TEMA 1: INTRODUCCION

¿Qué es internet? 1.1.1 Definición de Internet. Podemos ver a internet como una red de disp ositivos (también llamados host o sistemas terminales) conectados entre sí, es decir, como un conjunto de ordenadores, tablets, móviles, etc…
Que forman una red. También podemos verlo como una red de servicios, con servicios nos ref erimos a aplicaciones que tienen algún uso, como whatssap, el correo electrónico... Por último , también podemos verlo como una red de redes, en la cual podemos tener muchos dispositivo s conectados entre si y que a su vez se conectan, a través del router, a internet.
Estas conexiones se hacen gracias a los ISP. El ISP es el proveedor de servicios de internet

Existe un término que se ha popularizado últimamente, este es el internet de las cosas (IOT), y se refiere a la conexión establecida por cualquier dispositivo a través de internet en cualquier lugar, tiempo y por cualquiera. Esto nos da una gran cantidad de datos en nuestra red. Un eje mplo sería el conectar una tostadora o lavadora a la red de casa para poder utilizarlo mediante un ordenador.

1.1.2 ¿Qué es un protocolo? Un protocolo de red define todas aquellas normas que definen el formato de los datos que enviamos para poder entenderlos. Un protocolo define el formato d el mensaje, el orden en el que enviamos los mensajes, y las acciones que podemos tomar al re cibirlo. Los protocolos más conocidos son: TCP/IP, HTTP, SMTP,
Ethernet, Wi-Fi... Algunos organismos involucrados con protocolos son, IETF,
IEEE,ISO, ITU, Internet Society. 1.2 Estructura de la red. La red se puede dividir en tres partes f undamentales. La frontera de red, las redes de acceso y medios físicos y el núcleo de la red. 1. 2.1 La frontera de la red. La frontera de la red la componen los distintos sistemas terminales, t anto servidores como clientes, que se conecten a ella. Según la conexión entre ellos tenemos:

•Arquitectura cliente-servidor:
Unos terminales se conectan a otro llamado servidor que es el que proporciona la información o aplicaciones que necesitamos.

• Arquitectura peer-peer:
Se conectan distintos sistemas terminales comparten información como si fuesen cliente y se rvidor al mismo tiempo.

1.2.3 Red de acceso y medios físicos. Las redes de acceso y medios físicos tienen como objetiv o conectar la frontera de red con el núcleo. Hay diferentes modos de hacerlo: •
Acceso telefónico (Modem):
Es el método más antiguo. Para que funcionase era necesario conectar un ordenador, a través de un modem, a un cable del teléfono, es decir, se utilizaba la red de telefonía, lo que impedía la conexión a Internet y el uso del teléfono simultáneamente. La velocidad era de 128 Kbps en RDSI y de 56 Kbps en acceso directo a router.

• Digital Subscriber Line (DSL):
Utilizaba la línea telefónica, pero contaba con unos microfliters o spliters que permitían dividi r las señales e identificar por donde debían de ir. Los multiplexores de las oficinas centrales er an los que lo enviaban a un sitio y otro. Es una de las mejores opciones porque no compartes r

ed con otros usuarios, es decir, es dedicada hasta la estación telefónica .La velocidad oscilaba entre 52 Mbps de bajada y 16 Mbps de subida, aunque dependía de la distancia a la central y l a calidad del cable.

• Acceso por cable:
No se utiliza la estructura telefónica, si no que se utiliza el cable de la televisión para llevar int ernet a casa, lo que es lo mismo, el cable coaxial, el más habitual es el cable híbrido(Hybrid Fib er Coax). La velocidad depende del número de usuarios conectados, ya que es un acceso comp artido, a la red. Normalmente HFC es fibra desde la central al edificio pero luego a cada casa ll egan cables de HFC. Conexión asimétrica con velocidad de bajada de 30, 50 Mbps y de subida d e 2 Mbps.

• Tecnología FTTH (Fiber to the Home):
Se lleva fibra óptica desde una central hasta unos spliters que dividen la fibra óptica para que l legue a las respectivas casas. La velocidad llega hasta los 100 Mbps simétricos, de subida y baja da, y permiten que llegue el Triple-Play (tv, internet y telf). Hay dos tipos:

- Passive Optical Network (PON): más usada.
- Active Optical Network (AON).

• Acceso por Ethernet:
Utilizado en empresas, universidades, etc. La velocidad es de 10 Mbps a 10 Gbps. Los hosts se conectan a un switch ethernet. Es todo cable Ethernet. Esto no significa que todos tengamos Et hernet, nosotros tendremos ADSL, pero un cable Ethernet para conectarlo desde el router al p c.

•Acceso inalámbrico:
Esta red se hace desde un router hasta cualquier host. Puede ser una conexión LAN (redes de á rea local): Wi-fi con velocidades de incluso 300 Mbps, o WAN: 3G o WiMax. Como medios físic os por los que se transmiten los datos tenemos:

• Cable de par trenzado:
Son dos cables de cobre aislados y enrollados en espiral. El RJ45 tiene cuatro de estos cables. P ueden estar clasificados en dos categorías:

- UTP CAT 5: 1 Gbps distancia de 100 metros máximo.
- UTP CAT 6: 10 Gbps distancia de 100 metros máximo.

• Cable coaxial:

Se basa en dos conductores de cobre concéntricos. Tenemos un conductor en el centro, un aisl ante en el medio, y un conductor encima del aislante. Todo esto está recubierto por plástico. Hay dos tipos:

- Baseband: Solo un canal(digital) de 5 ohmios.
- Broadband: Varios canales(analógicos) de 75 ohmios.

• Fibra óptica:
Es fibra de vidrio que transporta pulsos de luz(bit) a grandes velocidades (10-100 Gbps). Tiene r atios de error bajos, lo que nos permite enviar señales sin repetidores y señales inmunes al rui do. Se utilizan, sobre todo, para enlaces de larga distancia, como transoceánicos.

• Enlaces radio:

◦ LAN (Ej: WIFI, 54 Mbps)

◦ WAN (Ej: 3g 1 Mbps)

◦ Microondas terrestres (vel: 45 Mpbs)

◦Satélite (vel: 45 Mbps y 270 ms de retardo)

1.2.3 Núcleo de la red. El núcleo de la red se define como todos aquellos routers y switches q ue están por detrás y que comunican los mensajes que enviamos entre sistemas terminales. Es ta comunicación puede hacerse mediante: • Conmutación de circuitos (CC):
En ella se reservan una serie de recursos antes de enviar la información. Las ventajas son que tenemos recursos guardados y las desventajas que esos recursos cuestan y que si no los utiliza mos se pierden. Otro problema que presenta es que no tenemos todas las conexiones posibles entre los distintos terminales.

Ej: No hay una conexión directa entre PC 1 y PC 4. Solución ir de PC1, a PC2 y a PC4, esto es lo que llamamos multiplexación Para solucionarlo se comparten enlaces mediante multiplexación de señal. Podemos utilizarlo de dos formas:

• FDM (Multiplexación de frecuencia):
Dividimos la frecuencia para que los usuarios tengan una franja de tiempo para que accedan a i nternet. Siempre se reserva, aunque no se use.

• TDM (Multiplexación de paquetes):
Damos a cada usuario un periodo de tiempo para utilizar toda la frecuencia. Cuando termina el slot del primero pasa al siguiente, y así sucesivamente.

La velocidad en FDM y TDM es la misma.

• Conmutación de paquetes (CP):
Aquí se divide la información en paquetes más pequeños sin necesidad de reservar recursos. A horramos costes, pero hay un problema fundamental y es que, a veces, habrá congestión porq ue se quieren enviar mucho al mismo tiempo. La compartición del enlace se denomina multiplexación estadística y consiste en que los paquetes no siguen un orden ni tiempo específ ico,no sabemos cada cuando se envían los paquetes, por ello comparten el ancho de banda baj o demanda. Es decir, si un host A y un host B tienen que enviar un paquete lo envían, y así se cr ea una cola de paquetes.

¿Entonces cual es mejor?
¿Conmutación de circuitos o conmutación de paquetes? CP admite un mayor número de usua rios que CC, pero esto no significa que sea siempre el ganador absoluto, ya que, puede haber u na congestión de paquetes. Aun así, cuando son ráfagas de paquetes de los que no conocemos el patrón si es recomendable utilizar CP. Sin embargo, para aplicaciones de audio y vídeo mejo r CC.

Los protocolos necesitan transferencias de datos fiables y controles de congestión, por tanto, muchas veces no es viable utilizar al conmutación de paquetes. Podríamos imitar el comporta miento de los CC, pero es un problema todavía no resuelto debido a su gran complejidad. 1.2 .4 Estructura de la red. Anteriormente, definimos Internet como una red de redes. Esta red de

redes presenta una jerarquía de niveles en la que podemos observar la red troncal de Internet, que son los llamados ISP. Existen tres tipos de ISP:

• ISP de nivel 1:
Dan velocidades muy grandes (2-10 Gbps). Son nacionales o internacionales. Se tratan como ig uales porque se prestan sus servicios unas a otras y no compran servicios a otros. Se conectan mediante POP. Están en esta lista los que más cobertura cubren. Algunos ISPs de nivel 1 son: L evel 3, Congent, Tinet...

• ISP de nivel 2:
Son redes más pequeñas que los anteriores, pueden ser nacionales o regionales. Se conectan a los de nivel 1 pagando por el servicio que este les ofrece o se conectan a otros ISPs de nivel 2 c on conexiones privadas. Ej: Telefónica.

• ISP de nivel 3:
Son locales y dan acceso directamente a los usuarios para conectarse a Internet. Se conectan a ISPs de nivel 2 para obtener el servicio deseado. También pueden conectarse a ISPs de nivel 1 pero es más raro. Un paquete va a pasar por muchas redes distintas que están interconectada s.

1.3 Retardos, pérdidas y tasa de transferencia en las redes.

1.3.1 Retardos y pérdidas La información se divide en diferentes paquetes de información para poder transmitirse desde un host inicial a uno final.
Un paquete pasa por muchas redes y en el envío sufre diversos retardos. Existen cuatro tipos d e retardos:

• Retardo de procesamiento (o de procesado en el nodo):
Comprueba que el paquete está bien y tiene que decidir por dónde hay que enviarlo. Su retar do es de microsegundos.

• Retardo de cola:
Es el más complejo de calcular. Determina el tiempo que tardan los paquetes desde que llegan al nodo hasta que se les sitúa en la salida. Depende de la congestión del router, es decir, de la c antidad de paquetes que haya. El retardo suele ser de microsegundos a milisegundos. Por si ac aso tenemos que calcular este retardo en algún momento, hay que tener en cuenta, a parte qu e depende de la intensidad del tráfico(I), la siguiente gráfica.

◦Si I=La/R=0 es un retardo pequeño.

◦Si I=La/R<=1 es un gran retardo.

◦Si I=La/R=1 el retardo es infinito.

L=Longitud del paquete (bits/paq) a=Tasa promedio de paquetes(paq/s) R=AB del enlace(b ps)

Los paquetes se pierden porque el buffer(cola en el nodo) tiene un determinado tamaño, y cua ndo este se llena van desapareciendo. Hay que intentar garantizar que llegan todos los paquet es. Ya veremos como hacerlo.

Este retardo también es infinito, aunque su valor sea un poco inferior a 1.

• Retardo de transmisión:
Se define como el tiempo requerido para poder meter los bits del paquete por el enlace de sali da. Esto depende del ancho de banda del enlace de salida (R (bps)) y de la longitud del paquete (L(bits)). Si el enlace de salida es estrecho o el paquete es largo tarda más. El tiempo requerido se calcula como T = L/R. Suele tardar entre milisegundos y microsegundos.

•Retardo de propagación:

Tiempo que tarda en viajar de un nodo al siguiente. Dependerá de la longitud entre los dos no dos (S) y la velocidad de propagación (V) que suele estar entre los 2*10^8m/s. Tarda entre los milisegundos y los microsegundos. La fórmula para calcular el tiempo total es T=s/v. Cada paq uete sufre un retardo determinado en esta ecuación: El retardo de cola y de procesamiento suele ser 0.

1.3.2 Tasa de transferencia.

La tasa de transferencia es la ratio(velocidad) a la que se transmiten los bits entre emisor y rec eptor. Se mide en bits/unidad de tiempo. Hay dos tipos:

• Instantánea: Ratio en un determinado momento.

• Medio: Ratio sobre el tiempo de transmisión.

Si un servidor envia un numero de bits por un canal, dependiendo de la velocidad de es e canal, que sería en este caso Rs(velocidad del servidor) bps, podría enviar R bits cada segundo, el cliente podrá recibir por un canal que tiene una capacidad de Rc(velocidad del cliente) bps. Existen también una tasa media.

• Si Rs<Rc, Rs es más estrecho y determinará la velocidad a la que podrá descargar.

• Si Rc<Rs, Rc es más estrecho y determinará la velocidad de descarga.

Es más habitual. Normalmente es Rc el que determina la velocidad, pero también pue de limitar Rs si le llegan muchas peticiones al servidor. Como conclusión diremos que e l que determina la velocidad siempre será el más estrecho. Actua como cuello de botell a.

1.5 Capas de protocolo.

Debido a la gran cantidad de elementos que encontramos en una red, como host, servidores, protocolos, aplicaciones, etc, surgió hace tiempo la idea de organizar la complejidad de las red es. Es por eso que crearon las capas de protocolo. Cada capa se encarga de una función que le transmitirá la información obtenida a otra, así juntando todas tendremos el protocolo. Se orga nizan en capas para simplificar el proceso.

Además de para facilitar el mantenimiento y actualizar el sistema de forma sencilla. De manera que podemos cambiar una capa sin afectar a todo. Las ventajas que nos ofrece agrupar en ca pa es controlar sistemas grandes de forma simplificada. Además de una modularización que fa cilita el mantenimiento y actualización del sistema. Los inconvenientes en las capas son que: p odemos duplicar la funcionalidad de estas o que no podríamos cumplir el principio de organiza ción de estas. Puede darse también que en una capa necesitemos información que esté en otr a. Internet se organiza en lo que se llama pila de protocolos de Internet
y está compuesto por cinco capas:

• Capa de aplicación: También se denomina capa 5 o capa superior.
Soporta aplicaciones de red que intercambian mensajes. Tenemos aplicaciones como el naveg ador que encía menajes al servidor web. Añade el mensaje. Tenemos FTP, HTTP. EJ: Navegado r que envía mensajes al servidor web.

• Capa de transporte:
Es la capa 4. Es la encargada de transportar el mensaje de aplicación a los puntos terminales. A ñade el segmento. Hay dos UDP, TCP.

• Capa de red: Es la tercera capa, o capa central, y envía los paquetes de la capa de red, conocidos como datagramas, de un host a otro. Utiliza el protocolo IP.

• Capa de enlace:
Se transfieren las tramas entre elementos de red contiguos. Es la capa 2. Tenemos FPP, Ethern et, Wifi. Añade la trama.

• Capa física:
Mueve los bits en el cable. Es la primera capa y, por tanto, la más básica. A veces la primera y l a segunda capa se unen formando una única capa. Esta división de capas se basa en el modelo de referenncia OSI, propuesto por ISO en 1980. En este modelo aparecían 7 capas, es decir, d os más a parte de las ya conocidas, y eran:

• Presentación:
Se dedicaba a interpretar los datos por parte de las aplicaciones. Ej: Encriptar.

• Sesión:

Delimita y sincroniza el intercambio de datos buscando puntos de restauración, esquemas de r ecuperación. Si necesitamos en Internet, interpretar los datos o mantener datos entre sesione s lo hacemos con la capa de aplicación. También se pueden interpretar capas intermedias.

1.5.1 Pasando información de un host a otro host.

Tienen que ser los mismos protocolos en las capas, es decir, el tipo de la capa de aplicación del primer host tiene
que coincidir con la del segundo host. Los datos que se pasan de una capa a otra se tienen que encapsular y desencapsular para poder enviarse. Así se encapsulan: Así se desencapsula:

Las capas al ser encapsuladas tienen que ser las mismas, pero ¿qué pasa si hacemos una petici ón mediante Wi-fi y lo recibe un servidor conectado mediante Ethernet? La respuesta es sencill a, para ello tenemos el router que se encarga de modificar la información.

TEMA 2. CAPA DE APLICACIÓN.

Es la capa más cercana al usuario y, por tanto, una de las más fáciles de comprender.

2.1. Principios de las aplicaciones de red.

Las aplicaciones de red
son programas que se ejecutan en distintos sistemas terminales y que se comunican entre sí a través de una red. Como un navegador que se quiere ejecutar en un cliente, y que se quiere conectar con un servidor web. Gracias a lo que se ejecuta en la capa de aplicación podemos enviarnos distintos mensajes. Aunque nos comuniquemos a través del núcleo de la red no es necesario programar nada para poder navegar por ahí. Así, es mucho más sencillo programar aplicaciones de red, ya que meterse a investigar sobre como funciona el núcleo de la red si queremos hacer una pequeña aplicación sería algo muy complejo. Las aplicaciones de red más extendidas son whatssap, line, etc.

2.1.1 Arquitecturas de las aplicaciones de red.

La arquitectura de la red es fija y proporciona un conjunto específico de servidores a las aplicaciones. Existen varios tipos de arquitecturas.

2.1.1.1 Arquitectura clienteservidor:

Tenemos unos sistemas terminales, servidores, que están siempre disponibles, es decir, tienen una alta disponibilidad. Estos servidores suelen tener una dirección IP fija. Además, suelen estar configurados de forma redundante en granjas de servidores para aumentar sus servicios. Por otro lado los clientes son sistemas terminales que no están siempre disponibles, yo voy a poder encender o apagar el cliente, van a tener una dirección IP variable, y no van a conectarse entre sí máquinas cliente.

2.1.1.2 Arquitectura peer-peer

En esta arquitectura la aplicación explora diferentes medios de comunicación directa entre parejas de host conectados de forma intermitente, con una IP variable, y que pueden ser encendidos y apagados cuando se quiera. Estos sistemas terminales son conocidos como peers (pares). Este método tiene una alta escalabilidad, una difícil gestión, problemas para incentivar contenidos, problemas de seguridad. Además, la red está orientada a ISP y no a peer-peer, lo que hace complicado su uso. Algunos programas que usan esta arquitectura son BitTorrent, eMule, etc.

2.1.1.3 Arquitecturas híbridas.

Los sistemas terminales se conectan mediante peer-peer, pero, hay un servidor que gestiona y controla para que todos los usuarios se conecten correctamente. Ej: Skype, programas de mensajería instantánea.

2.1.1.4 Nube.

Se parece a una arquitectura cliente- servidor, pero la gran diferencia es que en vez de utilizar un gran servidor, se utiliza la nube para almacenar los datos que los clientes quieran utilizar.

2.1.3 Comunicación de procesos.

Un proceso
es un programa que se ejecuta dentro de un sistema terminal. Un proceso emisor crea y envía

mensajes, mientras que, un proceso receptor, recibe y responde los mensajes. Los procesos se comunican entre sí utilizando la capa de aplicación.

Los procesos dentro del mismo host se denominan comunicación interproceso están definidas por el sistema operativo. Pero, en este caso, nos centramos en comunicación entre distintos host, es decir, comunicación de red .

En la comunicación de red el que inicia la comunicación es un proceso cliente y el que espera a que alguien se conecta es unproceso servidor.
Los procesos no se comunican directamente con otros procesos, sino que se hace mediante unos procesos denominados socket.

Un socket:
es una puerta a unas capas inferiores, y es el encargado de transmitir los mensajes entre estas. Es la puerta de la capa de aplicación y permite un control total sobre los elementos de dicha capa pero no sobre la capa de transporte que está justo debajo. Un protocolo
define el tipo, sintaxis y semántica del mensaje. Además de las reglas decuándoo (orden) y cómo (acciones) un proceso se envía y se recibe. Los protocolos pueden ser:

•De dominio público: Conocidos por todos porque están publicados y definidos por RFC. Podemos crear aplicaciones basados en estos protocolos. Permiten interoperar entre unas aplicaciones y otras que empleen el mismo protocolo.

•De dominios propietarios: No son conocidos y no puedo hacer una aplicación que implemente este protocolo.

2.1.4 Servicios de transporte disponibles.

Los servicios de transporte se clasifican según cuatro parámetros:

2.1.4.1 Transferencia de datos fiable

Para que todos los datos enviados desde un terminal a otro lleguen completos. Las aplicaciones tolerantes a fallos(videollamadas, audio) no son tan estrictas como otras aplicaciones que si necesitan que todos los datos lleguen correctamente.

2.1.4.2 Tasa de transferencia

Tasa a la que el proceso emisor puede suministrar bits al proceso de recepción. Existen aplicaciones que necesitan reservar una tasa mínima garantizada a una cierta velocidad para que funcionen correctamente, son las llamadas aplicaciones sensibles al ancho de banda. Hay otras, llamadas elásticas
, que usan la tasa disponible en cada momento, a las que no les importa la velocidad ya que se adaptaran a la que haya.

2.1.4.3 Temporización(timing)
Necesario para telefonías por internet, juegos en red, y , en definitiva, aplicaciones que no pueden tener grandes retardos porque sino no serían útiles. También existen aplicaciones que necesiten retardos para su correcto funcionamiento.

2.1.4.4 Seguridad

Los servicios de transporte me van a ofrecer diversas opciones de seguridad para garantizar la confidencialidad. integridad de datos... Estos son algunos ejemplos.

2.1.5 Servicios de transporte de Internet.

Internet proporciona dos protocolos de transporte: UDP y TCP. Cada uno ofrece un conjunto d e servicios diferente.

2.1.5.1 Protocolo TCP.

Es un servicio orientado a la conexión, similar a una conmutación de circuitos porque establece una conexión entre dos sistemas. Además, es un servicio de transferencia de datos fiable, no pierde paquetes, e incluye un mecanismo de control de congestión y un control de flujo que beneficia el funcionamiento en internet. TCP no proporciona temporización, ni mínima tasa de transferencia, ni seguridad, esto tendríamos que proporcionárselo en una capa de aplicación o en una capa intermedia.

2.1.5.2 Protocolo UDP.

Es un protocolo ligero y simple, no orientado a conexión, ya que los paquetes se envían según se reciben, y no proporciona una transferencia de datos fiable, porque si se pierden los paquet es no tiene porque volver a enviarlos. No proporciona ni control de flujo, ni control de congesti ón, ni temporización, ni tasa de transferencia mínima, ni seguridad.

Es decir, no proporciona casi ningún servicio, solo coge los paquetes que le manda la capa de a plicación y enviarlos a destino.

Casi todas las aplicaciones, siempre y cuando no sean de audio y vídeo que utilizan UDP, usan el protocolo TCP.

2.1.6 Direccionamiento de procesos.

Para poder identificar los procesos corriendo en una máquina, utilizamos los números de puerto. Las máquinas las identificamos por su dirección IP (magnitud de 32 bits), que es el nombre o dirección del host. Cuando diseñamos una aplicació n tenemos que asignarle un número de puerto. Para comprender mejor estos dos conceptos, podemos compararlo con una casa, que tiene una dirección (IP), en la que viven personas (que son los números de puerto).

2.2 Web y HTTP

Vamos a tener un proceso cliente, que va a ser el navegador, y un proceso servidor, que va a s er la web que atiende las peticiones del cliente. El navegador hace una petición al servidor con una URL que tiene la siguiente estructura: protocolo, máquina al que hace la petición(host) y r uta del documento al que se quiere acceder. El servidor cuando recibe esta petición responde al cliente devolviendo ese documento. Es un documento HTML que hará referencia a otros mu chos objetos. Estos objetos, cuando recibe el navegador la página html, los irá solicitando a los servidores en los que estén alojados, que pueden ser distintos al anterior. El protocolo que s e encarga de esto es el http (HyperText Transfer Protocol). Está basado en una arquitectura cliente-servidor en el cual tend remos distintas máquinas que pueden estar corriendo diferentes sistemas operativos. En el ten emos procesos clientes, navegadores, y procesos servidor, que son los servidores, esos navega dores hacen las solicitudes http. Este protocolo está descrito en las RFCs 1945 y 2616, el vigent e actualmente, y la nueva que es 7230-40. Utiliza TCP como protocolo de transporte y no tiene memoria de estado. El que utilice TCP quiere decir que el cliente crea un socket e inicia una co nexión TCP con el servidor en el puerto 80. Este servidor aceptará a conexión TCP y se interca

mbiarán los mensajes entre el servidor y el navegador,
posteriormente, esa conexión se cierra. El que http no tenga memoria significa que el servidor no va a mantener la información de peticiones hechas por el cliente anteriormente, esto se sol uciona mediante cookies. Los protocolos con memoria son complejos. Vamos a tener dos tipo s de conexiones http:

• Conexiones HTTP no persistentes:
Cada conexión TCP transporta un mensaje de solicitud y otro de respuesta, es decir, se envía u n objeto por cada conexión. Es el que está presente en HTTP 1.0.

• Conexiones HTTP persistentes:
se pueden enviar varios objetos en una sola conexión. Es el vigente en el HTTP actual.

Algo fundamental en la red, son los tiempos de respuesta y para entenderlo hay que definir el Round Trip Time (RTT).

Round Trip Time es el tiempo desde que hacemos una solicitud desde el navegador hasta el ser vidor web, hasta que obtenemos respuesta de ese servidor. Como http se basa en el protocolo TCP, ¡su conexión no persistente lleva un acuerdo entre tres fases; primero el navegador solic ita que se inicie esa conexión, la otra máquina responde a esa

solicitud, y por último el cliente le indica a ese servidor que ha recibido todo correctamente, en este mensaje ya se le puede pedir el siguiente objeto a ese servidor.

2.2.1 Tipos de navegadores

Hay dos tipos de navegadores: multihilo y multiproceso. Un navegador es multihilo si al abrir di ferentes conexiones solo tenemos un proceso corriendo mientras que en el multiproceso, por cada pestaña abierta se abre un nuevo proceso.

2.2.2 Formato de los mensajes HTTP.

Hay dos tipos de mensajes HTTP: de solicitud y de respuesta.

2.2.2.1 Mensaje HTTP de solicitud.

Todos los mensajes en HTTP 1.1 están en formato ASCII, lo que quiere decir que son legibles. L a estructura del formato del mensaje HTTP es la siguiente, primero hay una línea de solicitud, l a cual tiene tres partes: el comando, el objeto que queremos, y la versión del protocolo emple ado. A continuación, tenemos unas de cabecera, que son varios comandos, como el host utiliza do, el navegador, como está la conexión, el lenguaje aceptado. Por último, hay que añadir una línea en blanco, como retorno de carro, para decir que el mensaje ha terminado. El cuer po puede ir vacío. Los métodos disponibles en HTTP 1.0 y 1.1 tenemos:

• GET: Sirve para pedir un objeto. Aquí el cuerpo va vacío.

•POST: Pedimos una url pero el contenido depende de lo que vaya en el cuerpo, por ejemplo, s i es una respuesta a un formulario, en el cuerpo irá lo escrito en el formulario.

•HEAD: Es similar al GET pero responde únicamente si lo tiene o no el objeto, pero no lo envía, es para depuración. Además, solo en HTTP 1.1 tenemos:

• PUT: Carga un objeto en el servidor.

•DELETE: Borra el objeto del servidor. Todo esto tiene que ir con unos permisos.

2.2.2.1 Mensaje HTTP de solicitud.

Los mensajes de respuesta siguen la misma estructura. Es un mensaje en ASCII y comienza con el protocolo que va a utilizar, después un código de estado numérico, y una frase explicando es e estado. A continuación,
la conexión, la fecha, el servidor, la última modificación, el tamaño y el tipo de contenido. Y po r último el objeto que me pide. Algunos de los códigos más conocidos son:

•200 OK: Que indica que la conexión ha sido exitosa y devuelve la información en mensaje de r espuesta. • 301 Moved Permanently: El objeto se ha movido de forma permanente.

• 400 Bad Request: Significa que la solicitud no ha sido comprendida por el servidor.

• 404 Not Found: El objeto solicitado no existe en el servidor.

•505 HTTP Version Not Supported: Indica que la versión de HTTP no es soportada por el servid or. Los navegadores hoy en día utilizan más protocolos, como SPIDY, que es el que ha dado lu gar al nuevo HTTP 2.0, que está todavía en desarrollo pero podremos aprovechar las ventajas q ue tiene.

2.2.2.2 Cookies

Como sabemos, el protocolo HTTP no tiene memoria por lo que necesitamos de algún medio q ue almacene los datos. Este medio son las denominadas, cookies. Se crearon en el año 2000, y es tán descritas en el RFC 2965. Una cookie es un par nombre-valor, que tiene el nombre de es ta y el valor que almacena. Tiene una fecha de expiración, a partir de la cual se elimina, y el do minio y path al que pertenece. Se utilizan, principalmente, para mantener el estado que http n o puede mantener, para ello podemos crear, a parte de las cookies, una capa de sesión, que se mantengan durante toda la sesión del usuario. La seguridad de las cookies está mal vista, por que la gente cree que son virus. En verdad,
son simplemente ficheros de texto, y pueden almacenar todo tipo de datos, dependiendo de c uales hayas dado tu. Son peligrosas cuando son datos que llegan a terceros, pero esto es siem pre bajo tu permiso. Las cookies de un servidor no pueden ser vistas por otros servidores.

2.2.2.3 Cachés web (servidores proxy)

El objetivo de estos servidores es poder dar una solución más rápida al navegador que si tuvier a que consultar la web en un servidor original. No hay que confundir estas cachés, con las cach és de los navegadores. La que nosotros vamos a ver sirve para configurar nuestro ordenador p ara que navegue por un proxy, que almacenará las respuestas, y las facilitará más adelante par a posibles búsquedas de otro cliente. Actúa como servidor de los clientes que piden la inform ación y como cliente de los servidores web a los que pide la página. Son instalados por el prov eedor de servicios de internet. Las ventajas que presenta son:

- Navegación más rápida.
- Reducción y control del tráfico
- Navegación anónima.
- Navegación más segura.
- Decidir navegación desde otros países.

¿Cómo nos aseguramos de que la caché tiene los datos actualizados? Pues con gets condicionales .

El navegador mira primero en su caché si tiene el objeto, si lo tiene hace un get condicional dici endo que me envíe el objeto si ha sido modificado desde, y pone la fecha en la que recibió el o bjeto. Esto lo hacen los navegadores y los proxys. En caso de que no haya sido no modificado, s e envía un mensaje 304 Not modified y le envía el objeto desde la caché. Si ha sido modificado, envía un 200 OK con el objeto solicitado.

2.2 Protocolo FTP(File Transfer Protocol)

Este protocolo sirve para la transferencia de ficheros(tanto subida como bajada). Sigue un mo delo cliente-servidor. Fue uno de los primeros protocolos utilizados en internet,
pero no fue hasta el año 85 que no se plasmó en la RFC 959. Cada vez se usa menos, pero se s igue utilizando para subir ficheros de golpe. Se hace con conexiones separadas para los datos d e control y los datos propiamente dichos. El cliente establece la conexión de control, con el pu erto 21, se debe establecer la conexión con usuario y contraseña,
aunque algunos permiten que sea de forma anónima. Una vez aceptado, el cliente se pone a n avegar por los directorios de ese servidor y cuando ve algún fichero que le interesa puede solic itar la transferencia de ficheros. O bien, para descargarlos o bien, para subirlos a esa carpeta d ebe establecer una segunda conexión TCP, esta vez para los datos, puede ser en el puerto 20 o en otro, y por esta conexión es por donde se envían los datos. Si queremos transferir otro fich ero se establece otra conexión de datos. Este método se denomina 'Fuera de banda'(Out of ba nd), porque envía los datos fuera de banda. Estos servidores si mantienen el estado.

2.2.1 Similitudes y diferencias entre FTP y HTTP.

2.2.1.1Similitudes

●Ambos protocolos permiten transferencia de ficheros y se ejecutan sobre TCP. 2.2.1.2 Difere ncias.

● FTP utiliza dos conexiones distintas y HTTP solo una conexión,

●HTTP envía la información de cabecera de solicitud y respuesta por la misma conexión TCP qu e transfiere el archivo
(en banda), mientras que FTP envía la información de control fuera de banda.

●El servidor FTP tiene que mantener el estado. 2.2.2 Comandos FTP y respuestas Comandos e n ASCII, igual que en HTTP, por tanto, son legibles por los usuarios. Cada comando va en una lí nea y tiene unos argumentos opcionales.

2.2.3 Modos de conexión

2.2.3.1 FTP activo
(standar o modo port) Se llama así, porque el cliente envía un comando port, desde su puerto de control, que está por encima del 1024, y lo envía al puerto de control del servidor, que sie mpre está escuchando en el puerto 21, este envía un comando de respuesta de que se ha dad o por enterado y le envía los datos desde el puerto 20. No sabe a qué puerto enviarle el servid or al cliente los datos. Esto se selecciona de forma aleatoria, siempre por encima del 1024. Por tanto, ya se conocen los puertos, el de datos del servidor (siempre 20) y el de datos del cliente (el aleatorio). Presenta problemas de seguridad, porque abrimos un puerto aleatorio que con ocemos de antemano, que puede ser conocido por cualquiera. Por eso surgió el modo pasivo.

2.2.3.2 FTP pasivo.

En el modo pasivo en vez de enviar un comando port, enviamos un comando PASV, del cliente al servidor desde el puerto de control, por encima del 1024. Cuando está en modo pasivo, el se rvidor responde indicando al cliente en qué puerto va a establecer el puerto de datos ese servi dor(
por encima del 1023) , el cliente ya sabe por esta respuesta donde va a estar escuchando el ser vidor y abre una conexión en un puerto superior(1 más) a donde se ha establecido el puerto d e control.

2.3 Correo electrónico. (SMTP, POP3, IMAP).

En el correo electrónico tenemos varios elementos fundamentales:

● Cliente de correo (agente de usuario):
Los clientes de correo componen, editan, leen y envían mensajes de correo. SI queremos envi ar un mensaje se lo enviamos al servidor de correo que lo envía al servidor de correos salientes y si queremos leerlo vamos al buzón a leerlo. Ej: Apple mail, Outlook

● Servidores de correo: Tienen los mensajes entrantes y salientes de los usuarios.

● Protocolo SMTP:
Este protocolo utiliza TCP para enviar correos. Tene un lado cliente que envía el correo y un lad o servidor que lo recibe. Ej:

2.3.1 Protocolo SMTP:
Es un protocolo que fue concebido hace 30 años, está recogido en la RFC 5321. Tiene unas rest ricciones muy fuertes, ya que la codificación que admite es de ASCII de 7 bits. Esto hace que si queremos mandar una imagen un fichero tenemos que codificarlo en ASCII, y luego descodifica rlo. Utiliza TCP para una transferencia fiable y utiliza el puerto 25. Es una transferencia directa que se hace en tres fases: 1. El cliente especifica el correo y la dirección a la que lo enviará. 2. El servidor envía el mensaje. 3.
El cliente repite el proceso hasta enviar todos los mensajes. Los comandos y las respuestas so n en código ASCII y envían un código de estado y una frase que explica esto.

2.3.2 Comparación de SMTP con HTTP.

2.3.2.1 Semejanzas.

- Ambos transfieren ficheros entre máquinas.
- Utilizan conexiones permanentes.
- Tienen interacción ASCII.

2.3.2.2 DIferencias.

- SMTP requiere que los mensajes estén codificados en ASCII de 7 bits.
- SMTP utiliza CRLF
(línea en blanco . linea en blanco) para determinar el final de los mensajes.

●HTTP protocolo tipo pull)
cliente solicita los ficheros quue el servidor le enviará más adelante), SMTP protocolo ti po push(el cliente empieza y envia la comunicación y los ficheros).

●HTTP cada objeto se encapsula en el propio mensaje de respuesta, pero en SMTP los objetos se envían en un único mensaje.

2.3.3 Formatos de los mensajes de correo.

SMTP no especifica el formato. Estos son especificados en la RFC 5322. Tiene una línea de cab ecera en la que tienes que poner: para y de quien es el mensaje y el asunto. A continuación, ha y una línea en blanco para separar las distintas partes. Y, en el cuerpo, el mensaje con los carac teres ascii.

2.3.4 Protocolos de acceso de correo.

SMTP no es el único protocolo para utilizar el correo. SMTP que sirve para enviar y almacenar c orreos electrónicos, puedo utilizar también o POP3, IMAP o HTTP.

2.3.4.1 POP3(Post Office Protocol version 3).

RFC 1939. Tiene una fase de autorización donde se van a enviar una serie de comandos, el clie nte va a poder decir que usuario es y que password tiene para poder acceder a la conexión. El servidor puede dar dos respuestas OK o error. Después, la fase de transacción, donde el cliente puede descargar y borrar o descargar y guardar los mensajes, utilizaremos comandos como lis t, rete, dele, quit, etc. Por último, la fase de actualización, donde el servidor borra los mensajes marcados. Si borras un mensaje del servidor no puedes volver a leerlo. POP3 no tiene memo ria de estado, únicamente mantiene durante la sesión información de los mensajes marcados para ser borrados.

2.3.4.2 IMAP (Internet Mail Acces Protocol)

RFC 3501. Más funcionalidad que POP3, pero más complejo. Permite manipular mensajes alma cenados en el servidor creando carpetas IMAP deja, por defecto, los mensajes en el servidor. Ti ene una memoria de estado. Tenemos comandos para descargar parte de un mensaje, para d escargar, por ejemplo, sólo las cabeceras.

2.3.4.3 HTTP Se utiliza para acceder desde cuentas como GMail, Hotmail, desde el navegador

2.4 DNS.

Este protocolo nos sirve para identificar las máquinas en internet con sus direcciones. En inter net las máquinas las vamos a identificar mediante la dirección IP, que es un conjunto de númer os o mediante su nombre, que nos sirve para recordar las direcciones de una forma fácil. ¿Qu é correspondencia hay entre la IP y el nombre? DNS(Domain Name System) En el año 71 apare ció el fichero HOST mediante el cual se almacenaban la correspondencia entre nombres y dire cciones IP. Esto se hizo inmanejable y se creó DNS. DNS puede referirse a: Una base de datos di stribuida implementada de forma jerárquica con varios servidores de nombres. O, un protocol o de la capa de aplicación que permite al host consultar esa BD distribuida. Los servidores DN S suelen ser máquinas Unix que ejecutan BIND sobre UDP utilizando el puerto 53. Este definid o en las RFC 1034 y 1035. Los servicios que proporciona DNS son la traducción del nombre del host a dirección IP, un alias para los hosts y servidores de correo. Una distribución de la carga.

Que son servicios transparentes al usuario, es decir, el usuario no es consciente que los está uti lizando. ¿Por qué utilizamos una base de datos distribuida y no centralizada? Porque sino, solo habría un punto de fallo, tendríamos un gran volumen de tráfico, una gran distancia a la BD, pr oblemas de mantenimiento y de escalado.

2.4.1 Base de datos distribuida y jerárquica.

2.4.1.1 Servidores DNS raíz.

Existen unas 13 granjas de servidores en todo el mundo, hay una alta concentración en Nortea mérica, pero también hay en Asia y Europa. Cada uno de los servidores están clasificados en u n cluster de servidores por seguridad y fiabilidad. Estos servidores son raramente consultados gracias a la caché de otros servidores.

2.4.1.2 Servidores TLD y autoritativos.

● TLD:
Son los responsables de los dominios de nivel superior y de los de países: .com, .org, .edu, .es

● Autoritativos:
Son mantenidos por las organizaciones que tienen los hosts accesibles públicamente. Deben p ublicar esa dirección ip de la máquina que tienen junto con el nombre, y ese par hay que public arlos en un servidor DNS autoritativo. Son mantenidos por la organización o por un proveedor externo.

2.4.1.2 Servidores DNS locales.

No pertenecen estrictamente a la jerarquía. Cada ISP tiene un servidor DNS local. Cuando una máquina va a pedir una IP para un determinado nombre se conecta con el servidor local. Actúa como un proxy y si ya tiene la IP pues la devuelve y si no se va a la jerarquía. Es el primero con sultado por una máquina para saber la IP que queremos.

2.4.1 Resolución de nombres en internet.

Si una máquina quiere saber la IP de un servidor web, este servidor está dentro de una máquin a(www), que está dentro de un dominio (urjc .es). Este dominio está dividido en dos zonas, urjc y es. Podemos hacer dos consultas:

2.4.1.1 Consulta iterativa.

1. Mensaje de consulta al DNS local.

2. El DNS local envía la consulta al DNS raíz.

3. Toma el sufijo es y devuelve la IP del TLD correspondiente.

4. DNS local consulta tld.es

5. Toma el sufijo urjc.es y devuelve IP de DNS autoritativo.

6. DNS local consulta al DNS autoritativo de urjc.es

7. Devuelve IP de www.urjc.es

8.DNS local devuelve IP al host.

2.4.1.2 Consulta recursiva.

Pregunta a un servidor que a su vez pregunta a otro. Así sucesivamente hasta encontrar lo des eado. Se desaconseja su uso por la carga que requieren.

2.4.2 Almacenamiento de caché en DNS.

Los servidores DNS van almacenando las correspondencias entre nombres e IPS que consiguen y así reducen los retardos y mensajes DNS en la red. Esa caché la descartan al cabo de un tiem po y se irá actualizando a medida que se descarten lo obsoleto. Estos mecanismos de actualiz ación/notificación están siendo rediseñados debido al tiempo que se tarda en dar respuesta. L os servidores TLD están habitualmente en las cachés de los DNS locales.

2.4.3 Registros DNS

Al final los registros DNS no son más que cuadruplas que se denominan resource records.RR, c on un nombre, valor, tipo , y tiempo de vida. Los registros más importantes son los A, con el n ombre de la máquina y el valor, que es la IP. También, podemos pedir un registro tipo NS, en e l nombre tenemos el dominio, y en el valor vamos a tener el nombre de un DNS autoritativo pa ra dicho dominio, que tiene los registros que corresponden a las máquinas de ese dominio. Po r otro lado, tipo=CName, en el nombre figura un alias para un nombre canónico, nombre real. Por ejemplo en www.ibm .com, www es un alias, el nombre real es servereast.backup2.ibm.co m Por último, tenemos TIPO=MX, tenemos que el valor es el nombre canónico del servidor de correo con un Nombre.

2.4.3 Mensajes DNS

Hay dos tipos de mensajes; mensajes de consulta y mensajes de respuesta, ambos van a segui r el mismo formato, es decir, la misma cabecera y el mismo cuerpo. Lo que cambia va a ser qu e hay en cada uno de sus campos. Hay distintos campos: El campo de identificación va a ser común a ambos mensajes. Tenemos un número identificador, de 16 bits, que tendrá que ser el mismo para ambos, para identificar a que consulta está dando respuesta ese mensaje. Luego los flags , que tendrán un bit para indicar si es de consulta o de respuesta, otro que dirá si la consulta es iterativa o recursiva, otro que dirá si admite o no recursión, etc. Además, tenemos cuatro ca mpos, que indican el número de registros que vienen detrás. En la consulta que hagamos vam os a incluir el número de preguntas que estamos haciendo al servidor DNS. En la consulta en el cuerpo los tres últimos campos van vacíos, y en la respuesta estarían llenos. Además, de ten er, otra vez, el número de consultas hechas.

2.5 Inserción de registros en la bd DNS

Para qué del nombre lleve a la dirección IP, lo que hacer es comprar un registrador DNS. Tene mos que tener dos registros DNS, uno primario y otro secundario. Para entenderlo pondremos un ejemplo.

- Primero se registra el nombre NRA.com mediante un registrador DNS.
    - Se verifica la unicidad del nombre.
    - Se proporciona un nombre e IP de servidores de nombres autoritativos.
    - El registrador inserta dos RRS (registradores de DNS) en el servidor.

● Crea dos RR en los dos DNS autoritativos: Tipo A para www.nra.com y tipo MX para mail.nra.com.

2.5 Aplicaciones P2P

Las arquitecturas P2P puras no necesitan de servidores, y los host se conectan entre sí haciendo de clientes y de servidor a la vez, se conectan de forma intermitente y cambian su IP.

Cuando queremos, desde un servidor, distribuir un fichero a N máquinas, ¿Cuánto tiempo se tarda para distribuir este fichero utilizando una arquitectura cliente-servidor y una P2P? Suponiendo que el fichero tiene un tamaño F, y el servidor tiene una velocidad de subida de Us y cada uno de los host tienen un velocidad de subida de Un, y de bajada Dn.

En una arquitectura Cliente-servidor cada cliente solicitará el fichero al servidor, y el servidor tendrá que enviar secuencialmente N copias. El tiempo que va a tardar va a ser el tiempo que tarda al enviar las N copias de tamaño F por un enlace Us. Pero también podemos tener otro factor limitante, que sea la velocidad de descarga del cliente más lento. Hay que mirar las velocidades de descarga y ver cual es la mínima.

Este valor vemos que se incrementa el tiempo linealmente con una N muy grande. Si miramos la arquitectura P2P, al servidor no le hace falta que envíe las N copias a cada cliente, porque el par que enviará el fichero, subirá una única copia de ese fichero y los demás peers se encargarán de descargarlo a una velocidad determinada "Di" .

El mínimo va a marcar uno de los límites del tiempo máximo de distribución. También lo puede limitar la velocidad de subida de los N pares, por tanto para subir las N copias el tiempo es la suma de las subidas de los enlaces de los demás pares.

2.5.1 Distribución de ficheros. (BitTorrent)

Cada vez que queremos compartir un fichero se va a generar un torrente, es decir, un grupo de peers que intercambian los datos de ese fichero. La mayoría de los programas de este tipo no son P2P, sino que son híbridos, porque cuentan con un servidor centralizado, tracker, que rastrea los peers que participan en el torrente. Por tanto, cada vez que un peer se une al torrente se va a registrar en ese tracker y periódicamente va a decir que está en ese tracker.

Para compartir los ficheros se dividen en fragmentos, estos fragmentos son los que se intercambian. El peer que se conecta no los tiene, pero los tendrá cuando los descargue, y el tracker le asignará de forma aleatoria un conjunto de pares, y le da las IPs de esos pares al nuevo peer que se acaba de conectar. Este peer lo primero que hará es conectarse con un subconjunto de peers, los llamaremos vecinos.

Mientras se descarga el fichero, cada peer cargará fragmentos en otros pares. Una vez que el peer tiene el fichero completo, puede abandonar el torrente, que se considera egoísta, o permanecer para seguir compartiéndolo de forma altruista.

En cada instante cada peer tiene diferentes subconjuntos de fragmentos. Entonces, periódicamente el peer pide a sus vecinos la lista de los fragmentos que tengan y que el no. El más raro es el que primero intentará obtener. El envío de fragmentos sigue una filosofía toma y daca(tid-for-dat) el peer envía fragmentos a los 4 vecinos que le envían fragmentos a mayor v

elocidad. Y esos cuatro vecinos los reevalúa durante 10 segundos. Si se hace así no habría cabi da a vecinos nuevos, así que, cada 30 segundos se selecciona de forma aleatoria un nuevo veci no y puede entrar en la lista de los cuatro, esto se denomina un "optimistically unchoke". Bitt orrent no utiliza un tracker sino que utilizan tablas HASH distribuidas, son bases de datos con p ares clave valor(nombre contenido y dirección IP).

2.5.2 Distribución de ficheros. (Skype)

Es una aplicación P2P de un par de usuarios que quieren conectarse. Tiene un protocolo propie tario y no se puede estudiar en profundidad. En skype se introduce un índice que va a signar lo s nombres de usuario Skype a iP. Esto se hace en super pares y con un servidor Skype. Es más, una arquitectura mixta.

TEMA 3. CAPA DE TRANSPORTE.

La capa de transporte ofrece sus servicios a las aplicaciones que se encuentran en la capa supe rior. Llevará los mensajes generados por las aplicaciones desde el puerto de una máquina hast a el puerto de otra máquina.

3.1 Servicios de la capa de transporte.

3.1.1 Servicios y protocolos de transporte.

La capa de transporte se encarga de la comunicación lógica entre procesos de aplicaciones que se ejecutan en diferentes hosts, para que parezca que están conectados directamente. Los pr otocolos de esta capa, al igual que en la de aplicación, se ejecutan en sistemas terminales, no en el núcleo de red. Cuando una aplicación de un host quiere enviar un mensaje lo envía a l a capa de transporte, que será la que lo divida en varios segmentos, que a su vez los pasará a l a capa de red que se los dará a la capa inferior. Una vez encapsulado con las diferentes cabece ras, este se transmite por la red hasta el destino, donde se desencapsula, siendo la capa de tra nsporte la encargada de recoger los distintos segmentos en los que se descompuso el mensaje, recomponerlos y entregarlo donde corresponde.

Los protocolos de transporte más importantes son TCP y UDP. El protocolo TCP es orientado a conexión (antes de enviar cualquier proceso se establece una conexión full-duplex(en ambos sentidos) e ntre las dos máquinas (CC)), con una entrega de paquetes fiable(sin errores, ni pérdidas), cont rol de flujo y de congestión (regulando la velocidad de emisión cuando la red está saturada). P ero, no garantiza temporización, ni mínima tasa de transferencia, ni seguridad.

El protocolo UDP es ligero, no orientado a conexión y que no garantiza ni entregas de datos fiables, ni controles de flujo o congestión, ni temporización. Pero garantiza un pequeño control de errores básico. UDP es un protocolo simple, que únicamente coge el mensaje, le añade una cabecera con puer tos de origen y destino y se lo pasa a la capa de red para su envío inmediato. La ventaja, es que , al ser tan ligero, las aplicaciones de tiempo real funcionan mejor con UDP. Para garantizar un a temporización, una tasa de transferencia mínima y seguridad se hacen diseños inteligentes q ue en internet no se pueden garantizar.

3.1.2 Servicios y protocolos de red.

La capa de transporte se encarga de la comunicación lógica de procesos. Tiene por debajo la ca pa de red, que ofrece sus servicios a la capa de transporte encargándose de comunicar dos má quinas. La capa de transporte comunica los procesos de esas máquinas y la de aplicación las ap licaciones que ejecutan esos procesos. La comunicación de la capa de red se hace mediante el protocolo IP. Este es un protocolo de entrega de mejor esfuerzo, es decir, no fiable, por lo que la capa de red no garantiza ni entrega, ni orden,
ni integridad de los segmentos que le pasa la capa de transporte para ser enviados. Si queremo s garantizar esto debemos emplear un protocolo como TCP en la capa de transporte. En la cap a de red identificamos a cada host por su dirección IP, al igual que en la capa de transporte ide ntificamos cada proceso con un número de puerto.

3.2 Multiplexación y demultiplexación.

TCP y UDP amplían el servicio de entrega entre dos terminales a un servicio de entrega entre d os procesos de esos terminales. Para ello emplean la multiplexación y demultiplexación.

La multiplexación
es el proceso por el que el emisor recopila datos de diferentes sockets, les añade una cabecera y crea los segmentos que pasan a la capa de red para su envío. Por otro lado,
la demultiplexación
es la decisión para saber a qué socket enviar los datos desde la capa de transporte, que recibe todos los datos de la capa de red, a la capa de aplicación. El número de puerto es un número v irtual de 16 bits,
comprende valores entre 0 y 65535, aunque los primeros 1024 están reservados. Los más empl eados son los de la imagen. En el host destino, la capa de transporte recibe de la red un datagr ama IP, este datagrama tendrá en su cabecera los datos de la dirección IP de origen (host emis or) y de destino. Dentro de este datagrama estará el segmento de transporte que debemos ent regar al socket. El host emplea las direcciones IP y los números de puerto para identificar el so cket adecuado.

3.2.1. Demultiplexación sin conexión. Comunicación UDP(sin conexión)

El socket de la comunicación UDP, vendrá identificado por la IP destino y por el número de pue rto del proceso destino. Cuando un host recibe un segmento UDP mirará el número de puerto destino en el segmento y enviará, el mismo, al socket correspondiente a ese número de puerto . El origen de los datagramas (IP y puerto) no importa, ya que todos irán al mismo socket de la máquina destino. La IP y el puerto origen se utilizan para poner la dirección de retorno, pero n o para identificar el socket. La máquina destino cuenta con un buffer en el que se almacenan l os paquetes y que los cuente en orden. El puerto origen con el de solicitud coinciden.
El Comando netstat muestra un listado de procesos y Puertos abiertos en una máquina.

El netstat-o muestra las máquinas y el número de procesos que abren comunicación.

3.2.2 Demultiplexación orientada a la conexión

En este caso el socket TCP viene determinado por la IP destino, el puerto destino, la IP de orige n y el puerto de origen. Al introducir estos campos un servidor podrá tener varios sockets TCP simultáneos en el mismo puerto. De hecho, tendrá tantos sockets como procesos cliente de ot ras máquinas se estén conectando a ese puerto. Al contrario que en UDP, dos segmentos entra ntes con distinta IP de origen se dirigirán a sockets diferentes porque esta IP de origen

forma parte de la definición del socket. Por ejemplo, los servidores web van a tener diferentes sockets para cliente que se conecta. HTTP no persistente utilizará diferentes sockets para cada solicitud. Podemos diferenciar los sockets por la IP origen.
(lo de S-IP) También podemos tener procesos como el 2 de la máquina B que tendrán su socket creado que podremos demultiplexar porque el puerto de origen es distinto. Lo que hacen los servidores web es tener un único proceso que crea varios hilos de ese proceso que van escuch ando en distintos sockets las distintas solicitudes que les llegan de las distintas máquinas.

Servidor web con varios hilos

3.3Transporte sin conexión: UDP(User Datagram Protocol) (RFC 768)

UDP es un protocolo de transporte de internet sencillo y minimalista, es decir, no hace realme nte casi nada con los mensajes que le pasa la capa de aplicación. Es un servicio de mejor esfuer zo, hace lo que puede para que los mensajes lleguen, aunque cuenta con un pequeño control d e datos. Estos segmentos UDP se pueden perder y entregarse desordenados. Es un protocolo si n conexión, no hay negociación entre el emisor y el receptor, y cada segmento UDP se trata de forma diferente al resto. En caso de que nosotros quisiéramos implementar una transferencia fiable sobre UDP podríamos añadir esa fiabilidad en la aplicación.

¿Por qué existe UDP? Porque permite un mejor control en la capa de aplicación sobre qué se e nvía y cuándo (solo lo encapsula y lo envía). No sufre retardos. Además, no tiene la informació n del estado de conexión, con lo que evitamos los retardos y soporta más clientes activos que TCP. Y por último, hay menor sobrecarga en los paquetes porque la cabecera de UDP (8 bytes) es más pequeña que la de TCP(20 bytes). El protocolo UDP es el más utilizado en apli caciones multimedia, como streaming, dado que no son sensibles a pérdidas y si lo son con los tiempos. También se utiliza UDP para DNS, para administración de red (SNMP) o para protocol os de enrutamiento (RIP). En el ejemplo de la derecha, podemos ver como tenemos los 16 bits del puerto origen y destino, el siguiente es la longitud en bytes del segmento, con cabeceras in cluidas, y, finalmente un checksum que verifica la integridad del mensaje. El checksum, o sum a de comprobación, tiene como objetivo detectar errores en los segmentos transmitidos. El em isor coge todo el segmento y lo trata como una secuencia de enteros de 16 bits, a continuación , el checksum calcula a esa suma el complemento a 1 de todas las palabras del segmento, por ú ltimo el emisor pone el valor del checksum en el campo checksum del segmento. Por otro lado , el receptor, va a calcular esta suma y comprobar si coincide con el valor del campo emisor. Si coincide todo correcto, que no, ha habido algún fallo.

3.4 Principios de un servicio de transferencia de datos fiable.

Estos principios que veremos son genéricos y se aplican a cualquier capa, pero los vamos a apli car a la de transporte. Para hacer una transferencia de datos fiable es necesario proporcionar una abstracción a la capa superior de un canal fiable. Con fiabilidad nos referimos a que ningún bit se rompa, que no haya pérdida de paquetes y que se entreguen en el orden que deben.

Este es uno de los temas más importantes en las redes de ordenadores.

Las características que tenga el canal no fiable que tengamos por debajo y que queramos dotar de fiabilidad, van a ser las que determinen si este protocolo es más o menos complejo. Desde la capa de aplicación hay un proceso emisor que transmite los datos a la capa inferior, en este caso la de transporte. Si no hay fallos se entregan los paquetes al receptor. Lo que sucede nor

malmente es que tendremos un canal no fiable y en la capa de transporte tendremos que dota r de distintos protocolos (protocolos de transferencia de datos fiable) que añadan esa fiabilida d que no tenemos. RDT: Reliable Data Transfer, transferencia de datos fiable UDT: Unreliable Data Transfer, transferencia de datos no fiable En la capa de aplicación enviamos los datos ha sta la capa de transporte. A continuación, en el lado del emisor, este protocolo de transferenci a de datos fiable va a ser llamado por la función rdt_enviar() cuando tenga datos que enviar, e ste a su vez transfiere, mediante udt_enviar, los paquetes al canal no fiable. Cuando el recept or recibe esos paquetes por el canal de transferencia de datos no fiable, va a llamar a la funció n rdt_recibir(), que dice que al protocolo de transferencia de datos fiable que reciba esos paq uetes. Por último, mediante entregar_datos(), se entrega todo a la capa superior, la de aplicaci ón. Esto, es un ejemplo de lo que puede ser un protocolo de transferencia de datos fiable. Lo s protocolos que vamos a ver a continuación tienen una forma de funcionar denominada stop and wait o de bit alternante.

En la que el emisor envía un paquete y espera la respuesta del receptor,
vamos a tener esto siempre que tengamos que recibir una respuesta por cada paquete enviad o si queremos enviar otro paquete. En un protocolo de parada y espera el número de secuenci a nos bastará con un bit, un 1 o un 0, según si está transmitido por primera vez o ya es más de una. Como aquí estamos suponiendo que no hay pérdida de datos, no hace falta que el NAK o ACK digan que paquete han contestado.

3.4.1 RDT 1.0: Canal totalmente fiable

Es un canal totalmente fiable, sin errores de bit y sin pérdidas de paquetes es, por tanto, el má s sencillo ya que solo pasamos los datos de capas superiores a inferiores (envío) y de inferiores a superiores (recepción). Las máquinas de estado finitas serán distintas para el emisor y recept or. El emisor envía los datos por el canal, mediante rdt_enviar, así se crea el paquete con esos datos que le llegan, y esto se le pasa a la función de enviar, que lo envía a una capa que tiene p or debajo. El receptor espera esa llamada por debajo, cuando recibe ese rdt_recibir extrae los datos de ese paquete y utiliza la función extraer_datos y utiliza entregar_datos a la capa superi or con esos datos que ha traído. Tanto emisor como receptor vuelven al mismo estado.

3.4.2 RDT 2.0: Canal con errores de bit.

Aquí tenemos un canal con errores de bit, es decir, el canal que hay debajo puede corromper l os bits, pero no se pueden perder, los puede cambiar y transmitir con errores, pero siempre lo s transmite. Empleamos una suma de error, checksum, para comprobar los errores. Podemos recuperarnos de estos errores gracias a: -Reconocimiento positivo (acknowledgements- ACKs): El receptor le comunica explícitamente al emisor que el paquete se ha recibido bien, es decir, cada vez que recibe un paquete el receptor le envía un mensaje de OK. -Reconocimiento nega tivo(negative acknowledgements - NACKS): El receptor le comunica explícitamente al emisor q ue el paquete tiene errores, Para ello se envía un NAK. El emisor tiene que volver a transmitir el paquete cuando recibe un NAK. -Nuevos mecanismos en rdt 2.0 respecto a rdt 1.0 -Detecci ón de errores

Receptor tiene que enviar al emisor NAKs o ACKs

-Y si hay error hay que retransmitir. El emisor va a tener dos estados, en el primero espera la ll amada de arriba, en cuanto al receptor solo hay un estado. Existe un problema fundamental y es que pueden aparecer paquetes duplicados si lo que ha fallado ha sido el NAK/ACK

¿Cómo lo tratamos? Pues el emisor tendría que añadir un número de secuencia a cada paquet e y el receptor descartará el paquete duplicado. Descartar significa no entregar a la capa super ior. Hay otras dos opciones, una volver a preguntar lo que no hemos entendido, si el ACK o N AK, ¿Qué es esto último que me has mandado un ACK o NAK? pero

podríamos entrar en un bucle infinito sin entender la respuesta. Otra es añadir más bits de co mprobación,
que no solo detecten el error si no también puedan recuperarse de sus errores, esto no resolve ría la pérdida de paquetes, aunque rdt 2.0 es sin pérdidas.

3.4.2 RDT 2.1:

Como resumen de este protocolo, el emisor añade un número de secuencia o 0 o 1. Debería ch equear si el ACK o el NAk es corrupto o correcto. El receptor debe chequear si el paquete recibi do está duplicado, pero el receptor no sabe si su NAK o ACK se ha recibido correctamente, por lo que tiene que enviarlo de nuevo cada vez que se pierde un paquete.

3.4.3 RDT 2.2 protocolo sin NAK

Tienen la misma funcionalidad que rdt 2.1 pero solo lo envía ACKs, sin NAKs. Esto es porque el receptor va a enviar un ACK con el último paquete recibido correctamente. De esta forma el r eceptor debe incluir el número de secuencia del paquete al que se refiere el ACK. Si el emisor r ecibe un ACK duplicado, lo que quiere decir es que algo no se ha recibido bien, es lo mismo qu e un NAK, y tiene que retransmitir el paquete actual.

3.4.4 RDT 30: canal con pérdidas y errores de bit.

Aquí podemos perder tanto los paquetes como los ACKs. Por tanto, no nos basta con el checks um o el número de secuencia, hay que añadir un temporizador para poder recibir ese paquete. Cuando pasa ese tiempo retrasmitiremos el paquete si no hemos recibido el ACK,
bien sea porque se perdió el ACK o el paquete. Si ese paquete se ha retrasado y no se ha perdi do lo que hará será duplicarse al enviar ese mensaje, aún así, al tener ese número de secuenci a esto no es un problema. En este caso se pierde un paquete, el receptor nunca lo recibe, co mo habíamos iniciado un

temporizador, en cuanto el emisor vea que el tiempo se ha agotado piensa que algo se ha perd ido, y entonces vuelve a enviar ese paquete. Si se pierde el ACK el resultado es el mismo. Par a el emisor es lo mismo pero para el receptor no puesto que tiene duplicados. Aun así, envia e l ACK correspondiente pero manejando duplicados, y lo que hace es esntregar el primero reci bido a la capa superior y el segundo no. En el caso (d) volvemos a tener duplicados y solo en tregamos el primero a la capa superior. El problema del RDT 3.0 es el rendimiento, puesto qu e mandamos un mensaje, mientras que podríamos enviar varios a la vez y disminuir el tiempo.

El tiempo de utilización del canal es muy bajo si lo hacemos con protocolos de parada y espera. La solución a este problema son protocolos de procesamiento en cadena(pipelining),

3.4.5 Protocolos de procesamiento en cadena

Aquí puedo enviar paquetes sin esperar los mensajes de reconocimiento. Ejemplo, mando 20 y ya iré recibiendo los ACKs de cada uno. ¿Cuántos paquetes puedo enviar sin recibir el ACK? Pu es esto hay que decidirlo según lo que queramos. Cuantos más paquetes enviemos más rapide z hay en el canal, pero también más posibilidades de que falle. Esto va a determinar el rango d

e números de secuencia que vamos a utilizar. Además, necesitaremos un buffer para almacena r los distintos paquetes que vamos enviando. El uso del pipelining va a aumentar la utilización del canal. Existen numerosos protocolos de procesado de paquetes en cadena para recupera rse de los errores. Vamos a ver dos de los más utilizados. Ambos protocolos tienen en cuenta q ue el emisor puede tener hasta N paquetes sin ser reconocidos por el canal, es decir, hasta N

paquetes enviados sin esperar a recibir el ACK de ese paquete. El protocolo Go Back N: El rece ptor solo envía los ACKs acumulativos, quiere decir que, no reconoce paquetes con números d e secuencia no consecutivos. El emisor tiene un solo temporizador. Si el temporizador expira el emisor vuelve a retransmitir todos los paquetes no reconocidos, aunque algunos ya hayan sid o enviados.

La imagen muestra en el emisor un número de secuencia de k bits en la cabecera del paquete para diferenciar un paquete de otro, para rechazar duplicados y para hacer ACKs. La ventana d e emisión tendrá un tamaño N, es decir, podré enviar desde el emisor N paquetes sin esperar s u reconocimiento. Por tanto, desde el inicio (número de secuencia 0) hasta el número de secu encia base menos 1 son todos los paquetes transmitidos y reconocidos. Y desde el siguiente n úmero de secuencia hasta el número de secuencia base más N menos 1 son todos los números de secuencia que tengo disponibles para poder enviarlos de forma inmediata. A partir de base +n son los que no puedo enviar directamente porque están por encima de la ventana de emisi ón. Dicha ventana se va desplazando cuando va recibiendo los ACKs. Los números de secuenci a son finitos, tenemos k bits, por tanto, tenemos hasta 2^k -1 números de secuencia. N no pue de ser mayor que 2^k.

Si el receptor recibe un paquete desordenado se descarta,
no se almacena en ningún buffer, porque no tengo buffer, esto simplifica mucho el receptor.
En el protocolo de repetición selectiva
se reconocen los paquetes individualmente y si recibimos un paquete con números de secuenc ia no consecutivos los almacenamos. Aquí contamos con un temporizador para cada paquete n o reconocido. Si el temporizador expira solo retransmitimos el paquete al que corresponde ese temporizador.

El receptor envía un ACK individual de cada paquete. En caso de que el receptor reciba un paq uete que no le toca, se almacenará

en un buffer para enviarlo posteriormente a la capa de aplicación. Cuando el emisor detecte q ue no se ha recibido algún paquete, por la finalización de su temporizador, va a tener que env iar ese únicamente ese paquete. Contamos con una ventana de recepción, esta es lo que nos permite almacenar los paquetes que se envían posteriores al que pedimos en el buffer. A part e de almacenarse se envían los ACKs correspondientes pero seguimos esperando hasta que lle guen los anteriores. Una vez esté el paquete solicitado la ventana se mueve hacia la derecha.
Uno de los problemas que podemos tener en repetición selectiva es elegir el tamaño de venta na respecto al rango de números de secuencia que tenemos disponibles. Normalmente se elig en como máximo n/2 del tamaño de ventana, siendo n el número de secuencia. Si cogemos má s para la ventana habría un problema y es que al mover la ventana los ACKs no se recibirían cor rectamente, porque tendrían el mismo nombre que los anteriores.

3.5 Transporte orientado a conexión: TCP

TCP (Transmision Control Protocol) es el protocolo de transporte más utilizado. Es un protocol o de punto a punto, es decir, que cuenta con un emisor y un receptor. Es fiable, esto quiere de cir que los bytes de emisor a receptor van correctos y en orden, y para ello hay que implement ar las medidas vistas anteriormente. Además, es en cadena, pipelining, y vamos a tener en cue nta el tamaño de la ventana, en el cual vamos a poder enviar varios paquetes sin esperar el rec onocimiento de estos por el receptor. El tamaño de ventana determinará la velocidad a la que se transmiten y esta vendrá dada por el control de congestión y de flujo. Tanto el emisor

como el receptor van a tener unos buffers en los que se encuentran los paquetes antes de ser enviados. El proceso escribe los datos y los va pasando al buffer de emisión donde se transfier en los paquetes por la capa de transporte. En el buffer de recepción, se llena según la velocida d a la que se vayan leyendo los datos. Si ese proceso que lee los datos es un proceso lento, es te buffer se podría llenar. Para eso tenemos el control de flujo, para controlar la velocidad a la que enviamos los datos desde el emisor y no saturar al receptor. TCP es un servicio de datos f ull duplex, es decir, bidireccional y que podemos enviar los datos en ambas direcciones. Tendr emos un MSS: Maximun Segment Size, que vendrá determinado por el tamaño máximo que pe rmiten las capas inferiores. Es orientado a conexión
, es decir, necesito llegar a un acuerdo para poder enviar los datos. Es un
acuerdo en tres fases (handshaking)
en el cual se intercambian, emisor y receptor, unos mensajes de control para poder acordar c ómo van a hacer esa comunicación.

3.5.1 Estructura de segmento.

Esta es la estructura típica de un segmento TCP. Tendremos un número de puerto origen y otr o puerto destino. La longitud, no es la total, sino que es la de la cabecera, que es variable porq ue tiene unas opciones que se pueden, o no, incluir y el checksum. Después de la cabecera vi enen los datos de la aplicación. Hay que incluir el número de secuencia y el número de recono cimiento, que es el ACK que estoy recibiendo. Osea que puedo enviar un paquete y hacer un A CK a la vez. También tenemos una serie de bits que podemos activar o desactivar en cualquier momento. Los bits FRS sirven para establecer y cerrar la conexión. Por último, tenemos una ve ntana de recepción que dice cuantos bytes está dispuesto a aceptar el buffer del receptor.

3.5.2 Transferencia de datos fiable.

TCP funciona sobre la capa de red que utiliza como protocolo IP. IP es un servicio no fiable y ha y que dotarlo de fiabilidad, esto se hace gracias a TCP, mediante mecanismo como los que vim os anteriormente. Emplea ACKs acumulativos, aunque se puede implementar (esto se hace ha bitualmente)
en las opciones de la cabecera utilizar los selective ACKs (SACK) y son ACKs individuales, como l os de repetición selectiva.

TCP solo utiliza un único temporizador. Esas retransmisiones se pueden disparar porque termi na el temporizador o porque recibimos ACKs duplicados. Vamos a considerar inicialmente que es un TCP simplificado, e ignoraremos los controles y los ACKs.

¿Cómo funciona TCP?
Utiliza los números de secuencia y reconocimiento para poder evitar el tema de los duplicados y para saber que llega correctamente. En cuanto al número de secuencia el emisor, es bidirecci onal así que también se refiere a los que envía el receptor, va a incluir en su paquete un númer o de secuencia y uno de ACK.

El número de secuencia le dice siempre el siguiente byte que está esperando en el propio host. El número de secuencia no es el número de paquete, si no que hace referencia al flujo de bytes que se transmite en ese segmento. El emisor elige el número de secuencia al azar. Cuando el r eceptor recibe segmentos no ordenados tiene que ver que hacer con esa secuencia, si almacen arlo o descartarlo. Por lo general lo guarda, para que, al recibir los otros los entregue todos a la vez. ¿Qué funciones debe cumplir un emisor? Dependiendo del evento deberá hacer unas ac ciones u otras, si recibe datos de la aplicación para hacer un envío, tiene que crear el segmento , añadirle el número de secuencia, que será el primer byte de datos del segmento dentro del fl ujo de bytes y será aleatorio pero luego todos seguirán a este. Cuando recibe los datos para e nviar debe iniciar el temporizador. Ese temporizador se fija con un valor.

Cuando el emisor recibe un fin de temporizador tiene que retransmitir el segmento que lo caus ó y reiniciar el temporizador. Si recibe un ACK, si ese ACk reconoce segmentos no reconocidos, tendrá que actualizar la información e iniciar el temporizador si aun hay segmentos no recono cidos.

Suponemos que el emisor es A y el receptor B.

El primer ejemplo es la pérdida de un paquete ACK y el segundo que hemos puesto un tempori zador demasiado bajo para que se pueda recibir el elemento. El primero envía 92-99 (8 bytes) y el segundo 92-99 (8 bytes) y 100-119 (20 bytes). El ACK va a tener un valor siguiente al que hemos solicitado.

No es necesario reenviar el primero porque tenemos un ACK acumulativo, que dice que si hem os recibido el segundo bien el primero también.

¿Cómo se generan los ACKs? (Importante)

Una funcionalidad que se incluyó en la versión de TCP, que se denominó TCP RENO, es la retra nsmisión rápida. Lo que hace es que antes de terminar el temporizador, se reenvía el paquete, esto es porque prevé que algo ha sucedido. Lo sabe porque empieza a recibir ACks duplicados Lo que hacemos es que cuando se ha perdido el paquete nos llegan esos ACKs y el emisor sabe que al recibir tres ACks duplicados el segmento se ha perdido y lo reenvía sin esperar al tempo rizador.

¿Cómo decidimos cuánto tiempo esperar a la respuesta del paquete? Ese tiempo tiene que ser mayor que el tiempo de ida y vuelta del paquete (RTT), porque si es demasiado corto no le dar emos tiempo a volver. Si el margen que le damos es muy largo reaccionaremos de forma lenta a la pérdida de paquetes. ¿Cómo se estima el tiempo de ida y vuelta? Se envía una muestra y s e observa cuánto va a cambiar con el paso del tiempo. Se ignoran las retransmisiones para esti mar un RTT típico. Después calculamos un promedio en el que voy a tener en cuenta el tiempo que acaba de tardar, el RTT muestra, y los estimados anteriores. Esto lo pondero, voy a mirar c uanto voy a tener en cuenta la muestra actual, lo multiplico por un alfa y el resto es lo anterior. El valor normal de alfa es 0.125

Si le damos mucha importancia a la muestra actual los picos influyen en el tiempo, si se lo doy al estimado anterior los picos no me influyen tanto. Aquí se ve cómo afectan las muestras q ue voy tomando.

Antes del TCP Tahoe se ponía el temporizador al doble del

tiempo que estimabas que iba a tardar. El problema es que había una distancia muy grande. L a solución es el TCP tahoe. Ponemos el temporizador que se ajusta más al RTT estimado. Cuan do hay una variación muy grande damos más margen. De esta forma vamos a colocar el temp orizador poniendo el RTT estimado más un margen de seguridad, antes del tahoe ese margen e s el doble Y, a partir de tahoe estimamos cuando se desvía la muestra actual respecto al RTT E stimado. Eso se calcula con la siguiente fórmula:

Así calculamos la desviación del RTT. Nuestro intervalo acabaría siendo:

3.5.3 Control de flujo

El control de flujo es una forma de regular y adaptar la velocidad de emisión de tráfico que tien e TCP. Otra forma de regularlo es con el control de congestión. En el control de flujo el emisor intenta no inundar al receptor enviando los datos demasiado rápido. Si el receptor tiene un bu ffer de recepción se irá llenando con datos, y estos los irá leyendo el proceso de la aplicación q ue accederá para poder extraer los que han recibido en esa máquina. Si esa app es muy lenta l eyendo o el tráfico que llega es demasiado rápido, ese buffer se va a llenar y, llegará un mome nto que los datos en el buffer se pierdan. Para evitar esto se adapta la velocidad de emisión. ¿Cómo se adapta la velocidad de emisión? Tenemos la ventana de recepción, que es el espaci o libre de ese buffer, y ese tamaño de ventana, que se puede calcular como: La ventana de r ecepción y el ACK del receptor se envían a cada datagrama que recibe del emisor. Por tanto, el emisor recibirá un valor en el que conocerá cuantos datos le caben al receptor en ese buffer. El emisor limitará en el siguiente segmento el número de bytes no reconocidos, garantizando no desbordar el buffer. Pero, ¿qué pasa si esa ventana se va reduciendo y llega a ser 0? en ese ca so el emisor no emitiría y llegariamos a una situación de bloqueo, donde no envía paquetes y n o se reciben ACK ni nuevos valores para la ventana. Para solucionar esto el emisor envía al rec eptor paquetes de un byte, cada cierto tiempo,
para obtener una respuesta del receptor y que le indique el tamaño de ventana, además de pa ra detectar cuando hay espacio en el buffer. 3.5.4 Gestión de la conexión TCP Tanto el emisor como el receptor establecen la conexión antes de intercambiar segmentos. Por lo que se inicial izan los números de secuencia, los buffers, la ventana de recepción, etc. El cliente es el que ini cia la conexión y el receptor tiene que aceptar dicho socket. Para aceptar la conexión tenemos el triple handshake o acuerdo en tres fases. Este acuerdo en TCP tiene las siguientes fases:

Paso 1.

El cliente envía un segmento SYN al servidor, por tanto pone el bit SYN de la cabecera a 1, no e nvía datos en este segmento y especifica el número de secuencia elegido de forma aleatoria. P aso 2: El servidor recibe el SYN y responde con un segmento en el que pone el bit SYN a 1 y el A CK a 1. Además, rellena con el valor de ACK el número de secuencia que envía el cliente increm entado en uno. Por otro lado, el servidor asigna los buffers correspondientes e indica el númer o de secuencia inicial del servidor. La conexión es bidireccional por eso tienen números de sec uencia. Paso 3: El cliente recibe ese SYN ACK y va a responder con un segmento ACK, en este c aso ya podemos enviar datos, muchas veces simplificamos el establecimiento de la conexión c on un paquete de ida y otro de vuelta, porque en este tercero ya podemos enviar datos, dado

que ya está la conexión establecida. Aunque realmente son tres fases: El paquete SYN, el SYN ACK, y por último el ACK que se recibe en el servidor. donde el bit SYN ya estaría a 0. Para cer rar la conexión establecida se hace en diversas fases. Paso 1: El cliente envía un segmento de c ontrol FIN al servidor en el que indica que quiere terminar la conexión.

Paso 2:

El servidor recibe FIN y envía un ACK, diciéndole al cliente que ha recibido el FIN, si este no tien e más datos que enviarle cierra la conexión y envía a su vez un FIN.

Paso 3:

Cuando el cliente recibe el FIN responde con un ACK entonces entra en un tiempo de espera e n el que responde con ACk a todos los FIN que reciba.

Paso 4:

Cuando el servidor recibe ese ACK se cierra la conexión.

3.6 Principios del control de la congestión

La congestión en la red significa que hay demasiadas fuentes enviando demasiado tráfico para ser manejado por la red. En el control de flujo tratábamos de no inundar al receptor, aquí inte ntamos no hacerlo a la red, a los routers del camino antes de llegar a destino.

¿Cómo vamos a darnos cuenta de que existe congestión? Porque se producen grandes retardo s en la red, es decir, llega demasiada información que se almacena en los buffers que tienen es tos routers, y la cola de los buffers aumenta hasta llenarse e incluso perder paquetes. Por tant o, tendremos retardos y pérdida de paquetes en los routers de la red. Este problema es uno de los principales en redes. Podemos ver como se causa esta congestión en el siguiente escenari o.

TEMA 4. CAPA DE RED

4.1 Introducción

La capa de red se encuentra bajo la capa de transporte. El emisor encapsula los segmentos, en datagramas y los transporta hasta el host receptor, donde coge el mensaje y lo entrega a TCP o UDP. Los protocolos de la capa de red estarán presentes en los hosts y en los routers interme dios, ya que el router tiene que examinar las cabeceras de esos paquetes para determinar qué camino ha de seguir Las funciones clave que vamos a tener en la capa de red son: - Reenvío (Forwarding) : Mueve paquetes entre los puertos de entrada y salida del router. - Enrutamiento (Routing)
: Determina la ruta de los paquetes del origen al destino. Eso lo realizarán los algoritmos de en rutamiento. Como analogía podemos ver que el enrutamiento es el proceso de planificar el via je desde el origen al destino y el reenvío es el proceso de paso por un cruce de carreteras. Un ejemplo: Tenemos un router que tiene que decidir por donde tiene que salir un paquete, para ello examina el valor de la cabecera y consulta una tabla, denominada tabla de reenvío local, e n la cual le dice que enlace de salida tiene que tomar. Estas tablas se rellenan siguiendo algorit mos de enrutamiento, que son los que deciden la ruta que tiene que tomar el paquete para lle gar a destino. Cada uno de los routers tendrá unas tablas de reenvío que han sido rellenadas p or estos algoritmos. Estos algoritmos son de un nivel superior.

4.2 Redes de circuitos virtuales y de datagramas.

Es un sistema parecido a los servicios de la capa de transporte, UDP o TCP, pero, en este caso, el servicio es terminal a terminal y no podemos elegir qué red utilizar ya que viene determinad o por la red. Los dos servicios que nos proporciona la capa de red son:

4.2.1 Circuitos Virtuales (VC):

Especifican una ruta origen-destino parecido a un circuito telefónico. Proporciona un servicio d e conexión en la capa de red. Es similar a la conmutación de circuitos. Podemos simular una re d de conmutación de circuitos, para ello tendremos en cuenta:

-Criterios de rendimiento.

- Si la red participa en la ruta origen-destino.

Además, hay que:

-Establecer la conexión antes de enviar los datos.

-Cada paquete tiene que tener un identificador de VC, y nos dirigirnos según este ident ificador, no según el router destino.

- Cada router del camino mantiene información de estado de la conexión.

- Se asignan recursos de routers y enlaces para cada VC. Los circuitos virtuales se utiliz an para dar prioridad a determinados clientes. Esto no se puede hacer en una red de d atagramas, que son las utilizadas habitualmente.

4.2.2 Red de datagramas

No hay que establecer la conexión, por tanto, los routers no tienen que mantener el estado de la conexión. Además, los paquetes se reenvían utilizando la dirección del host de destino,así los paquetes podrán tomar distintas rutas aunque el origen y el destino sean el mismo.

La toma de decisiones en las que se elige por dónde tendrán que ir saliendo esos paquetes, esto se denomina Packet Switching.

Las tablas de reenvío
indican al router qué salida debe tomar cada paquete Esta tabla se rellenará con las direcciones IP de destino. La dirección IP será en la IPV4, que es la más extendida, una dirección de 32 bits, por tanto tendremos 4 mil millones de entradas posibles(2^32). No se ponen todas las direcciones si no que se utiliza la coincidencia con el prefijo más largo. Ponemos el prefijo, es decir, los primeros bits de cada dirección y diremos que los que coincidan por tales bits van por una interfaz de enlace u otra.

4.4 IP: Protocolo de Internet.

En esta capa de red tenemos una serie de protocolos de enrutamiento como RIP ,OSPF o BGP, que son los que van a decidir que ruta van a tomar los paquetes, además de rellenar las tablas de reenvío de los distintos routers. O el protocolo ICMP que es un protocolo de señalización de informe de errores en la red. Pero, el más importantes es el protocolo IP, ya que es el único que permite formar los datagramas disponibles en la capa de red.

4.4.1 Formato de un datagrama en el protocolo IP:

Un datagrama en la IPv4, que es la versión más extendida, tiene este formato:

La cabecera es de longitud variable porque hay opciones que podemos o no añadir. Por eso se indica la longitud de esta. El segmento TCP o UDP está encapsulado.

Tenemos en IP un aspecto de fragmentación y reensamblado
, que viene debido a que hay un tamaño máximo (MTU) en la capa de enlace que podría ser superado. Para evitarlo hay que fragmentar (dividir el paquete original en pequeñas partes) los datagramas. Una vez llegado a destino se hace el reensamblado de estos fragmentos para formar el mensaje original. Un datagrama no está fragmentado si el desplazamiento y el indicador no son ambos iguales a 0. El campo identificador será siempre el mismo que el datagrama padre. En el indicador indicamos si hay o no más fragmentos después de él. Si está a 0 no hay más fragmentos. Si está a 1 y el desplazamiento a 1, estamos en el primer fragmento.

4.4.2 Direccionamiento IPv4 Una interfaz es la conexión entre un host/router y el enlace físico.

Los routers suelen tener múltiples interfaces y los hosts suelen tener sólo una interfaz. Cada interfaz tendrá una dirección IP. Por otro lado, las direcciones IP se asocian con cada interfaz.

Una dirección IP tiene 32 bits y vienen expresados en bloques de 8:

En los comienzos de internet, el reparto de de direcciones IP de los distintos host se realizó mediante la arquitectura de clases, en la que se dividió todo el rango de las direcciones IP en cinco clases.

La dirección 0.0.0.0 está reservada por IANA para la identificacióno local. La dirección de red tiene los bits de host iguales a cero y sirve para definir la red en la que se ubica. Se utiliza para hablar de paquete de difusión a toda la red. EJ: 192.168.0.0 La dirección de broadcast que tiene los bits correspondientes a host iguales a uno, sirve para enviar paquetes a todos los hosts de la red en la que se ubica. Se utiliza para hablar de la red. EJ: 192.168.255.255 Las dire cciones 127.x.x.x se reservan para designar la propia máquina. Se denomina dirección de bucle local o loopback.

A partir del año 93 vieron que dividir en rangos podía limitar la difusión de las direcciones y ent onces era muy poco flexible, por ello, comenzaron a utilizar CIDR (Classless InterDomain Routin g) enrutamiento entre dominios sin clase. Con este nuevo método, el tamaño de la dirección d e la subred era arbitrario, es decir, que se puede decidir bit a bit. Así, podemos decidir los bits de subred y los del host. Y definimos las direcciones con a.b.c.d/x (dirección/bits que forman la subred).

Así tendremos identificadas las subredes, que serán los bits de mayor orden y los host que son los bits de menor orden.

¿Qué es una subred?

Son interfaces de dispositivo cuya parte de dirección IP es la misma, interconectados entre ell os sin router. Para determinar las subredes tengo que quitar los hosts y los routers, y me que darían, únicamente, las subredes.

¿Cómo obtiene un ISP un bloque de direcciones?

Con un ICANN(Internet Corporation for Assigned Names and Numbers). Tiene como funciones asignar direcciones, gestionar DNS, asignar nombres de dominios y resolver conflictos. ¿Cómo hace la red para obtener la parte correspondiente a la subred de una dirección IP? Recibe una porción del espacio de direccionamiento de su proveedor (ISP). El haber elegido las direccione s IP y como han sido asignadas facilita el enrutamiento.

Las direcciones IP han sido divididas en varios bloques. Si la organización 1 pasa del ISP A al ISP B hay un problema de enrutamiento que se soluciona por la regla de coincidencia del prefijo más largo.

¿Cómo obtiene un host una dirección IP? Se puede obtener de forma manual en la que el admi nistrador del sistema introduce la dirección en un fichero. Pero, también, utilizando el protocol o de DHCP (Dynamix Host Configuration Protocol). Se trata de un protocolo transparente al us uario y que el servidor va a proporcionar esa dirección IP al host. Muy utilizado en redes inalá mbricas o residenciales. -DHCP: DHCP es un protocolo plug-&-play que ofrece, de forma diná mica, una dirección al host. Este servidor le va a ofrecer la IP al host y el host va a poder renov ar ese tiempo de arrendamiento de la IP durante el tiempo de uso que necesite. Esto nos perm itirá reutilizar direcciones. Es un soporte muy sencillo para usuarios móviles que se conectan in termitentemente y en sitios distintos a la red. La IP tiene que ir cambiando para poder conecta rse en distintos sitios. Este protocolo consta de 4 fases para la conexión:

-En la primera buscamos el servidor. Descubrimiento DHCP.

-Después el servidor nos responde diciendo que existe y que está ahí. Oferta DHCP.

-Solicitamos la dirección. Solicitud DHCP.

-Por último, el servidor podrá enviar esa dirección. ACK DHCP .

EJEMPLO:

1)Tiene dirección origen = 0.0.0.0 porque todavía no tiene dirección IP. Y su dirección t ambién es 0.0.0.0.

2) A continuación, le da una IP en la etapa de Oferta DHCP.

3) El cliente vuelve a solicitar su dirección porque puede que haya varios servidores D HCP en la red y le puedan llegar distintas ofertas al cliente. No es lo normal pero puede darse el caso en el que respondan varios a la misma petición. Si pasa el cliente será el que tenga que elegir a quien pedirselo.

4)Cuando coinciden los dos ID de transacción el cliente ya tendrá la dirección IP que ha pedido. El servidor DHCP da, además que las direcciones IP, tres cosas: la ip del route r del primer salto (first-hop), también denominado pasarela o Gateway
que va a ser el router a la cual el host tiene que conectarse para conectarse a internet .

El nombre y dirección IP del servidor DNS local. Y, también, le va a dar la máscara de subred , es decir, que porción de las direcciones ip son de subred y cuales de host. Ejemplo: Un clie nte llega a una red e intenta conectarse a la misma. En esa red tenemos un servidor DHCP que , normalmente está incluido en el router, y está en la dirección IP 168.1.1.1. Lo que necesita es te cliente/host es una dir IP, la dir del gateway, la del servidor DNS y la máscara de la subred. El cliente generará un mensaje DHCP que se encapsulará en las distintas capas y se enviará a t oda la red(255.255.255.255) y le llegará al servidor DHCP. Ese servidor recogerá ese paute y lo desencapsulará. Al final el servidor DHCP envía un ACK DHCP con la ip del cliente, la del gatew ay, el nombre y la IP del servidor DNS y la máscara de subred. Eso se lo envía al cliente, lo enca psula, crea un paquete y lo envía. De esta forma el cliente ya conoce su IP, nombre e IP del ser vidor DNS, la IP de su gateway y la máscara de subred.

Uno de los servicios más empleados en internet es el NAT
(Network Addres Translation). NAT traduce las direcciones privadas, que son compartidas(em piezan por 10 o 12.168), por la dirección pública, única, que es la que nos asigna el ISP. El ISP n os asigna una única IP y, con ella, tengo que gestionar todos los equipos que tengo en mi red p rivada . Todos los datagramas que salgan de mi red tienen que guiarse por las direcciones públ icas. NAT utiliza los números de puerto para poder jugar con una única IP. Asigna distinto núm ero de puerto con la misma IP de salida a cada uno de los procesos que está corriendo. Ventaj as:

-No será necesario solicitar varias direcciones al ISP.

-Podré cambiar las direcciones en la red local sin notificar al resto del mundo.

-Se puede cambiar de ISP sin modificar las direcciones de los dispositivos de la red local.

-Los dispositivos de la red local no van a ser directamente visibles. No vale como única medida de seguridad. Ejemplo de tablas de traducciones NAT:

Una de las máquinas de la red local, tiene un navegador corriendo en un puerto (3345) y quie re conectarse a un servidor web que está en otro puerto (80) de una máquina de fuera. Nuestr

a máquina, que tiene una dirección IP en la red local de 10.0.0.2, envía un datagrama al servid or web. Esa petición le llega al router y le asigna, a esa máquina y a ese proceso, un número d e puerto (5001) y la IP pública, cambiando la dirección de origen que tenía en la red privada (10.0.02.3345) por (138.76.29.7.5001). Al cambiarlo se registra en la tabla de direcciones NAT. Posteriormente cuando llegue la respuesta del servidor web, llegará a la dirección destino que veía en el origen del datagrama anterior. Cuando llega esa respuesta traduce la dirección. Para traducirlo lo hace cambiando la dirección del datagrama y el puerto destino. Por tanto, el ro uter NAT debe: -Reemplazar los datagramas salientes, IP origen y nº de puerto con la IP del NA T y el nuevo número de puerto -Recordar la tabla de traducciones NAT que tiene que corresp onder a IP origen, y nº de puerto origen con IP del NAT con nuevo nº de puerto. -Reemplazar l os datagramas entrantes cambiando el IP del NAT y el número de puerto que está almacenado en la tabla. El número de puerto es de 16 bits (tenemos 60000 conexiones simultáneas con so lo una IP.. NAT es motivo de controversia, porque el número de puerto tiene que direccionar p rocesos, no host. Con esto estamos direccionando procesos y host a la vez.

También los routers tienen que procesar paquetes solo en la capa de red y aquí estamos viola ndo el enfoque terminal a terminal (los routers son nodos intermedios que no deberían modifi car los paquetes) Todo esto viene porque las direcciones IP se estaban acabando. Con NAT ap arece un problema y es que cuando un host está detrás de un router NAT no puede actuar de servidor, porque tenemos una dirección privada que no es conocida fuera y no podemos dejar que otros establezcan conexiones con él, porque no puedo publicar mi dirección. Para ello apa rece el NAT transversal, con el que se crea un "nuevo host C" que permita, mediante TCP, con ectar A con B. La forma más sencilla de proporcionar NAT transversal es con el protocolo Plug-&-Play (UPnP). El protocolo Plug-&-Play proporciona un mecanismo de NAT transversal. Si ten emos una app que se está ejecutando en un host puede solicitar la correspondencia NAT, de e sta forma, si el traductor acepta la solicitud los host exteriores se pueden conectar a la IP priva da, puerto privado. El protocolo también permite conocer la IP pública, puerto público. UPNP yo puedo conocer de antemano cual va ser lo público y puedo anunciarme con esto. Ejemplo: Imaginemos un host situado detrás de un router NAT.

Tiene una dir privada : (10.0.0.4. 3345) En este puerto se está ejecutando una aplicación BitTo rrent (app P2P) El router NAT nos proporciona el puerto y dirección pública le corresponde a l o anterior para poder conectarse con el exterior. ¿Cómo hace esto? BitTorrent le pide al route r NAT que cree un túnel en el que le haga le de un puerto y dirección IP nuevos, anunciando d onde pueden estar. Por tanto, cuando un host pregunte dónde está corriendo esa app de BitT orrent le responderá y se podrá enviar un paquete de conexión ahí. De esta forma, el router re cibirá el paquete de conexión y cuando lo reciba lo cambiará internamente a la dir privada y lo reenviará dentro.

También utilizamos NAT transversal con VoIP, lo que hacemos es que nuestra máquina se con ecta al RELAY, que está fuera del NAT, con ese se puede conectar a otros equipos. De esta for ma cuando otros me busquen se conectan ahí y cuando se establezca la conexión el relay func iona de puente.

4.4.3 El protocolo ICMP

El protocolo ICMP (Internet Control message protocol) lo utilizan los host y routers para comu nicar información, como informes de errores o mensajes de solicitud/respuesta de eco, a nive l de red. ICMP es un protocolo que va sobre IP, es decir, los mensajes se encapsulan en datag ramas IP. Tienen una cabecera simple, en la que se identifican un campo tipo, un código, más

los 8 primeros bytes del datagrama IP causante del error. Dependiendo de los campos tipo y código tenemos distintos mensajes. Uno de los usos más comunes se da cuando utilizamos tr aceroute, este comando determina los routers por los que va a ir pasando un determinado pa quete, y da información de los tiempos de retardo de cada router, además de su nombre e IP. Esto se hace mediante el siguiente mecanismo: El origen envía tres series de segmentos(3 seg mentos UDP con TTl 1, otros tres con TTL 2 y otros tres con TTL 3) a destino incrementando el tiempo de vida de cada uno de ellos, con un número de puerto destino poco probable. Cuand o lleguen a destino, como el número de puerto es poco probable que esté abierto, nos va a en viar el un mensaje de puerto inalcanzable o de TTL caducado. En este mensaje se incluye el no mbre del router y su dirección, además de información de routers anteriores. El origen puede c alcular el RTT, así sabrá el retardo de la red en cada momento. (HOJA CUADROS)

4.4.4 IPv6.

Es una nueva versión del protocolo IP, actualmente la más extendida es la versión 4. IPv6 se co menzó a utilizar en 1994-8, porque en el año 91 la ITF empezó a estudiar el problema de cómo expandir el número de direcciones, ya que en IPv4 están muy limitadas. Se introdujo IPv6 con l as RFCs 2460 y 3513.

El número de direcciones con IPv4 es de 4.294.967.296 pero el número fue creciendo y nos qu edamos sin direcciones IP. Se utilizaron nuevas técnicas para evitar agotar las direcciones, por ejemplo, el subnetting, o utilizar CIDR, NAT, o el poder compartir direcciones (PPP/DHCP). Exis te un organismo para asignar direcciones en IPv4, IANA
(Internet Assigned Numbers Authority) este organismo se divide en cinco IR y son los responsa bles de asignar a los usuarios las IPS en cada una de las regiones. Tenemos como regiones ARI N (América del Norte), LACNIC (América latina) AfriNIc (África) , APNIC (Asia), y RIPE (Europa).

En cuanto al agotamiento de espacio y direcciones, desde el 3 de febrero de 2011, IANA reparti ó los últimos bloques a los RIR, esos bloques de direcciones se han ido agotando dependiendo de la demanda. La primera se agotó en APNIC, a continuación, fue RIPE. Hay necesidades de d irecciones porque cada día hay más dispositivos y personas que quieren estar conectadas a int ernet. Por tanto, tenemos muchos usuarios, muchos dispositivos y muchas redes. Otra limitaci ón es que las tablas de enrutamiento eran cada vez de mayor tamaño. El protocolo es difícil de procesar en los routers. Tecnologías de acceso constante, Las redes y usuarios son más móviles . Protocolos que solicitan el hecho de ser transparentes (Plug&Play). Las soluciones que ofrec e IPv6 son:

-Mayor espacio de direccionamiento. (128 bits).

-Direcciones globales y transparentes, es decir, para cada usuario puedes darle un espacio eno rme y puedo utilizar direcciones públicas en internet. As´ahorramos las privadas. Tráfico de ter minal a terminal, nos conectamos con la máquina directamente.

-Estructura de red jerarquizada.

-Cabecera simplificada.

-IPsec obligatorio para encriptar la comunicación y que sea más seguro.

-ICMP revisad.

-Multicast en lugar de broadcast.

-Autoconfiguración y Plug&Play mejores que en IPv4.

Se ha aprovechado IPv6 para cambiar el formato de cabecera y mejorar la calidad de servicio. Cabecera de longitud fija de 40 bytes, y no se permite fragmentación, si es más grande se avis a el origen para cambiarlo y disminuir el tamaño.

Algunos cambios:

-No hay checksum porque lo hacemos en otras capas.

-Las opciones de IPv4 se permiten, pero fuera de la cabecera,

- Se incluyen mensajes nuevos como que el paquete es demasiado grande. Además de funciones de gestión de grupos multidifusión. (protocolo IGMP)
Para pasar de IPv4 a IPv6 no podemos hacerlo todo simultáneamente en un mismo día, por lo que se funciona con un periodo de transición en el que coexistirán ambos proto colos en un mismo router.

Esto se hace mediante:

-Pila dual: Los nodos IPv6 tienen también IPv4. En pila transformamos en B-a-C a IPv4, perdemos información

-Tunelización: el datagrama IPv6 es contenido en IPv4. Creamos un tunel entre los dos routers IPv6.

4.5 Algoritmos de enrutamiento.

El algoritmo de enrutamiento decide cómo rellenar las tablas de reenvío. Cada router tiene su propia tabla de reenvío. Para decidir esto vamos a seguir un modelo de grafos para ver las rede s. Los nodos van a ser los routers y los enlaces los arcos que unen los nodos. En estos grafos s e incluye un valor numérico en cada enlace que indica el coste del mismo. Lo podemos poner s iempre a 1, y si todos los enlaces cuestan 1 lo que buscaremos es aquella ruta que tenga un m enor número de enlaces. El coste también puede ser inverso al ancho de banda, de forma que nos cueste menos ir por las rutas que tienen un mayor índice. Nuestro algoritmo va a buscar la ruta de menor coste. La que minimice esos costes de nuestro grafo. Podemos clasificar los alg oritmos de enrutamiento de dos formas:

-Global: Todos los routers conocen la topología los costes de los enlaces.
Estos algoritmos son de estado de enlaces o Link-State.

-Descentralizada: El router conoce solo la topología y el coste de sus vecinos, pero no e l resto de la red. Vamos a necesitar un proceso iterativo para intercambiar informació n con los vecinos. Son algoritmos denominados vector de distancias o Distance Vector.
También se pueden clasificar según las rutas:

-Estáticos: las rutas cambian lentamente en el tiempo.

-Dinámicos: Las rutas cambian más rápidamente, actualización periódica en respuesta al cambio de los costes o de la topología de la red.

4.5.1 Algoritmo de estados de enlaces.

Algoritmo de Dijkstra:

Todos los nodos conocen la topología de la red y los costes de los enlaces. Es un algoritmo LS y todos los nodos tienen la misma información. Desde un nodo origen puedo calcular la ruta de origen mínimo al resto, esto determina la tabla de reenvío para ese nodo.Esto se calcula de for ma iterativa y después de un número de iteraciones vamos a conocer esa ruta de coste mínimo de los k nodos destino. Hay que conocer una cierta notación: -c(x,y): Coste del enlace entre l os nodos x e y -D(v) Valor actual desde el origen al destino v. -p(v) nodo predecesor en la ruta desde el origen hasta v.

-N': conjunto de nodos cuyo camino de coste mínimo se conoce. (EJEMPLO HOJA). Uno de los problemas principales que tenemos en este algoritmo es la complejidad, ya que es cuadrática, esto se puede mejorar llegando a nlog n. Otro problema son las oscilaciones por congestión. E n el caso de que el coste de enlace lo pongamos con la cantidad de tráfico que transporta ese e nlace se recalculan varias veces los costes.

La solución es ejecutar estos algoritmos en momentos aleatorios en distintos routers.

4.5.2 Vector de distancias.

Vamos a ver que es la ecuación de Bellman-Ford, define como dx(y) el coste de la ruta de coste mínimo de x a y. Ecuación: V es uno de los vecinos El algoritmo de vector de distancias utiliz a esta ecuación para calcular la ruta de menor coste. Lo hace haciendo estimaciones de ese val or D. Para cada vecino v, x mantiene Dy. La idea básica es que de vez en cuando, cada nodo e nvía su propia estimación de vector de distancia a los vecinos. Esto se hace de forma asíncrona . Cuando un nodo x recibe una nueva estimación actualiza su propio DV usando la ecuación ant erior. De esta forma este algoritmo es iterativo y asíncrono, cada iteración se produce porque cambia el coste de enlaces locales o mensaje de actualización de DV desde un vecino, esta es l a razón por la que actualizo los valores. Cada vez que cambie la información de un vector dista ncias tendré que notificar a mis vecinos de mi nuevo vector distancias y en ese caso, si mis veci nos tienen que cambiar su vector distancias cambiarán a sus vecinos y así, recursivamente, por eso se denomina un sistema distribuido. (EJEMPLO)

Los cambios en los costes de los enlaces se pueden ver de distintas formas: Una de ellas es qu e el nodo detecta un cambio en el enlace local. Es decir, que un nodo cambie su valor. Tenem os dos casos, uno en el que el valor disminuya que se siguen estos pasos: -En el instante t0, y d etecta un cambio en el coste de un enlace, actualiza su DV, e informa a sus vecinos.

-En el instante t1, z recibe la actualización de y, y actualiza su tabla. Calcula un nuevo coste mín imo.

-En el instante t2, y recibe la actualización de z y actualiza su tabla. Los costes mínimos de y no cambian, por lo que no envía ningún mensaje a z. EL algoritmo entra en estado de reposo. Si el coste aumenta en vez de disminuir tenemos un pequeño problema. En este caso necesitare mos hasta 44 iteraciones para estabilizar el algoritmo. A esto se le denomina "cuenta hasta infi nito".

Para solucionar esto existe la técnica de la inversa envenenada en la que si la ruta de z a x para por y, z le dice a y que su distancia a x, es infinito. Para que y no quiera enrutar a x por z. Esto n o resuelve al 100% el problema. Comparación entre LS y DV. Complejidad de envío de mens: LS es más complejos, porque DV solo intercambia distancias entre vecinos. Velocidad de conve rgencia: LS tiene el problema de las oscilaciones y en DV pueden producirse bucles.

Robustez ¿Y si un router falla?:

En LS los nodos pueden difundir costes de enlaces incorrectos y cada nodo calcula su propia ta bla. En DV los nodo cada nodo usa tablas de otros nodos, el error se propaga a través de toda l a red

4.5.3 Enrutamiento jerárquico.

Hasta ahora, nuestro estudio ha sido ideal, todos los routers han sido idénticos, hemos conside rado la red plana, no hemos tenido en cuenta la escala (no se pueden almacenar tantos datos en las tablas y los intercambios de información colapsarían los enlaces). Además, cuenta la aut onomía administrativa, internet es una red de redes y como administrador todos queremos te ner el control de su propia red. Sus routers van a enrutar la información por su red. Esto nos lle va a organizar los routers en distintas regiones, que vamos a denominar "sistemas autónomos" , los conocidos como AS. Los routers dentro de un mismo AS van a ejecutar el mismo protocolo de enrutamiento, llamado protocolo de enrutamiento interno. Cada AS ejecuta su propio prot ocolo interno y lo va a decidir la propia red, eso sí,
habrá ciertos routers, los gateway, que son los que conectan AS entre sí, que ejecutan otros pr otocolos para poder entenderse entre ellos. Así los sistemas autónomos quedan conectados a sí: Dentro de cada uno de estos routers tenemos la tabla de reenvío que se va a configurar m ediante dos tipos de algoritmos:

-Intra-AS: son internos al sistema autónomo y deciden el administrador de ese sistema autóno mo. Configuran entradas para destinos internos.

-Inter-AS: son los que se comunican con otros sistemas autónomos. Configuran entradas para destinos externos.

Si un router no es un gateway, tendrá que utilizar los algoritmos Intra-AS para llegar a un gate way y después llegar hasta fuera. Si sólo tuviéramos un gateway cada router interno enviará l a información cuando va hacia fuera se le enviaría a ese único gateway. Suponemos que un ro uter recibe un datagrama con destino fuera de ese propio sistema autónomo, 3C.

El router 1D tiene que aprender qué destinos se alcanzan desde AS3 y AS2, de esta forma el pr otocolo inter-AS, además de comunicarse hacia afuera, tendrá que propagar esa información a todos los routers de dentro del sistema autónomo diciéndoles que destinos se pueden alcanz ar desde AS3 y del AS2. Esto es un trabajo del inter-AS.

Imaginemos que el sistema 1 aprende que, a través de su protocolo interno, la subred x se alc anza desde AS3, pero no a partir de AS2, este sistema autónomo (AS1) tiene que propagar la i nformación de que X es accesible desde AS3, y por ejemplo. el router 1D va a determinar que para llegar a esa subred,
tiene que ir por 1C, que es el gateway, y para ello la ruta más corta está por el 1A. Por tanto, in serta la entrada de (x,I) en la tabla de reenvío. Eso significa que para llegar a "x" se tiene que ir por la interfaz "I".

Si la subred x se puede alcanzar desde AS3 y AS2, para poder configurar la ruta de reenvío, el r outer 1D debe decidir a qué gateway tiene que mandar el paquete, si al 1C o al 1B, esto es el tr abajo del protocolo de enrutamiento interno, y este trabajo se conoce como el enrutamiento de la patata caliente, el sistema autónomo suelta el paquete de la forma más barata posible. E n caso de empate se elige uno al azar.

Al hacer tablas jerárquicas podemos elegir rutas que no sean la óptima, porque puede no inter esarme que vaya por cierto sitio

4.6 Enrutamiento en Internet.

Como algoritmos de enrutamiento internos se utilizan los protocolos de pasarela interior (IGP) conocidos como protocolos de enrutamiento itnerno.De estos protocolos los mas conocidos so n el RIP, OSPF, y IGRP.

4.6.1 Protocolo RIP:

Sigue un algoritmo de vector-distancias y los vecinos van a darnos la informacion que tengan e n cada momento. Se ha extendido mucho porque se incluyó en UNIX en 1982. Utiliza una métri ca de distancia, es decir, número de saltos, con un máximo de 15 saltos. Esto quiere decir que t enemos un límite de 15 saltos, por lo que si tenemos sistemas autónomos de un diámetro may or a quince no podremos utilizar este protocolo.

Se incluye el último salto.

Rip es un algoritmo de vector distancias, entonces se intercambiarán mensajes entre vecinos c ada cierto tiempo, se denominan anuncio RIP, son una lista de hasta 25 subredes de destino, c on la distancia del emisor a cada una de esas subredes. Cada router tendrá una tabla RIP (tabla de enrutamiento) que incluye su vector distancias y su tabla de reenvío. Ejemplo:

Ejemplo: Cuando D recibe un anuncio del router A

Al recibir este mensaje se actualiza la tabla de enrutamiento de D. ¿Qué sucede en RIP cuand o falla un enlace o algún router? Si no se escucha un anuncio después de un cierto tiempo el ve cino/enlace se declara muerto. De forma que se invalidan todas las rutas de este vecino. Esto s e propaga por toda la red de forma rápida. Para prevenir los bucles de enrutamiento se emple a la técnica inversa envenenada (con 16 saltos) .

Aunque Rip sea un protocolo de la capa de red su implementación se hace como un protocolo en la capa de aplicación.

4.6.2 Protocolo OSPF:

Es un protocolo abierto que utiliza un algoritmo de estado de enlace. Por tanto, tendremos qu e diseminar paquetes de estado de enlace, y vamos a conocer en cada nodo el mapa topológic o completo. El cálculo de rutas lo hacemos utilizando el algoritmo de Dijkstra. De esta forma va mos a tener anuncios OSPF que transportan una entrada por cada router vecino. Estos anuncio s se diseminan vía inundación cada 30 minutos y se emplean unos mensajes cortos de actualiza ción. Estos anuncios son transportados sobre IP, no sobre UDP. El protocolo OSPF se suele im plementar en los ISP grandes y RIp en los pequeñitos. ¿Qué funcionalidades nos va a ofrecer O SPF que no tiene RIP?

-Seguridad: todos los mensajes OSPF pueden ser autenticados.

-Se permiten varias rutas de igual coste.

-En casa enlace permitimos múltiples métricas para calcular el coste.

-Soporte integrado para enrutamiento por unidifusión y multidifusión.

-OSPF se puede hacer de forma jerárquica, y subdividirlo en más áreas. Ejemplo de esto último .

De cada área tenemos un router frontera de área. Al tener una jerarquía de dos niveles en OS PF (área local y zona troncal) nos va a permitir que la inundación de los anuncios de estados de enlace sea únicamente en el área y, que, cada nodo conozca,
únicamente, la topología del área. Además del camino más corto a las subredes en otras áreas, pero no necesitan conocer la topología de todo el sistema autónomo. Tenemos los routers fro ntera de área que coleccionan distancias a redes en su área y lo difunden a otros routers de fro ntera de área y los routers de la zona troncal que ejecutan enrutamiento OSPF. Y los routers d e frontera que conectan a otros AS.

4.6.3 Protocolo BGP:

Es el protocolo de pasarela de frontera que solo se utiliza como protocolo inter-AS. Es el están dar de facto. BGP proporciona a cada AS mecanismos para:

-Que subred se puede alcanzar a través de los Sistemas Autónomos (AS).

-Propagar información de alcanzabilidad a todos los routers internos del sistema autónomo.

-Determinar buenas rutas a subredes, basándose en información de alcanzabilidad y en la polít ica del sistema autónomo.

Este protocolo permite a las subredes anunciar que existen al resto de internet. Se basa en ses iones BGP, que son conexiones TCP semipermanentes creadas entre pares de routers, pares B GP, para intercambiar esa información de enrutamiento. Estas sesiones BGP no tienen porqué corresponderse con enlaces físicos, sino que pueden ser, con TCP extensiones dos routers.

TEMA 5: SERVICIOS

5.1 Introducción y servicios.

Los hosts y routers los denominaremos nodos.

A los canales de comunicación que conectan nodos adyacentes los denominaremos enlaces .

Pueden ser cableados, inalámbricos, LANS. A los paquetes los denominaremos tramas o frames, y encapsulan un datagrama. La capa de enlace transmite esos datagramas d e un nodo a otro a través de un enlace.

Un datagrama puede ser transmitido mediante diferentes protocolos de enlace sobre diferent es enlaces. Es decir, de un nodo a otro se puede transmitir por Ethernet y de ese último nodo a l siguiente por WI-Fi. Cada protocolo puede proporcionar transferencia de datos fiable o no. ¿Qué servicios me proporciona la capa de enlace?

-Entramado: Consiste en encapsular datagramas en tramas, añadiendo cabeceras.

-Acceso al enlace: Si el medio es compartido hay que gestionar el poder compartir ese medio. Utiliza direcciones MAC, para identificar el origen y destino.

-Entrega fiable: Podemos proporcionar fiabilidad. En algunos enlaces como fibra o par t renzados no es necesario, pero si en enlaces inalámbricos.

¿Por qué añadir fiabilidad a nivel de enlace y punto a punto? Porque a veces nos interesa dete ctar el error en el que está y corregirlo. -Control de flujo: Podemos regular el envío y recepción entre nodos adyacentes. -Detección de errores: Podemos detectar errores debidos a la atenu ación de la señal o ruido. O porque el receptor detecte errores.

-Corregir errores: El receptor puede identificar y, en ocasiones, corregirlos.

-Servicio semi duplex y full-duplex:

-Semidúplex: Ambos servicios pueden recibir y transmitir, pero no a la vez.

-Full-duplex: Ambos servicios pueden recibir y transmitir a la vez.

¿Dónde se implementa la capa de enlace? Se implementa en todos los nodos. Se implementa en un llamado adaptador de red, conocido como NIC. Puede ser, por ejemplo, una tarjeta Ethe rnet. Tengo mi tarjeta Ethernet que se encarga de la transmisión física de los datos. Mediante un controlador puede enviar los datos y estos son manejados por la CPU. Va a estar trabajando junto a los buses del host y junto a una serie de materiales hardware, software y firmware.

La comunicación entre adaptadores de red se hace mediante un medio por el que transmito lo s datos, y los nodos. Desde el lado del emisor, encapsula datagramas en una trama, y añade bit s de comprobación de errores. Desde el lado del receptor mira si hay errores y extrae el datagr ama para pasarlo al nivel superior.

5.2 Detección y corrección de errores.

Para poder detectar o corregir errores hay que utilizar bits redundantes, también llamados códigos EDC. Esos datos que queremos proteger (D) pueden incluir o no la c abecera. Esta detección de errores no es fiable completamente. A mayor campo EDC mejor det

ección y corrección, pero también mayor carga. Hay que encontrar un compromiso entre la ca ntidad de sobrecarga y la cantidad de errores que queramos detectar. Una de las formas más simples de comprobar errores es la comprobación de paridad. Lo que hace es detectar un únic o error de bit, si hay dos errores o más no se pueden detectar. Suele detectar números impare s de errores, no pares. Ese bit es la suma de todos ellos, y si hay dos errores en dos bits distinto s se compensa uno con otro.

Otra opción es la matriz de paridad bidimensional. Aquí calculamos las paridades en las filas p or separado y luego en las columnas por otro lado. Aquí podemos encontrar cualquier combina ción de errores. La siguiente opción se llama, suma de comprobación de Internet. El objetivo e s ver el número de bits cambiados en el paquete enviado. Trata una secuencia de enteros com o si fuese de 16 bits, los suma,
calcula el complemento a 1, y se coloca como checksum en UDP/TCP. Cuando llegamos a rece pción comprobamos si este es correcto. Para ello lo calcula y compara con el original. O lo sum a todo en complemento a 1 y si es todo 1s es correcto y si hay un cero existe un error. Sin em bargo, lo más utilizado es la comprobación de redundancia cíclica (CRC). Estos códigos consiste n en considerar los datos como un número binario (D) cogemos un generador, que tenga un ci erto patrón (G) para tratar el número de bits (R) ,de forma que, al concatenarlos con los datos, sea divisible por el generador. De forma que, como el receptor conoce ese G, cuando recibe < D,R> lo divide entre G y si el resto es distinto de 0 hay un error. Este código puede detectar ráf agas de errores menores que r+1 bits.

5.3 Protocolos de acceso múltiple.

Tenemos dos tipos de enlaces múltiples principales:

-Punto a punto: El medio no se comparte. Como PPP, o switch Ethernet y host.

-Difusión(broadcast) El medio se comparte, como antiguo Ethernet, HFC, o enlaces wi-f i.

Tenemos un solo canal de difusión compartida. Existen dos o más transmisiones simultáneas e ntre nodos que pueden provocar interferencias. Si el nodo recibe dos o más señales al mismo t iempo se da una colisión. Esto provoca que el ancho de banda utilizado se desperdicie. Un algo ritmo distribuido puede determinar cómo los nodos comparten el canal y cuando debe transmi tir cada uno. Si lo vemos de forma ideal tendríamos:

a. Cuando sólo hay un nodo que quiere transmitir lo puede hacer a la tasa máxima (R).

b.
Si tenemos M nodos que quieren transmitir, cada uno puede hacerlo a una tasa media de R/M.

c.Si es totalmente descentralizado: no hay relojes para sincronizar ni nodos que coordi nen. Esto simplificaría mucho.

Esto no es así, por ello contamos con los protocolos MAC, vamos a tener tres tipos:

5 3.1Particionamos el canal:

Dividimos el canal en distintas partes. Y a cada nodo le asigna una de las partes. No entra en co nflicto con los otros nodos porque utilizan otros slots. Podemos hacerlo de varias formas: -TD MA: Partimos el canal por tiempo. Accedemos al canal por rondas, cada nodo va a tener un slo

t de longitud fija. Si no utiliza su slot se queda inactivo. Lo malo de esto es que es ineficiente. P or ejemplo: -FDMA: Partimos el canal por frecuencia. El espectro del canal se divide en frecue ncias, y cada estación tiene asignadas una banda concreta. El tiempo de transmisión no utilizad o habrá frecuencias que tendremos inactivas. Estamos desaprovechando velocidad. Pero tene mos durante todo el tiempo una frecuencia reservada. Tenemos también CDMA.

5.3.2 Protocolo de acceso aleatorio.

El canal no se divide y se accede de forma aleatoria al canal. Pueden suceder colisiones.

Cuando un nodo tiene un paquete que enviar lo envía a la velocidad máxima. Si hay alguien má s transmitiendo hay una colisión. Tenemos que detectar colisiones y recuperarnos, para ello co ntamos con estos protocolos:

ALOHA con particiones:

Suponemos que todas las tramas tienen el mismo tamaño (L bits), el tiempo dividido en slots d e igual tamaño (L/R). Los nodos comienzan a transmitir al principio de cada partición, y los nod os están sincronizados. Si hay dos o más tramas que colisionan en un slot, se detecta esa colisi ón. Cuando un nodo obtiene una nueva trama, transmite en el siguiente slot.

Si no hay colisión se envía en el slot siguiente. Si hay colisión retransmite la trama en un slot po sterior con una probabilidad p. Las ventajas es que un único nodo puede transmitir a la veloci dad máxima del canal, que es altamente descentralizado, que solo se sincronizan los slots y qu e es simple. Como desventajas tenemos que puede haber colisiones, slots inactivos, los nodos pueden detectar colisiones en menor tiempo que tardan en transmitir un paquete y hay que si ncronizar los slots. En cuanto eficiencia podemos ver que la fracción de slots satisfactorios en un largo periodo de tiempo es baja. Solo tendremos transmisiones útiles un 37% del tiempo.

ALOHA puro sin particiones.

Es mucho más simple y no hay sincronización. Cuando la trama llega por primera vez, se trans mite inmediatamente. La probabilidad de colisión es mayor que en el caso anterior. Se disminu ye, por tanto, la eficiencia. Los protocolos CSMA
(Carrier Sense Multiple Access) Es un acceso múltiple con sondeo de portadora.

Con múltiple se refiere a mirar si el canal está ocupado o no. Si el canal está desocupado transf iere si no retrasa la transmisión. Hay colisiones porque puede que no hayas oído a uno que ha ya empezado a transmitir. Estas colisiones se deben al retraso en la propagación. Podemos int entar detectar esas colisiones,
en cuanto detectamos la colisión abortamos la transmisión. A esto se se llama Collision Detecti on (CD, detección de colisión). Esto es fácil en redes cableadas pero en redes inalámbricas es c omplicado debido a la fuerza de la señal.

5.3.3 Toma de turnos:

Los nodos toman turnos y tiene que haber alguien que lo coordine y podemos dar paso a que l os nodos que tienen que transmitir más información tengan turnos más largos.

-Los protocolos MAC que particionan el canal y tienen una alta carga del canal compartido lo p ueden hacer de forma eficiente y justa. Pero si tenemos baja carga es ineficiente.

-Los protocolos MAC de acceso aleatorio son eficientes con carga baja pero con alta lo sobreca rgamos de colisiones y su eficiencia es baja.

-Los protocolos MAC de toma de turnos buscan lo mejor de ambas aproximaciones.

-Se utiliza un sondeo (polling), utilizamos un nodo maestro que invita a los nodos escla vos a transmitir por turnos. Las inquietudes son la posible sobrecarga que se puede pro ducir, la latencia y que solo hay un punto de fallo.

-Otra opción es utilizar un Paso de testigo (token): El token es un testigo que se va pasa ndo cada nodo de forma secuencial, en el que tenemos una sobrecarga del token, una l atencia y un único punto de fallo. El token pasa de un nodo a otro y quien lo tenga pue de transmitir datos. Si no tiene datos el token va al siguiente.