**TEMA 0 - Repasito del sistema de capas y protocolo TCP**

LIBRO: Servicios en red, editorial gaceta (o garceta), el del profe es 2ª edición

El ejemplo de los pintores que no saben idioma y se comunican

La clave de las capas es la encapsulación. Cuando mandamos un mensaje por una app, eso va a la capa de transporte en forma de datos, y se le añade la cabecera (TCP) y una comprobación de errores. Esta capsula de transporte pasa a la capa de red, donde se convierte en una trama IP, contiene una nueva cabecera y comprobación, y un paquete de datos que tiene toooda la capsula de transporte. Esta trama IP pasa a la capa de enlace, donde vuelve a haber toda su parafernalia y la pestaña de datos, y finalmente pasamos a la capa física, donde tenemos la última parafernalia con todos los datos.

Esta trama física contiene TODO, es la que se manda y lo que recibe la otra parte, cuando esta trama es recibida, se va desencapsulando poco a poco, y finalmente, el usuario receptor, ve en su aplicación el mensaje final de aplicación. Todo lo demás está por detrás.

Capa física: Aquí se incluye la dirección MAC de origen y de destino.

Capa de red: Aquí es donde se incluyen las direcciones IP de origen y destino.

Capa de transporte: TCP, UDP, ARP, etc

VENTAJA: Cada capa es independiente, todo lo anterior, funciona por su cuenta, si tenemos que cambiar algo de una capa, solo se cambia eso, lo anterior funciona, es como el ejemplo del pintor, yo puedo cambiar al traductor, o el cable de telegrama, sin que se vea afectado todo lo demás.

Básicamente en cada capsula va información relativa a cada herramienta empleada, identificadores.

El 80% de las conexiones son de tipo TCP, mientras que un 10% son UDP, y el resto, lo demás.

A la hora de conectarse a internet, no basta solo con las ips, puesto que, al conectarnos, lo habitual es hacerlo desde redes privadas, que salen a internet como una única red pública (proveedor), cuando el servidor nos responde, el router proveedor, utilizando las tecnologías NAT (network address translate) y PAT (port address translate), se encargará de mandarnos a nosotros en específico esa respuesta.

Si recuerdas del año pasado, el router tiene una tabla de enrutamiento, donde tiene apuntadas las ips de los equipos conectados, sus macs, y los puertos de origen.

El puerto no es más que un identificador de un proceso, cada puerto es único.

Si recuerdas el ejemplo de Juan Carlos del año pasado, yo tengo 4 pestañas de Amazon abiertas, cada una de ellas es un proceso diferente, y tiene un puerto diferente, el servidor de Amazon está conectado a los 4 a la vez, pero por vías diferentes, de forma que cuando yo le hablo por una pestaña, Amazon me responde SOLO a esa pestaña, sin afectar a las demás, y cuando hablo por otra pestaña, me responde a esa pestaña, además de actualizarme las cosas que haya cambiado en la otra pestaña.

Cada puerto es independiente, para no verse afectado.

Tenemos los puertos bien conocidos (0-1023) y los puertos efímeros (aunque me gusta más temporales, pero bueno) 1024-65535 (2¹⁶). Los puertos conocidos son aquellos más frecuentes, que están configurados para ser siempre los mismos (apache, salir a internet, UDP, TCP, etc) (aunque si recuerdas a Fernando, lo odiamos cambiar a gusto). Los puertos efímeros.

Cuando hacemos una conexión, nosotros mandamos nuestra IP con el puerto asociado a ese proceso.

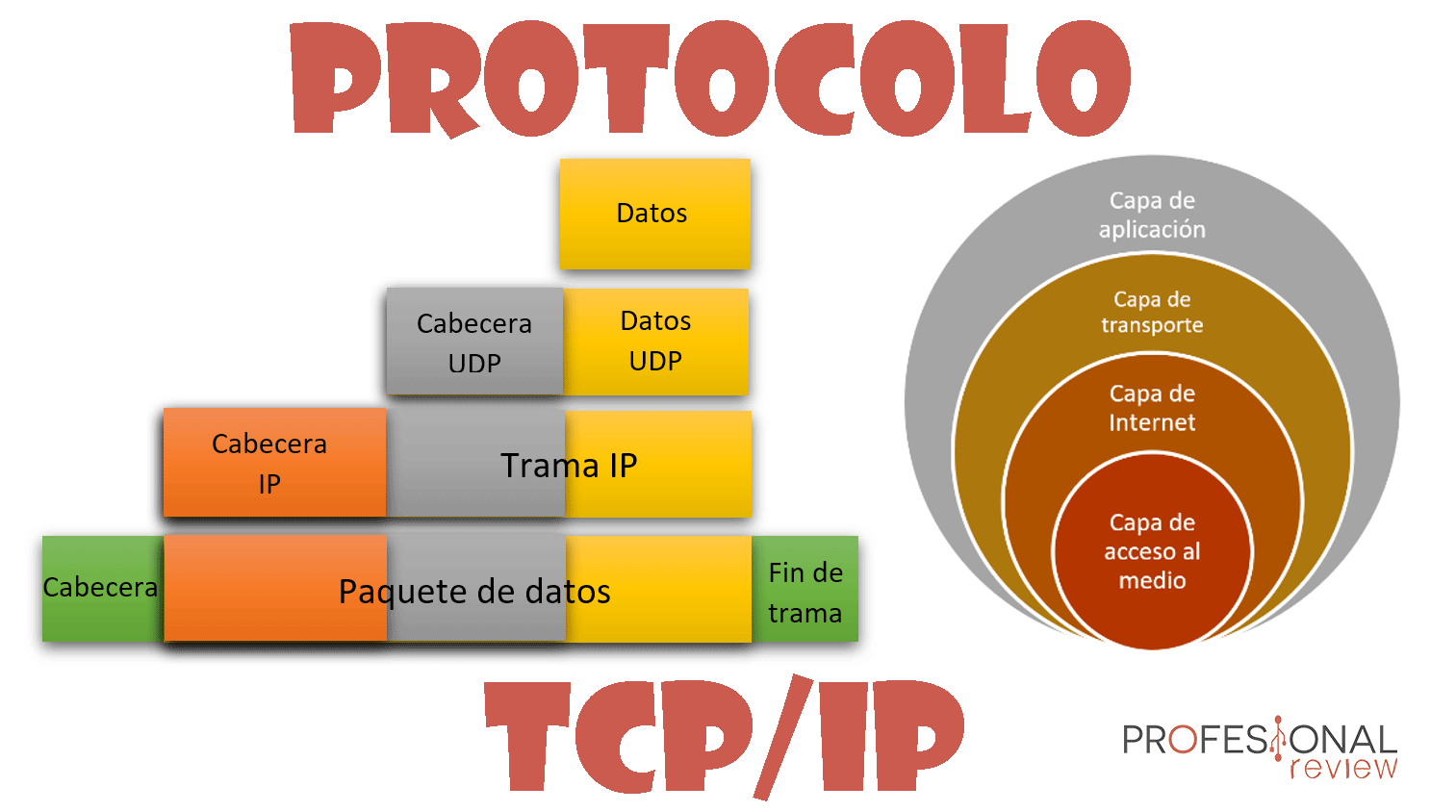
Yo tengo la APP opera, esa APP tiene 4 procesos abiertos, cada proceso tiene un ID y un puerto, pues eso es lo que se manda.

Todo esto es posible gracias a la multiplexación. Se realiza en la capa de transporte, significa que, cada nivel de transporte, puede ramificarse en múltiples vías, cada una con un destino. Y esto se repite por cada tipo de transporte.

Imagínate el ejemplo de Amazon, pero en forma de árbol, tengo todo el tronco (físico-enlace-red), y de ahí salen las ramas, que sería el transporte (TCP, UDP, etc). De cada rama salen otras ramas más pequeñas con sus hojas (opera y sus pestañas, steam con la actualización del CS, BNET con el wow abierto, etc).

NO CONFUNDIR que hay un máximo de 65536 puertos totales. Hay esos puertos PARA CADA RAMA DE TRANSPORTE.

No es lo mismo conectarme al puerto 5600 de TCP que al puerto 5600 de UDP. Van por ramas diferentes. Pero nunca habrá 2 procesos con el mismo puerto de la misma rama.



Capa física → mac → contiene 6 bytes hexadecimales(----:-----:-----:----:----:----) → si cada dato es un byte, se compone de 8 bits, es decir, 2 números/letras, la mac serán 12 símbolos (48 bits). Las MACs siempre son únicas.

Capa enlace datos → acceso al medio o LLC

Capa de red → IP

Capa transporte → Puerto

Capa aplicación

En la capa de transporte tenemos las PILAS TCP/IP y la UDP. TCP es orientado a la conexión y utiliza el 3 way handshake = yo me comunico, tú me respondes y yo te respondo, es decir, 3 pasos.

No son ni mejores ni peores, cada uno tiene diferente utilidad, uno es lento y seguro, otro es muy rápido, pero poco fiable.

Aunque las macs sean únicas, conocer todas las macs de todo el mundo seria horrible para poder establecer comunicaciones. De aquí surge el concepto de IP, es una forma de categorizar.

Yo agrupo muchas maquinas con sus macs en una misma red, cuando yo tenga que mandar algo, en vez de buscar ese usuario especifico, lo mandaremos a esa red 10.20.30.XY, y el router de esa red será quien busque la mac destino. Dicho de otra forma, se agiliza el trabajo de búsqueda y envío.

Igual que los números, 91 1234567, ese 91 indica que es de Madrid, mientras que 93 1234567, el 93 es de Barcelona.

Las Ips son conjuntos de 4 bytes (32 bits), y se dividen en 2 partes, La parte de RED y la parte de HOST. La parte de red indica pues eso, la red de ese equipo, mientras que la parte de host indica el n.º de equipos que podrían ser utilizados en esa red.

La máscara de red es lo que sirve para identificar que parte es la de red, y que parte es la host.

IP 192,168,131,250

Mascara 255,255,255,0

IP 10,20,30,40

Mascara 255,255,128,0

La red serán todos esos bits a 1, y la parte de hosts son los bits a 0.

255,0,0,0 → 1111 1111, 0000 0000, 0000 0000, 0000 0000

Recordando el año pasado, en la parte de host disponibles, el máximo de host que podemos tener por cada BYTE, es de 2⁸ -2, es decir, 255 menos la dirección de red y la dirección de broadcast.

Suponiendo red 192,168,131,X cuya mascara es 255,255,255,0

192,168,131,0 → red

192,168,131,1-254 → hosts útiles

192,168,131,255 → broadcast

Y todo esto ligado al subnetting.

Ejercicio rápido del profe: Tenemos 2 ips

192,168,1,20

192,168,2,20

Siendo la máscara 255,255,255,0

Las 2 ips pertenecen a REDES diferentes, no están conectados.

Ahora, cambiamos a mascara 255,255,254,0

x,x,0000 0001. 0001 0100

x.x.0000 0010. 0001 0100

254 = 1111 1110

LA parte de red de las IPS es diferente a la máscara, ergo las IPS no están en la misma red.

Si cabíamos la primera ip a 192,168,3,20

x.x. 0000 0011. 0001 0100

x.x. 0000 0010, 0001 0100

255,252,252,0 → 1111 1111, 1111 1111, 1111 1100, 0000 0000 /22

Ahora SI estarían en la misma red, pues su mascara coincide.

Dispositivos capa 2 → Switches → Me importa la MAC

Dispositivos capa 3 → Router → Me importa la IP

Cuando un Switch recibe tramas, solo se fija en las MACs, lo que haya por encima se la suda.

En cambio, el router se fija en las Ips, los puertos, y en última instancia la mac.

Cuando yo hago un ping entre 2 máquinas, si esas 2 máquinas están dentro de mi red, el ping ira hacia el switch y hacia la máquina, en cambio, si yo hago un ping a una maquina fuera de mi red, ese ping será mandado a la puerta de enlace (gateway), es decir el router, y saldrá fuera a buscar ese equipo en otra red.

A la hora de conectar un router con otro router, el circuito no es lineal como un cable entre 2 máquinas, sino que se dan muchos saltos entre diferentes routers hasta llegar al destino.

El router utilizando las métricas de salto (que vimos el año pasado), elegirá la ruta más óptima para llegar al destino (evitar saturación de red, o caminos de baja velocidad, interferencias, problemas, etc).

Volvemos a las capas.

Dentro de la capa de enlace-datos existe una subcapa conocida como capa de acceso al medio. Hace referencia a la conexión entre equipos.

Antaño se usaba el cable coaxial, que es un cable con 2 cables, uno de enviar y otro de recibir, como en la televisión.

¿Qué pasa? Que, si yo conecto 2 equipos con cable coaxial, ambos equipos tienen que enviar y recibir, ergo con el coaxial, habiendo solo un canal de envío, se provocan COLISIONES, los mensajes chocan entre sí.

De aquí surge el HALF dúplex → Es un cable con la misma función pero que solo permite que uno de los 2 equipos mande información. Como un walkie talkie.

Full dúplex supone la existencia de 2 cables (4 cables), de forma que ambos equipos pueden mandar y recibir a la vez, sin la existencia de colisiones.

El cable normal sería simple dúplex, o dúplex a secas.

A nivel inalámbrico, como el wifi, esto no funciona así, la antena solo puede ENVIAR O RECIBIR, no puede las 2 a la vez. De aquí surge CSMA/*CD y CSMA/*CA

CSMA /CA → Carrier sense multiple acces / colision avoidance

CD → Colision detection

Esto significa que 2 dispositivos conectados por wifi, cuando uno quiere comunicarse, primero se conecta a la antena y escucha el flujo, si no escucha nada, entonces manda señal.

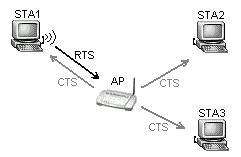
RTS → Request to send

CST → clear to send

Es decir, si yo quiero mandar algo, primero pido permiso al router para poder mandar algo, entonces el router, si está libre, manda un CST a todos, que les dice “hay alguien que va a mandar algo”, y se ponen a la escucha.

Entonces mando el flujo, y los equipos escuchan, y si quieren ellos responder, o mandar algo, hacen el mismo proceso.

Esto es lo que ocurre en dispositivos compartidos, de acceso al medio.



Por su parte CSMA/CD, es algo que hoy día ya no se usa, lo vimos el año pasado, básicamente es que el dispositivo manda información y escucha, si al mandar el mensaje ve que hay colisión, para de mandar información, se espera un tiempo y vuelve a mandar.

Recuerdo que había 2, uno que miraba cada X tiempo para mandar, y otro que estaba en espera hasta que estuviera libre el canal.

Si recuerdas un poco el año pasado, teníamos las frecuencias de 2.4 GHz y las de 5 GHz, las frecuencias están reguladas, no podemos emitir donde nos dé la gana, cada cosa tenía sus rangos con sus canales.

Las paredes frenan la luz, y la luz son longitudes de onda (ya lo vimos en percepción), entonces, las paredes también frenan las ondas wifi.

Cuanto más pequeña es la onda, más se parece a una luz, y, por tanto, más problemas dará para atravesar paredes y demás, es por ello que se desarrollaron las ondas de 5GHzs, para tener más espectro disponible, y así poder emitir reduciendo las interferencias. Además, al tener mucho más espectro, se desatura la red. Hoy día los routers ya vienen con opción para elegir la red wifi de 2.4 o la de 5.

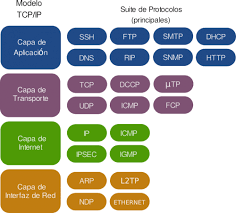
Antiguamente, los routers podían tener 2 antenas para emitir, esto era porque, cuando se conecta con el Pc, podía ocurrir desfases y por tanto recibir paquetes con errores o incompletos, el PC le dice al router que le ha llegado algo, varios paquetes, que contienen lo mismo, y le responde.

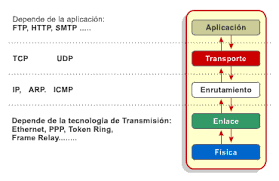
Si ese router manda paquetes por 2 antenas, el PC recibe paquetes de ambas, y le dice al router, qué paquete ha llegado con menos errores, entonces el router aprende qué antena manda mejor los paquetes a ese ordenador, y cuando le mande algo, lo hará solo por esa antena.

Capa de aplicación: Protocolos →Http, dhcp, imap, etc (hyper text transfer protocolo (el https sería con Secure)) DHCP (dynamic host configuration transfer, vamos, el protocolo que te asigna direcciones ip, te da la puerta de enlace, acceso a la dns, etc), IMAC (internet mesasge acces protocolo, servidores de correo).

Nos ha contado que Google, a la hora de posicionar en su buscador las páginas, tiene tier list, cuantos más puntos tengas, más adelante sales. 2 de los criterios que suman puntos son: tener el SECURE (https) y otro ser dinámico, adaptable a móviles, tablets y demás (resolve en inglés).

Capa de transporte: Protocolos → TCP ( transmisión control protocol, 80% del tráfico, orientado a la conexión, lento, pero con mucha fiabilidad y seguridad, con el 3 way handshake), UDP (user datagram protocol, 10%, sin conexión, muy rápido pero sin fiabilidad)

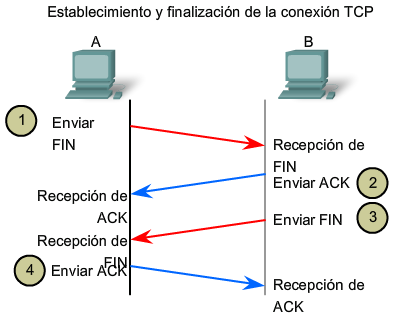




Al hacer una conexión TCP y mandar trazas, yo le mando una trama con 10 letras, y el otro las recibe, me responde, espero la letra 11, lo recibo, veo que todo perfecto y le mando las siguientes.

Si me dice espera otra letra, veo que algo va mal, se ha perdido algo, así que le vuelvo a mandar lo anterior para que tenga todo.

Esta comprobación de errores se incluye en las cabeceras TCP que se mandan.

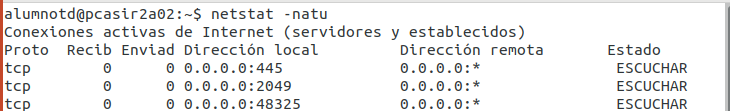


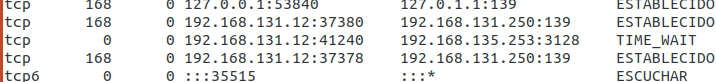
En la cabecera UDP hay un puerto de origen y una comprobación, pero son opcionales, normalmente no se usan, pues no va a haber ni conexión ni comprobación, ergo sobran, si es necesario se vuelve a mandar todo de nuevo.

Entonces, según lo que se vaya a utilizar, se construirán las aplicaciones basándose en TCP o UDP, u otras.

Con el comando netstat -natu podremos ver nuestras ips con sus puertos correspondientes, esto son los procesos que se están utilizando con conexión, y nos indica a que dirección remota y que puerto se conectan, además también indica el estado del mismo.

El puerto 443 es el habitual de HTTPS





En este caso, todas nuestras conexiones son hacia el proxy, por lo que no van a salir las direcciones destino, si no que los paquetes HTTP irán dirigidos al proxy, y ahí entonces SI que se leen las diferentes capas, a diferencia de las conexiones directas.

Las conexiones pasan por diferentes estados. Las conexiones siempre mueren, pero no instantáneamente, por si acaso hubiera que seguir conectado (las tenemos en el pdf del aula).

Cuando tú haces log out, gráficamente lo haces, pero a nivel de conexión, sigues conectado un tiempo, hasta que finalmente te desconectas.

El estado Syn sent es básicamente esperando conexión, sincronizando, yo mando una petición a un sitio y espero a que me responda. Si no me responde, se caduca la petición y es cuando a nosotros nos saldría la pestaña de no se puede establecer conexión, y un emoji con cara triste :(

Abrimos packet tracer y hacemos algo sencillito: pc, switch y servidor, cada uno con sus ips y el server con servidor web activo y tal.

Entonces, entramos al PC y nos metemos en el navegador, nos conectamos al servidor (vía ip).

Si abrimos la simulación, veremos todo el proceso de las tramas icmp, arp, tcp y finalmente http.

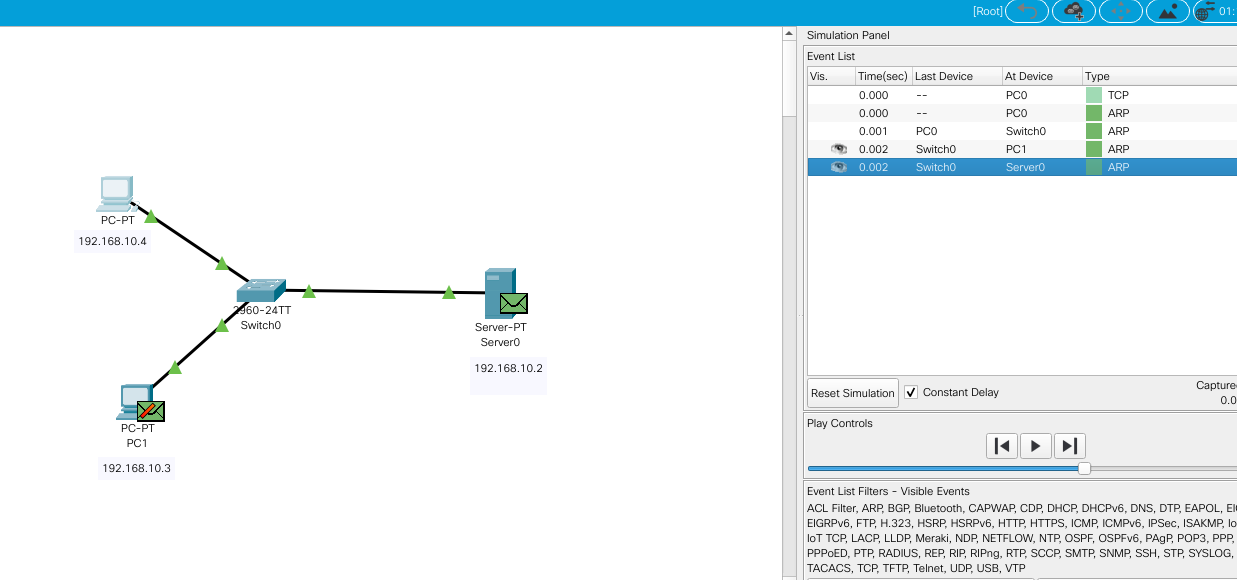
Se genera una trama TCP, pero como está incompleta porque no se conoce el destino, se genera un ARP, este ARP se manda por broadcast por el switch, para preguntar QUIEN de todos los conectados, tiene la IP destino. Entonces el equipo implicado le responderá y se completa el ARP.

Una vez tenemos la MAC, se completa la TCP y se manda, podemos abrir la trama para leer la cabecera TCP.

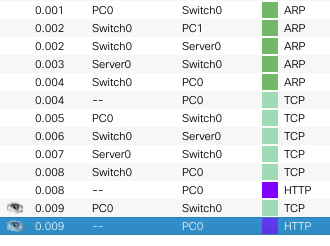
Cada paso del a trama TCP representa un paso del 3 way handshake, podemos mirarlo en las flags, cada flag representa lo que se hace, por ejemplo, el servidor responde con un 10001 en las flags, eso significa que tiene las flags de ACK y de FIN.

Terminado todo el proceso TCP, el PC genera el datagrama HTTP, se da otra vuelta de cosas y finalmente se abre la web del servidor en la aplicación del pc.

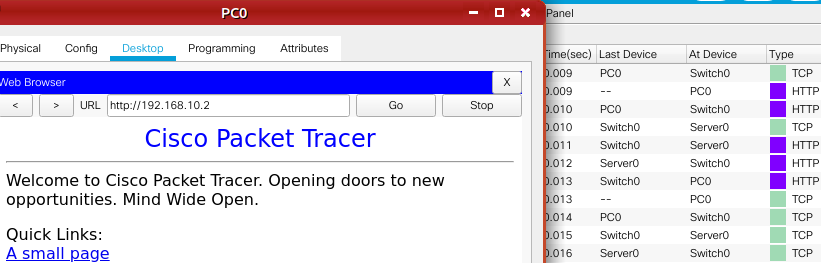
Montamos un PC extra en el escenario, y empezamos de 0. Vamos a comprobar que realmente, cuando el paquete ARP llega al switch, se envía un Broadcast, ya que con 1 solo equipo difícil.



Aquí se ve que ha llegado al switch y se ha mandado a cada dispositivo conectado, como el Pc no es el destino, descarta el paquete, mientras que el servidor se lo queda, y procede a responder con todo el proceso.



Aquí se ve en la TL que efectivamente, lo recibieron todos, pero luego solo actúa el servidor, y solo se comunica el servidor y funciona la web.



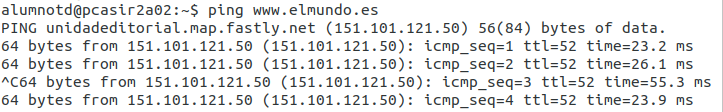
El comando ping sirve para comunicarse entre dispositivos. Es un protocolo de tipo ICMP (internet control message protocol).

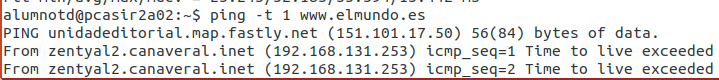
Va dentro de una capsula IP, su cabecera es muy simple, tipo, código y checksum, porque lo usamos cuando ya se conocen origen y destino.

Es un mensaje que se resume en: Request (te llamo) y answer (me respondes).

Este mensaje tiene un TTL (Time to Live), es el parámetro que indica la vida restante de un paquete, el n.º de saltos que puede dar una trama antes de ser desechada. Si un paquete termina su TTL, se considera como una no respuesta, y yo recibo el mensaje de que no hay respuesta.

Podemos mirar el TTL usando traceroute o traceert





Como se ve en las imágenes, podemos manipular el TTL con el comando ping, y poner cuantos saltos queremos que dé como máximo.

Usando el comando de traceroute, podemos ver información detalla de donde está el destino y cuantos saltos serán necesarios dar para llegar.

Ahora pasamos a un ejercicio de packet tracer, hay que subirlo luego.