gateway> <interface>