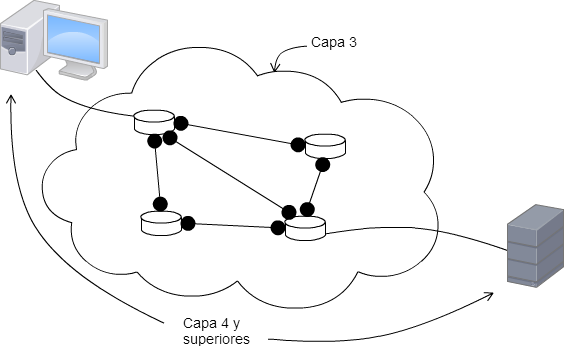
# CLASE 17 DE SEPTIEMBRE

**SERVICIOS PROPORCIONADOS POR LA CAPA DE TRANSPORTE:**

1. Comunicación lógica extremo a extremo:

Dentro de la “nube” se ejecuta hasta la capa 3, la capa 4 y superiores solo se ejecutan en los extremos de la red (PC, Servidores, etc). Cuando se inicia la comunicación entre una máquina y un Servidor Web, se dice que se establece una conexión lógica, no física, extremo a extremo (de la red). La comunicación “lógica” ocurre entre capas del mismo nivel de los extremos. Físicamente esto no es así ya que cada dispositivo va encapsulando y desencapsulando encabezados de las distintas capas según lo necesita, hasta llegar al destino donde la capa 4 de transporte, “entiende” lo que le mandó el origen.

1. Servicio orientado a conexión y sin conexión (TCP – UDP):

La capa de transporte brinda dos tipos de servicio:

* + **Orientado a conexión:** Para que un cliente y un servidor web puedan comunicarse, antes tiene que establecerse una conexión. La capa 4, establece una conexión, transfiere los datos y una vez que ya no se envían más datos, libera la conexión.
  + **Sin conexión:** El receptor no tiene que estar “preparado” para recibir los datos, cuando un cliente necesita enviar datos, solo los envía.

Existen estos dos tipos de servicios, para que las aplicaciones los soliciten de acuerdo a lo que necesitan.

1. Servicio fiable y no fiable:
   * **Servicio Fiable:** Garantiza que los paquetes se entreguen en orden, que el receptor va a recibirlos, que lleguen todos (si se pierden, retransmite). Además, existe control de integridad, acuses de recibo, etc.
   * **Servicio No Fiable:** En este servicio no se garantiza nada, si se pierden paquetes, no se reenvían, ni se garantiza que el receptor los recibirá.

Existe una relación entre el servicio orientado a conexión y el fiable, como también existe entre el servicio sin conexión y el no fiable. Si una aplicación necesita fiabilidad en la red, le solicita a la capa de transporte un servicio “Orientado a conexión y fiable”, el cual se lo brinda el protocolo TCP.

De la misma forma si la aplicación necesita velocidad en la transmisión, solicitará un servicio “Sin conexión y no fiable” proveído por el protocolo UDP.

1. Direccionamiento: (Puertos)

En cada capa existen formas de direccionamiento, en capa 2 se utiliza MAC, en capa 3 se utiliza IP. En capa 4.

Los números de puertos sirven en el caso en que si en una misma máquina hay varias aplicaciones que envían y reciben paquetes a internet, como estos paquetes entran con la misma dirección IP, se utiliza el número de puerto para distinguir a que aplicación corresponde el paquete que acaba de ingresar. Este número de puerto está formado por 16 bits, con lo que puedo obtener 65.535 puertos posibles que se pueden clasificar en 3 tipos (definidos en RFC):

* + - De 0 a 1023  Puertos públicos o “bien conocidos” (well known):
    - De 1024 a 49151  Puertos Registrados
    - 49152 a 65535  Puertos Dinámicos o Privados

1. Multiplexación y demultiplexación:
   * **Multiplexación:** Multiplexar es compartir, en este caso significa que vamos a enviar datos de muchos procesos por la misma placa de red. La capa de transporte “mezcla” los segmentos de distintos procesos, y se lanzan en el mismo medio.
   * **Demultiplexación:** Es la inversa de multiplexar, se ejecuta en el destino. La capa de transporte lo va a demultiplexar en función del número de puerto de destino del segmento que está recibiendo.

**PROTOCOLO TCP (Transmision control protocol):**

Características:

* + Fiable, orientado a conexión
  + Garantiza entrega ordenada, integridad
  + Segmenta los datos entrantes en trozos
  + Reensambla los segmentos
  + Brinda conexiones dúplex y punto a punto
  + Brinda-Colabora con el control de congestión9
  + Implementa control de flujo
  + No implementa Multidifusión

La placa de aplicación le entrega datos a la capa de transporte y ésta lo que hace es segmentarlo en trozos en función del MTU (Maximum Transfer Unity, *del inglés: unidad máxima de transferencia)*, es decir en función de la tecnología de la capa de enlace (p.e; Ethernet soporta 1500 bytes); y lo hace para que cuando estos segmentos lleguen a la capa de Red el protocolo IP no tenga que volver a dividirlo en paquetes más pequeños para poder enviarlos.

Luego de segmentado los datos, les agrega una cabecera que indica el número de secuencia y se lo entrega a la capa inmediata inferior, que es la de red.

En el destino lo que se realiza es, cuando los paquetes de la capa de Red son entregados a la capa de Transporte, ésta desencapsula los segmentos y los une o re ensambla los segmentos para entregarle los datos completos a la capa de Aplicación.

TCP permite enviar datos y recibirlos al mismo tiempo (ya que es dúplex) y como es orientado a conexión, cuando se establece la misma, se establecen dos caminos o canales (sockets) entre el origen y el destino uno para transmitir datos y otro para recibirlos. Debido a esto para cada aplicación/pestaña que se encuentra abierta en una máquina se reservan dos buffers, uno de salida y otro de entrada.

El socket que se establece en este tipo de conexiones full dúplex, tiene cuatro parámetros:

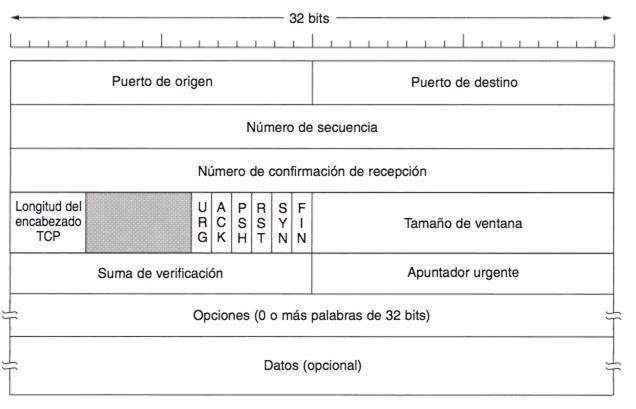
* IP origen
* Puerto Origen
* IP destino
* Puerto Destino

Por esto dicha conexión es única y no deberían existir dos iguales en todo el mundo.

Para explicar las conexiones punto-a-punto tomamos como ejemplo un servidor al cual están conectados tres clientes:

Por cada uno de los clientes se establecen dos sockets, uno de transmisión y otro de recepción. Pero dicho servidor establece una conexión punto a punto para cada uno de dichos clientes. Si el mismo cliente quiere conectarse al mismo servidor, se abre otra conexión punto a punto al mismo servidor, pero con otro socket.

**FORMATO DEL SEGMENTO TCP**

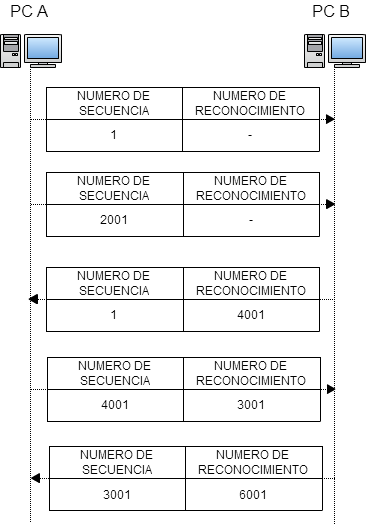


Todos los encabezados se organizan en cabeceras de 32 bits, al igual que IP. Son renglones o palabras de 4 Bytes. Estaban tan seguros cuando desarrollaron este protocolo que tomaron la decisión de no colocarle un campo versión.

Los campos son:

* **Puerto origen:** (16 bits) – Desde o de dónde sale el segmento.
* **Puerto Destino:** (16 bits) – Hacia dónde va el segmento
* **Número de secuencia:** (32 bits) – Como TCP segmenta los datos y para no indicar el tamaño del segmento, lo que se hace es enumerar los bytes. Como tengo 32 bits (y 2 elevado a la 32 es un número muy grande). Supongamos que los segmentos son de 2000 bytes, el primer segmento tendrá en este campo el número 1, el siguiente tendrá el 2001, el siguiente el 4001, y así sucesivamente. Por esto se dice que TCP maneja “corrientes o flujos de bytes”.
* **Numero de confirmación de recepción (o número de reconocimiento):** (32 bits) – Supongamos que se desean enviar una secuencia de segmentos, desde una PC A a una PC B. Es necesario que la PC B envíe un acuse de recibo a la PC A para confirmar que se recibió el segmento. Para que no se envíe un acuse de recibo por cada segmento que se envía de A a B, B le puede enviar un único acuse de recibo por un conjunto de segmentos recibidos, esto lo hace colocando el número del “siguiente byte” que espera recibir de A.

Como B nunca manda un segmento vacío con solamente el acuse de recibo, como la conexión es dúplex, B puede enviar datos a A, y mezclado con esto el acuse de recibo de lo que recibió de A. Veámoslo en un ejemplo:

Supongamos que la PC A tiene que enviarle a la PC B, 3 segmentos con los números de secuencia: 1, 2001, 4001. Además, B un instante después debe enviarle a la PC A, 2 segmentos con número de secuencia: 1 y 3001. Si lo graficamos en el tiempo tendremos que:

* **Longitud del encabezado TCP:** (4 bits) – La cabecera puede tener como 15 renglones de palabras de 32 Bits, lo normal sin opciones ni nada, son 5 renglones.  **4 Bits sin uso**
* **Flags:**
* **URG: (urgente)** Dentro de la parte de datos se puede enviar información de control extra. Se enciende en 1 este bit, para avisarle al origen/destino que se envían datos urgentes, y además se establece el apuntador urgente (ver apuntador urgente a continuación) para conocer en q parte de los datos se encuentra esta información.
* **ACK:** Se activa este bit, cuando el segmento tiene contenido en el número de reconocimiento. Es decir cuando en el segmento se transporta información de acuse de recibo. o **PSH:** (push) se le indica al destino que no lo almacene hasta conseguir más datos. Es decir que se prende este bit para que no se almacene en el buffer del destino, y que apenas llegue se lo envíe a la aplicación.
* **RST:** (Reset) Como TCP es orientado a conexión y se puede cortar, la forma de reanudarla si eso sucede es prendiendo este bit, sirve para resetear la conexión.
* **SYN:** Se prende cuando se quiere iniciar una conexión (para avisarle al otro dispositivo que se quiere iniciar una conexión).
* **FIN:** Se prende cuando se termina la conexión para liberar los recursos que se utilizaron en la misma.
* **ECN:** Sirve para notificar al origen de que la red está congestionada, y que baje la tasa de transferencia.
* **VCR:** (ventana de congestión de red) Cuando se recibe un ECN prendido, se baja la tasa de transferencia pero además se prende este bit para avisarle al destino que ya se bajó la misma. Esto es para que el destino no sigua avisando que la disminuya.
* **Ventana de recepción (tamaño de ventana):** (16 bits) – Se utiliza para colocar el valor del tamaño de buffer que se tiene. Por ejemplo supongamos que A le manda información a B, y como B es lento, llegado un momento se le llena su buffer, y para evitar que los paquetes se pierdan, pone este campo en 0, de esta forma A detiene la transmisión hasta que B se descongestione. Cando a B se le libere su buffer, va a notificar a A, y en este campo le indicará cuanto buffer libre tiene. Este campo se controla en ambos sentidos. Otra forma de definirlo es la cantidad de Bytes que están dando vueltas en la red, sin acuse de recibo.
* **Suma de verificación (Cheksum):** (16 bits) – Sirve para controlar que los datos lleguen correctamente, ya que TCP es un protocolo fiable. Cuando llega al destinatario, calcula el código de redundancia, y lo comprar con el que llega en este campo. Además para controlar la fiabilidad se utiliza el número de secuencia.
* **Apuntador urgente:** (16 bits) – Numero de Byte de datos a partir del cual comienza la información urgente.
* **Opciones:** (0 o más palabras de 32 bits) – Cuando se establece una conexión se define un MSS (Maximum segment size o tamaño máximo de segmento) entre otras cosas, todos las negociaciones que se realizan en el establecimiento de la conexión se envían en este campo. Durante la conexión, no se utiliza.
* **Datos**

**CONTROL DE CONGESTIÓN:**

Todos colaboran al control de congestión de la red, ya que si está congestionada, todos nos perjudicamos. Por ello es una actitud colaborativa no solo por parte de los Routers sino también por parte de los hosts ya que son estos mismos los principales causantes de la congestión. El protocolo TCP tiene varias formas de darse cuenta que hay congestión, una de las formas es que se informe la congestión en los hosts. Si un host recibe un aviso de que hay congestión, lo que hace es bajar la tasa de transmisión a la mitad de la velocidad de lo que se estaba transmitiendo antes.

Existe un campo que no está en el encabezado de TCP que se conoce como “ventana de congestión” la cual sirve para indicar la cantidad de bytes que un emisor puede enviar a la red en un momento determinado.

Además esto lo realiza con la ayuda de un valor conocido como MSS (Tamaño máximo de segmento), que cuando se establece la conexión, cada uno se pone de acuerdo para determinar cuánto va a ser el tamaño máximo de segmento que se va a utilizar y se lo define en ese atributo. La ventana de congestión no sabe si la red está o no está congestionada, para ello envía un segmento e inicia un temporizador, si llega el acuse de recibo antes de que finalice dicho temporizador, entonces la red no está congestionada y aumenta el tamaño del próximo segmento a enviar, esperando que se venzan los temporizadores. Si en algún momento no se recibe un segmento antes de que se termine el temporizador, es decir, que finaliza el temporizador antes de recibir el acuse de recibo (time out), entonces significa que la red está congestionada y para colaborar a la descongestión, reduce a la mitad la velocidad de transferencia actual.

De esta forma se puede observar una diferencia entre la ventana de congestión y la ventana de recepción, en resumen:

 **Ventana de congestión:** Sirve para no saturar de paquetes a la red, es decir colaborar con la congestión.

 **Ventana de recepción:** Se utiliza extremo-a-extremo, y es para conocer si no está congestionado el receptor de los datos.

Entonces si al construir un segmento, se tiene que la ventana de congestión es 3000 y la ventana de recepción es 5000, el tamaño de dicho segmento va a ser el valor mínimo de las dos ventanas ya que en este caso el receptor no está congestionado pero la red si lo está. Si la red comenzara a descongestionarse, y el valor de la ventana de congestión aumentara hasta el valor 10000, significaría que el emisor tiene derecho a lanzar 10000 bytes a la red pero como el emisor es más lento, solo puedo enviar 5000 bytes.

**HANDSHAKE o SALUDO DE TRES VÍAS (inicio de conexión):**

Se intentará iniciar una conexión entre un dispositivo A y otro dispositivo B. Normalmente el que inicia la conexión (en la arquitectura cliente-servidor) es el cliente, pero para esto los servidores deben estar funcionando.

Supongamos que A quiere establecer una conexión con B, para esto le va a enviar un segmento donde los campos relevantes van a ser:

 **Flags:**

* **SYN = 1** (Se quiere establecer la conexión)
* **ACK = 0** (No se recibe nada)

 **Numero de secuencia:** Indica el valor donde comienza a numerar los segmentos, p.e: 1001.

Cuando llega a B, éste se da cuenta que alguien quiere establecer una conexión y la acepta, en este momento es cuando se abre el socket en un sentido (desde A hasta B). Ahora es necesario abrir el socket en el sentido contrario, para esto B envía otro segmento que contiene:

 **Flags:**

* **ACK = 1** (Se recibió el segmento enviado anteriormente) o **SYN = 1** (B también quiere enviar segmentos a A)

**Número de secuencia:** valor inicial de segmento que comenzará a enviar B, p.e.: 8000



**Número de reconocimiento:** Si prendí el ACK, debo enviar datos en este campo, se coloca el valor del siguiente byte esperado, es el valor del número de reconocimiento + 1, ya que se espera el siguiente saludo. Entonces es el 1002.

La conexión es de 3 vías, ya que se debe enviar otro segmento de A  B, para completar el establecimiento de la conexión. Hay que aclarar que el flag SYN sólo se utiliza en los dos primeros mensajes, luego de eso ya no se utiliza. En este tercer segmento, los campos relevantes son:

 **Flags:**

* **SYN = 0**
* **ACK = 1** (si acepta la conexión)

**Número de secuencia:** 1002, (corresponde con el valor del número de reconocimiento anterior). **Número de reconocimiento:** Valor del siguiente Byte esperado, p.e. 8002.



Luego de esto queda establecida la conexión y se comienzan a transmitir los datos. Cuando llega a su fin la transmisión hay que finalizar los sockets iniciados en ambos sentidos, para liberar los buffers y otros recursos utilizados en la conexión. Hay que aclarar que la conexión puede ser finalizada por cualquiera de los extremos, supongamos que A quiere finalizar la conexión, entonces envía un segmento con el flag FIN en 1. En este momento B debe enviar un acuse de recibo, pero también debe finalizar la conexión en el otro sentido. Para esto B conforma un segmento que envía a A, también con el flag FIN en 1, y A debe enviarle otro acuse de recibo a B, hecho esto se da por terminada la conexión entre ambos.

# CLASE 1 DE OCTUBRE

**SOCKETS (PRIMITIVAS):**

Cuando quiero establecer una conexión desde un punto a otro, debo abrir un canal de comunicación a lo que se lo conoce con el nombre de socket. Es una conexión que queda identificada por una dupla IP + Puerto.

Hay dos tipos de sockets:

 **Pasivos:** Estos sockets se configuran en los servidores, ya que estos dispositivos tendrán una actitud pasiva al estar esperando que se inicien conexiones desde los clientes.

 **Activos:** Estos sockets se configuran en el cliente, ya que es quien inicia la conexión.

**Tabla de socket:**

Cuando se inicia un socket el sistema operativo reserva muchos recursos, entre los cuales crea una tabla. En función de cómo se vaya completando la tabla vamos a definir si estamos configurando un socket activo o un socket pasivo. La tabla se muestra a continuación:

|  |
| --- |
| **TABLA DE SOCKET** |
| **Familia** |
| **Tipo** |
| **IP\_Local** |
| **Puerto\_Local** |
| **IP\_Remoto** |
| **Puerto\_Remoto** |

* **Familia:** Al crearse los sockets la idea era que pudieran ejecutarse con cualquier pila de protocolos de transporte, no obstante en la actualidad el único que se está usando es TCP/IP. De todas maneras, para que sea totalmente flexible, es necesario indicar que familia de protocolos va a utilizar.
* **Tipo**: Tipo de servicio que se va a ofrecer (orientado a conexión o no orientado a conexión) **Primitivas de comunicación:**

Las primitivas de comunicación hacen referencia a las operaciones que se utilizan para implementar los sockets. Existen primitivas que se ejecutan en el cliente, otras en el servidor y algunas en ambos. A continuación se detallan los parámetros que hay que pasar al momento de crear un socket: **Descriptor de socket = socket (familia, tipo, protocolo)**

* **Tipo:** Al crear un socket hay que indicar el tipo de protocolo de capa de transporte sobre el que va a correr el socket. Los tipos pueden ser: sockets fiables que corren sobre TCP (Stream) o sockets no fiables que corren sobre UDP (Datagram).

Al crear el socket se genera un descriptor de socket que es un número que utiliza el sistema operativo cuando quiere hacer referencia a un socket en particular.

**Primitivas para el servidor (Operaciones):**

**Socket:**



**Bind (nro\_socket, ip\_local + puerto\_local):**Sirve para asociar una dirección IP + un puerto al socket. El numero de socket que hay que especificar hace referencia al descriptor del socket (es el mismo número). La ip local y el puerto local que le asociamos al socket significa que los clientes van a poder acceder al mismo a través de esa IP y ese puerto, ya que la forma de identificar de manera única a un socket es a través de dicha dupla. Para IPv4 se necesitan en total 48 bit: 32 corresponden a la dirección IP y 12 al puerto. Con esta información se va llenando la tabla de socket del sistema operativo.

 **Listen (nro\_socket, longitud\_cola):** Sirve para que el servidor este atento a ver si le llegan conexiones, para lo cual se especifica el número de socket o descriptor, y además la longitud de la cola. Esto es necesario porque el servidor tiene abierto un socket principal a través del cual se receptan las conexiones de los clientes. Cuando llega una petición de un cliente al socket lo que éste tiene que hacer es procesarla creando otro socket, pero puede ocurrir que mientras el socket receptor está haciendo esta labor lleguen nuevas peticiones de conexión de los clientes, por lo tanto (y para que no se pierdan) es necesario que esas peticiones sean encoladas. De esta manera, la especificación de la longitud de cola hace referencia a la cantidad de peticiones de conexión que pueden estar pendientes en la cola en un momento determinado.

 **Accept (nro\_socket, ip\_remota, puerto\_remoto):**Dijimos que cuando le llega una petición de conexión del cliente, el servidor tiene que procesar esa petición y crear un nuevo socket, lo que se hace con la primitiva accept. Es como si de la cola de peticiones el servidor tomara alguna petición, y creara un nuevo socket al que le asocia una IP Local + Puerto Local e IP Remota + Puerto Remoto. Es decir que va a completar todos los datos de la tabla porque ese socket va a quedar abierto entre el cliente y el servidor. Como la petición va a provenir de un cliente ese necesario que la IP y el puerto especificado sean las remotas, ya que para el servidor estos datos son los del cliente. Una vez hecho esto la tabla queda completa y puede comenzarse a transferir información entre el cliente y el servidor pero sobre otro socket. A nivel de sistema operativo, con cada conexión nueva que se establece se abre un hilo de conexión diferente.

 **Read/Receive (nro\_socket, buffer, tamaño\_buffer):** Una vez que está establecida la conexión el cliente comienza a hacer peticiones, por lo que el servidor se dispone a leerlas, es decir que el cliente va a enviar datos y el servidor los va a recibir. Va a leer lo que este almacenado en el número de socket especificado. Buffer hace referencia a la dirección de memoria de donde se van a leer los datos almacenados (lugar físico donde está almacenado el buffer), mientras que tamaño de buffer hace referencia a la cantidad de datos a leer.

 **Write/Send (nro\_socket, buffer, tamaño\_buffer):**Esta primitiva se ejecuta generalmente después de un read, ya que el servidor tiene que enviar la información que busco a partir de la petición solicitada a través del número de socket especificado. Para esto, el servidor toma los datos del buffer de salida y especifica también el tamaño de dicho buffer.

 **Close (nro\_socket):** Mientras está establecida la conexión el servidor lee y escribe en muchas oportunidades hasta que ya no hay datos que transmitir y llega el momento de cerrar la conexión. En este caso es necesario ejecutar la primitiva close, cerrando el socket indicado a partir de su correspondiente descriptor o numero. Cuando esto ocurre se liberan recursos y espacios en la tabla del sistema operativo.

**Primitivas para el cliente:**

**Socket**



**Connect (nro\_socket, ip\_remota, puerto\_remoto):**Se conoce como una petición de conexión activa, y sirve para establecer una conexión contra un servidor. Es necesario especificar el número de socket con el cual se desea establecer la conexión, junto con IP Remota y el Puerto Remoto que vistas desde el cliente son las del ¿SERVIDOR?

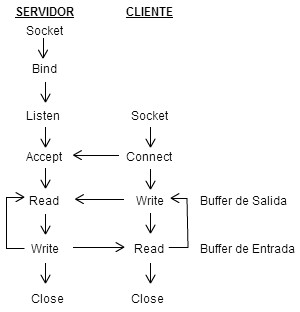
**Read/Receive (nro\_socket, buffer, tamaño\_buffer)**



**Write/Send (nro\_socket, buffer, tamaño\_buffer)**  **Close (nro\_socket)**

Como la conexión es del tipo dúplex es necesario que se abran dos sockets, uno hacia un lado y el otro hacia el otro, de los cuales unos será pasivo y el otro activo.

**Diagrama de relación de primitivas:**



Nótese que antes de que el cliente pueda comenzar a comunicarse tienen que establecerse varias primitivas. Lo primero que hace el cliente es ejecutar un connect, el cual es aceptado por el servidor. Luego el cliente escribe lo que necesita pedirle al servidor, mientras que este último lee las peticiones. En función de esto el servidor busca los datos necesarios y se los entrega al cliente, es por ello que existen dos buffers: uno de entrada y uno de salida de datos.

**UDP (User Datagram Protocol – Protocolo de Datagrama de Usuario) Características:**

 **No orientado a conexión:** Significa que cuando la aplicación tiene datos para transmitir no es necesario que pase por la fase de establecimiento de la conexión, sino que directamente encapsula los datos en un segmento, le agrega un encabezado de 8 bytes, se lo pasa a la capa de IP en donde se encapsula en un paquete y de ahí sale el mensaje.

 **No fiable:** Significa que no le importa si los mensajes llegan o no llegan al destino, el protocolo simplemente se encarga de enviar los datos lo más rápido posible. No existen retransmisiones en caso de que un paquete no llegue al destino.

 **Baja sobrecarga:** Significa que es muy rápido porque no es necesario reservar recursos como variables, ventanas de congestión, temporizadores, etc.

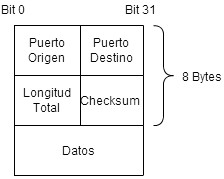
**Rápido**



**Usado por aplicaciones en tiempo real:** Por ejemplo cuando se desea hacer una consulta a una base de datos, donde se requiere obtener la información lo más rápido posible. También se utiliza para voz sobre ip, videoconferencias, etc.

En caso de que se necesite que la aplicación sea rápida pero fiable, es decir que los datos lleguen rápidamente pero también ordenados y todos (que no se pierda nada), es necesario un hibrido entre TCP y UDP. Para lograr esto se debe utilizar el protocolo UDP para garantizar la rapidez de transmisión pero a su vez la aplicación deberá gestionar todo el control necesario de la información (el desarrollador será el encargado de garantizar esto último). Cabe mencionar que en este caso no se puede tener una fiabilidad completa, pero si se puede establecer un control mínimo que ayude con el propósito y no consuma muchos recursos.

**Formato del Segmento:**



Nótese que la cabecera es muy sencilla (presenta poco campos) debido a que no existe control alguno, por lo que es muy rápido crearla (esto se conoce como baja sobrecarga). Se utilizan palabras de 32 bits. En la longitud total se especifican la cantidad de bytes que va a tener el segmento completo, y como este campo tiene 16 bits significa que puedo tener un tamaño máximo de 64 Kb aproximadamente. En el campo checksum se hace un cálculo consistente sumar palabras de 16 bits y al resultado de esa suma se le calcula un complemento a 1 (donde hay un cero se pone un uno, y viceversa), y el valor obtenido es el que va en el campo. El checksum se hace sobre todo el segmento, incluidos los datos, y esto se hace para que cuando el destino recibe un segmento pueda determinar si hubo algún error en la transmisión de los datos, y de ser así lo descarta sin avisarle a nadie.

**Aplicaciones que usan UDP:**

Las aplicaciones que utilizan este protocolo son:

 **DNS:** El objetivo de un DNS es dado un dominio obtener rápido la dirección IP asociada a dicho dominio para establecer una conexión, lo que requiere un acceso a una base de datos. Utiliza el puerto 53.

 **RIP:** Este es un protocolo de encaminamiento que envía actualizaciones de encaminamiento que son encapsuladas en un segmento UDP para que sea más rápido. Si esa actualización se pierde no pasa nada porque a los 30 segundos vuelve a enviar otra (no es necesaria la retransmisión). Utiliza el puerto 520.

 **SNMP (Simple Network Management Protocol – Protocolo Simple de Gestión de Red ):** Es un protocolo de gestión de red. Utiliza el puerto 169 **TFTP:** Utiliza el puerto 69.



**BootpS:** La S hace referencia a Server. Escucha en el puerto 67

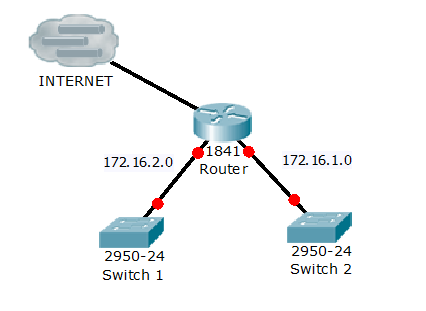
**BootpC:** La C hace referencia a Client. Escucha en el puerto 68. El mensaje DHCP Discover se encapsula en un segmento UDP con el puerto 68 como origen y el 67 como destino.

# CLASE 8 DE OCTUBRE

**Traducción de direcciones de Red:**

**NAT(network adress translation):**

Supongamos el siguiente escenario



200.0.0.1

Mientras haya tráfico entre la subred 172.16.1.0 (desde ahora subred 1) y la subred 172.16.2.0 (desde ahora subred 2) no hay traducción de direcciones ya que el tráfico es dentro de la red. La traducción de direcciones IP se produce cuando hay tráfico entre algún dispositivo de la subred 1 o la subred 2, hacia internet. En este caso el router está pre-configurado para que no salgan al exterior paquetes con direcciones de IP privadas, es decir que se produce una traducción en dos situación: tanto cuando sale un paquete hacia internet como cuando viene un paquete de internet y entra a la red local.

Ventajas: La ventaja principal es que desde el exterior a la empresa no pueden ver la topología de la red, y no se puede acceder a una dirección de ip privada.

Otra ventaja es la seguridad, es decir que al tener acceso a internet en un solo punto, puedo configurar un firewall que indica que paquetes pueden salir y entrar, y que paquetes no.

Desventaja: La principal desventaja es que hay que hacer traducciones tanto cuando sale un paquete como cuando entra, entonces en el router se produce un “cuello de botella” porque todo el tráfico entrante y saliente pasa por él. Lo que provoca que se pierda mucho tiempo.

**Tipos de traducciones NAT:** Todas las traducciones NAT se realizan a través de tablas, estáticas o dinámicas.

* **NAT Estático:** Se produce una traducción 1 a 1, es decir que por cada dirección de IP privada voy a necesitar una dirección de IP pública. Este tipo de traducción se utiliza generalmente para servidores, y esto se debe a que de otra forma estos servidores no serían visibles desde el exterior. El NAT del router se configura de tal forma que en la tabla de direcciones, la dirección de IP privada que pertenece al servidor siempre va a verse desde el exterior con una determinada IP pública. Una desventaja de este tipo de traducciones es que no se ahorran direcciones de IP.
* **NAT Dinámico:** Es similar al DHCP. En este caso se definen un conjunto (pool) de direcciones públicas para realizar las traducciones. Por ejemplo, suponiendo que tengo 100 direcciones privadas en la empresa y se asigna un pool de 40 direcciones públicas, entonces cada vez que un dispositivo con IP privada quiere salir a internet, el NAT del router va a asignar alguna dirección del pool de direcciones públicas al dispositivo, registrando esta asociación en la tabla NAT. El proceso se repetirá con todos los dispositivos hasta agotar las 40 direcciones públicas que tenía disponibles.

La principal desventaja es que puede haber como máximo 40 dispositivos conectados a internet al mismo tiempo. Cuando un dispositivo ya no necesita conectarse a internet, la dirección de IP pública que utilizaba se libera, es decir que ser elimina la asociación de la tabla NAT, quedando disponible para otro dispositivo que requiera conexión. Otra desventaja es que si en un momento determinado desde internet se requiere acceder a un dispositivo que no tiene una asociación hecha en la tabla NAT no se podrá acceder a él, siendo posible solo si previamente el dispositivo solicitó conexión a internet.

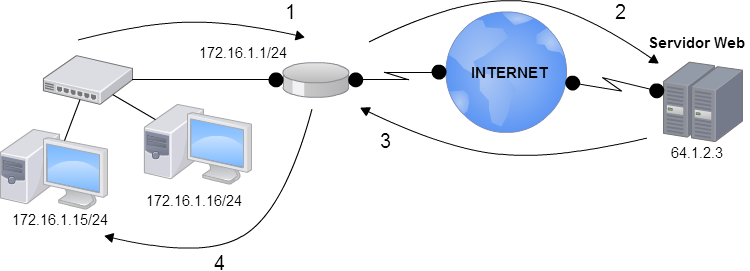
* **PAT (Port Address Translation – Traducción de direcciones de puerto):** Hoy en día cuando se quiere acceder a internet ya no da “ocupado”, esto se debe a que existe un agotamiento de las direcciones IP se propuso la idea de hacer traducciones a nivel de puerto, es decir a nivel de capa de transporte.

Los números de puertos sirven en el caso en que si en una misma máquina hay varias aplicaciones que envían y reciben paquetes a internet, como estos paquetes entran con la misma dirección IP, se utiliza el número de puerto para distinguir a que aplicación corresponde el paquete que acaba de ingresar. Este número de puerto está formado por 16 bits, con lo que puedo obtener 65.535 puertos posibles que se pueden clasificar en 3 tipos (definidos en RFC):

* + - De 0 a 1023 🡪 Puertos públicos o “bien conocidos” (well known):
    - De 1024 a 49151 🡪 Puertos Registrados
    - 49152 a 65535 🡪 Puertos Dinámicos o Privados

Por otro lado, la forma de describir una comunicación única y distinguible en internet es mediante un número que se conoce como “socket”. El socket está formado por una dupla IP + Puerto. Nunca existirán dos sockets que sean iguales.

En conclusión, PAT modifica la dirección IP (privada/publica) y el puerto (privado/público). Esto es útil ya que si una empresa tiene solamente una dirección de IP pública para “salir” a internet, todas las comunicaciones desde dentro de la empresa utilizarán solo esa dirección, sin tener problemas con la cantidad de host que pueda haber. Una desventaja es que el router que es un dispositivo de capa 3, al utilizar PAT modifica el puerto, metiéndose en la capa 4 de OSI, lo cual para algunos esto no debería ser así.

Veamos el siguiente gráfico cual es el comportamiento de 4 paquetes que entran y salen de la red LAN o de un Servidor Web.

Analizaremos los paquetes a continuación, donde se muestran los encabezados de paquete y segmento. Además los datos se encapsulan en una trama, con una dirección MAC que es obtenida del destino mediante ARP.

Paquete 1: De host a router

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  | | --- | | Paquete (IP) | | | Segmento (puerto) | |  |
|  |  |  |  |  |
|  |  |  |  |  |
| 172.16.1.15 | 64.1.2.3 | 6000 | 80 | DATOS |
| *Origen* | *Destino* | *Origen* | *Destino* |  |

Paquete 2: del router de la empresa a internet

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Paquete (IP) | | Segmento (puerto) | |  |
| |  | | --- | |  | |  | |  | | --- | |  | |  |  |
|  |  |  |  |  |
| 201.4.6.8 | 64.1.2.3 | 6000 | 80 | DATOS |
| *Origen* | *Destino* | *Origen* | *Destino* |  |

Paquete 3: del servidor al router

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Paquete (IP) | | Segmento (puerto) | |  |
| |  | | --- | |  | |  | |  | | --- | |  | |  |  |
|  |  |  |  |  |
| 64.1.2.3 | 201.4.6.8 | 80 | 6000 | DATOS |
| *Origen* | *Destino* | *Origen* | *Destino* |  |

Paquete 4: del router al host

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Paquete (IP) | | Segmento (puerto) | |  |
| |  | | --- | |  | |  | |  | | --- | |  | |  |  |
|  |  |  |  |  |
| 64.1.2.3 | 172.16.1.15 | 80 | 6000 | DATOS |
| *Origen* | *Destino* | *Origen* | *Destino* |  |

Supongamos ahora que una maquina tiene dos procesos contra el mismo servidor, es decir se accede a una página web, desde dos ventanas distintas para realizar tareas distintas. Si bien son de la misma PC para el SO son dos procesos distintos, entonces en ese caso en la tabla PAT del router vamos a tener la misma IP privada para ambos procesos, pero cada uno tendrá un puerto distinto para distinguirlos.

Otro caso que podría plantearse sería el de dos PC distintas que se encuentran dentro de una misma red, para las cuales se genera el mismo número de puerto para un proceso. En este caso, lo que hace el router es controlar que no exista un socket idéntico al que se quiere generar y si fuera así le cambia el número de puerto, es decir lo “traslada”.

Una desventaja de PAT es que si yo tengo una sola IP pública para salir a internet, y quiero tener 70000 servidores no puedo, a pesar de que tengo 65536 puertos, es demasiado procesamiento y no llegaría, se saturaría antes. La solución a esto es solicitar al ISP otra dirección pública y se le hace un overload o sobrecarga a 2 direcciones públicas.

**DNS (Domain Name System – Sistema de Nombre de Dominio) Características:**

 **Aplicación:** Es una aplicación que tiene como función principal brindar la traducción de nombres de dominio a direcciones IP. Esto es muy importante ya que, para navegar, es mucho más fácil para el usuario recordar nombres de dominios que las direcciones IP asociadas a esos dominios. Por lo tanto, cuando deseamos ingresar a alguna pagina, lo único que hacemos es escribir el dominio de esa página en el navegador y luego la computadora, a través del DNS, se encargara de obtener la dirección IP correspondiente del servidor que aloje a dicha pagina. Una característica de este protocolo sirve como base para el resto de las aplicaciones, es decir que brinda servicios a todas las aplicaciones de Internet; si no existiera, estas aplicaciones no se podrían ejecutar.

**Funcionamiento:** Cuando escribimos el nombre de dominio en un navegador (que es un cliente del protocolo HTTP) el protocolo HTTP lo que hace es construir el segmento y el paquete. Para construir el segmento no hay ningún problema porque sabe que el puerto origen corresponde al propio de la maquina, mientras que el puerto destino es el 80. Al momento de encapsular este segmento en un paquete es necesario indicar la IP origen, que se tiene porque es la que utiliza la Pc cliente, y la IP de destino que es la que estamos tratando de averiguar ya que nosotros ingresamos un dominio en el navegador y no una dirección IP. Por lo tanto, el protocolo HTTP le pide al DNS que le averigüe que dirección IP se corresponde con el dominio ingresado. Una vez que el dominio averigua este dato se lo pasa al protocolo HTTP quien construye el paquete completándolo con la IP de destino y lo lanza por la red. Por ello se dice que el DNS sirve de base a todas las otras aplicaciones, ya que por debajo de un correo electrónico o de un navegador siempre está el protocolo DNS traduciendo.

 **Es una base de datos distribuida (Como se implementa):** En un comienzo el DNS consistía en un archivo que tenía muchas entradas del tipo dominio/IP. Al tener una base de datos centralizada con estas asociaciones, llego un momento (generado por el crecimiento de internet) en donde esto ya no fue posible de mantener debido al gran volumen de datos. Por ello, se invento el DNS que físicamente se puede representar como una base de datos distribuida en muchos servidores esparcidos a lo largo de todo el mundo. Esto no significa que los datos estén replicados a lo largo de todo el mundo, sino que cada uno de los servidores tiene una parte de la base de datos.

 **Corre sobre UDP:** Una consulta DNS es un acceso a base de datos, y un acceso a base de datos se implementa mucho más rápido sobre UDP que sobre TCP. Utiliza el puerto 53.

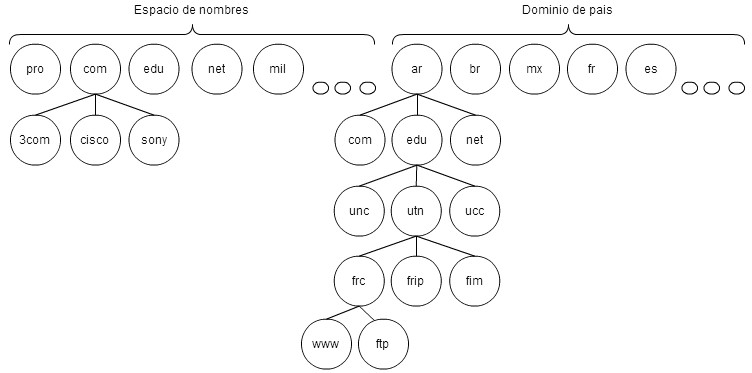
 **Distribución de carga:** Para los correos electrónicos, por ejemplo gmail, existen más de un solo servidor que les da servicio, que se conocen como servidores replicados. Es decir que físicamente los correos de gmail están en distintas maquinas. Esto implica que para un mismo dominio, pueden haber muchas direcciones IP diferentes, por lo que la base de datos del servidor DNS tendrá como entradas:

[www.gmail.com](http://www.gmail.com/)IP1 [www.gmail.com](http://www.gmail.com/) IP2

[www.gmail.com](http://www.gmail.com/) IPN

Esto significa que cuando una maquina le haga una consulta al servidor DNS, éste va a buscar todos los registros que coinciden con el dominio [www.gmail.com](http://www.gmail.com/) y le va a responder con todas las direcciones IP que tiene asociadas para ese dominio. El navegador recibe las tres direcciones IP pero va a utilizar siempre la primera que recibe (IP1). De esta manera, si el DNS siempre respondiera con todas las direcciones y en ese mismo orden, todas las consultas irían (y serian atendidas) por el mismo servidor de gmail físico y el resto no tendrían procesamiento alguno. Por lo tanto, para evitar esto, el DNS hace lo que se conoce como distribución de carga, que consiste en rotar para cada respuesta el orden en el que presenta las direcciones IP, de manera que las consultas se van “distribuyendo” entre los distintos servidores físicos.

 **Arquitectura cliente-servidor** **Espacio de nombres:**



También conocido como árbol de dominio. Todos cuelgan de un dominio raíz. En el primer nivel que se ve existen diferentes dominios: dominios genéricos y dominios de país. Los primeros fueron creados en un comienzo cuando internet se usaba mayormente en Estados Unidos, entre ellos podemos destacar .com, .edu, .net, .pro (profesionales),etc. que físicamente se encuentran en Estados Unidos. Estos dominios constan generalmente de 3 letras, pero en la actualidad se están empezando a crear dominios de 4 letras. Una vez que Internet comienza a expandirse se crean los dominios de países, los cuales expresan con dos letras la extensión del país al que pertenecen. La organización que maneja los dominios de este nivel se llama ICANN, que se encargan de garantizar que no haya dominios repetidos para dicho nivel.

Dentro de un dominio de país, cada país puede organizarlo como desee. La mayoría lo que hizo fue replicar la estructura de los dominios genéricos utilizados por Estados Unidos en el segundo nivel, para realizar la distinción entre paginas de educación, comercial, militar, etc. Como se ve, el árbol de dominio es una estructura jerárquica, con la salvedad de que va desde lo general hacia lo particular. Esto implica que para escribir un dominio el árbol debe leerse desde abajo hacia arriba separando cada nivel por un punto, por ejemplo: [www.frc.utn.edu.ar](http://www.frc.utn.edu.ar/)

Como el árbol de dominios es muy extenso lo que se hizo fue dividirlo en zonas, por ejemplo toda la parte de educación puede representar una zona. Estas zonas se denominan no superpuestas o no traslapantes, esto significa que cada zona se responsabiliza por los dominios que comprende, es decir que las correspondencias dominio/IP de cada zona va a estar físicamente almacenada en un mismo servidor físico. Supongamos que el servidor antes mencionado se destruye, esto provocaría que nadie pueda resolver consultas de la zona del árbol que comprende ese servidor. Por este motivo se requiere que siempre, como mínimo, haya dos servidores: uno primario y otro secundario, de manera que si uno se cae el otro le de respaldo para resolver las consultas.

**Tipos de servidores DNS:**

Como la aplicación del DNS es compleja maneja distintos tipos de servidores:

 **Servidor local:** Es el servidor que está más próximo a los usuarios que van a enviar las peticiones de traducción. Todos los usuarios deben tener configurados un servidor local en su computadora, que se conoce como servidor DNS primario y servidor DNS secundario. Si el usuario no tiene configurado este parámetro no va a poder navegar porque nadie realizara las traducciones de direcciones que necesita. Este servidor maneja una cache de DNS en donde se encuentran algunas traducciones que ya fueron realizadas, y en caso de que el usuario necesite realizar nuevamente una conversión para un dominio al que ya accedió previamente, el servidor revisara primero en su cache para satisfacer dicha traducción. En caso de que el servidor local no tenga en su base de datos o en su cache la traducción para la página a la que desea acceder debe comunicarse con un servidor raíz.

El servidor local para un usuario que accede a internet a través de una empresa a la que le contrata el servicio es el servidor que le ofrece el ISP. Algunas empresas o facultades grandes tienen su propio servidor local.

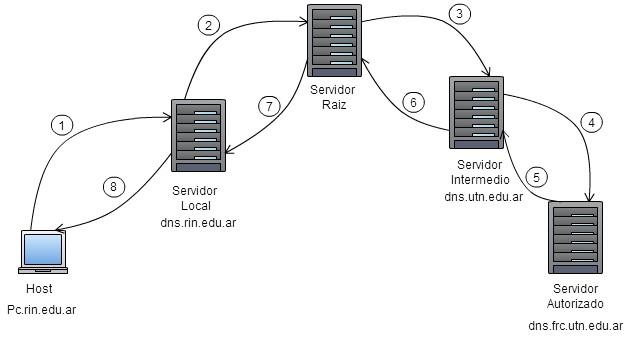
 **Servidor raíz:** En el mundo existen 13 servidores raíz. Los servidores raíz se llaman **a-rootservers**¸ reemplazando la primer letra para distinguir entre cada uno de ellos.

 **Servidor intermedio:** Es un servidor que no tiene la resolución a la petición que se le hace, pero conoce que servidor puede tenerla, y le pasa al usuario que hizo la petición la dirección IP de un servidor autorizado de nombres.

* **Servidor autorizado de nombres:** Todo host o empresa debe estar registrado en un servidor autorizado de nombres, que es el que contiene físicamente almacenado en su base de datos la información fuente o original de las asociaciones dominio/IP. Como mínimo debe haber dos. Hay muchas empresas que son servidores locales y servidores autorizados, es decir que estos servidores pueden estar físicamente en la misma empresa.

**TIPOS DE CONSULTAS (FUNCIONAMIENTO DEL DNS):**

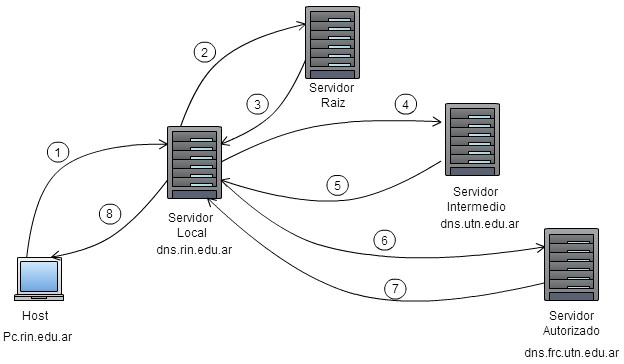
* **Recursiva:**

Supongamos que tenemos una Pc que emite una consulta o quiere comunicarse con un servidor a través de escribir en un navegador [www.frc.utn.edu.ar,](http://www.frc.utn.edu.ar/) es decir que el navegador comienza una consulta DNS. Dijimos que el protocolo DNS tiene una estructura cliente-servidor, en donde para este caso la maquina host va a ser el cliente y el servidor DNS el servidor. El primer servidor que se consulta es el servidor local, a quien se le pregunta si conoce el dominio [www.frc.utn.edu.ar.](http://www.frc.utn.edu.ar/) El servidor local actúa consultando en su cache si tiene la asociación correspondiente a tal dominio, y en caso de que así sea se la devuelve al host quien se lo pasara a la aplicación que lo solicito (el navegador). En el caso en el que el servidor local no tenga en su cache la asociación correspondiente al dominio ingresado lo que hace es detener el proceso y colocarse él como cliente para averiguar la dicha relación. Para ello le envía una consulta o petición DNS al servidor Raíz para ver si tiene la asociación referida a la pagina ingresada. El servidor Raíz consulta si posee dicha respuesta, y en caso de ser positivo se la devuelve al servidor local, o caso contrario, el servidor Raíz comienza a funcionar como cliente del servidor intermedio para averiguar los datos. Entonces, el servidor Raíz le envía la petición correspondiente al servidor intermedio quien toma la misma postura que los servidores antes mencionados. Como se puede ver, la idea que persigue este tipo de consulta es que cuando se le hace una consulta a un servidor DNS, si ese servidor no posee la respuesta se encarga de conseguirla colocándose como cliente de la comunicación.

Una vez que llega la consulta al servidor autorizado de nombres, que es quien realmente tiene en su base de datos la resolución dominio/IP, el mismo la busca de su base de datos y se la pasa al servidor intermedio, quien a su vez se la reenvía al servidor Raíz, y así hasta llegar finalmente al Host.

Las consultas recursivas presentan la desventaja de que cada vez que llega una petición es necesario poner el proceso en espera (es una tarea pendiente) e iniciar otra consulta, esperar a que se brinde la respuesta para recién ahí poder responder al host. Por lo tanto se tarda mucho tiempo porque se tienen que pasar muchos mensajes, 8 pasos para ser más específicos.

 **Iterativa:**

La característica de este tipo es que cada servidor DNS tiene su propia cache, de manera que una vez que consigue una relación dominio/IP si otro Host vuelve a realizar la misma consulta más tarde, directamente toma la relación de su propia cache disminuyendo los tiempos de respuesta. Puede demorar la primera vez que se haga una consulta, pero luego al reiterarse se demorara mucho menos.El funcionamiento es similar al anterior: si se quiere ingresar a una página, por ejemplo [www.frc.utn.edu.ar,](http://www.frc.utn.edu.ar/) el host le envía una petición al servidor local solicitando la relación dominio/IP correspondiente. Suponiendo que es la primera vez que se consulta esta página, y por lo tanto, el servidor local no tiene dicha relación en su cache, lo que hace es enviar la petición correspondiente al servidor Raíz (en este tramo la consulta se comporta de manera recursiva). En este caso el servidor raíz no posee la respuesta a la petición, pero en lugar de ser él quien intente encontrarla lo que hace es darle la dirección IP del servidor Intermedio que puede tener dicha solución. El servidor raíz sabe que el servidor intermedio puede tener la respuesta porque busca una coincidencia en el dominio de la dirección que le pasan. Ahora el servidor local hace una consulta iterativa al servidor intermedio, preguntándole si tiene la relación dominio/IP buscada. Suponiendo que el servidor intermedio no tiene dicha solución, lo que hace es mandarle una respuesta con la dirección IP del servidor autorizado que efectivamente posee la solución. Ahora sí, al realizarle la petición el servidor local (que si tenía una consulta recursiva pendiente) al servidor autorizado podrá obtener la traducción buscada para luego pasársela al host que la solicito.

La ventaja que presenta este tipo de consulta es que los servidores una vez que responden a la petición que le hacen se desentienden, descartando de esta manera el problema que ocurre con la consulta recursiva donde los servidores quedan pendientes a la respuesta. La idea es que los servidores raíz no ejecuten nunca una consulta recursiva debido a que hay tan solo 13 y podrían saturarse.

**Base de datos del DNS (registros de recursos):**

La base de datos de los servidores está dividida en registros, donde cada registro tiene 4 campos:

 **Nombre, tipo, valor, ttl, clase**

**Clase:** Cuando se creó el DNS se pensó que el protocolo debía servir no solamente para TCP/IP, sino que si llegara a surgir otra arquitectura de protocolos también debería funcionar con esta. Por ello en realidad son 5 los campos, solamente que en la actualidad todos los registros poseen en el campo clase el valor “IN” de internet. Por este motivo se dice que el protocolo DNS es muy flexible.

**Nombre:** Este campo va a depender del campo Tipo. Para el nombre de dominio, entre punto y punto, puedo tener como máximo 63 caracteres, y como máximo el dominio puede tener 255 caracteres.

**Valor:** Este campo va a depender del campo Tipo.

**TTL:** Este campo se mide en segundos, que corresponden a todos los segundos de un día (86400 segundos). El TTL representa la vida útil de cada registro que averigua un servidor. Cuando un servidor local averigua una asociación dominio/IP la guarda en su cache, lo que se denomina una resolución no autoritativa. Esto quiere decir que cuando un usuario le vuelve a hacer una consulta sobre el dominio averiguado al servidor local, este lo va a resolver desde la cache, pero teniendo en cuenta que estos datos no son los originales porque el realmente no accedió a la base de datos. Por este motivo, dicho registro en la cache debe tener un tiempo de vida pequeño para que se borre de la memoria en un tiempo razonable, debido a que el servidor autorizado puede llegar a cambiar la IP de la asociación conseguida quedando obsoleta con el paso del tiempo.

**Tipo (tipos de recursos):**

 **A (Address – Dirección):**Si un registro el del tipo A significa que dado un dominio obtengo una IP. En el campo nombre va a ir el dominio, mientras que en el campo valor va a ir la dirección IP de dicho dominio. Este tipo se utiliza para dominios que tienen asociadas una dirección IPv4, para aquellos que utilicen una dirección IPv6 se utiliza otro tipo de registro conocido como AAAA.Ejemplo: [www.frc.utn.edu.ar,](http://www.frc.utn.edu.ar/) A, 200.3.4.5

 **NS (Name Server – Servidor de nombre):** Para un dominio obtengo el nombre del servidor DNS de ese dominio, que básicamente es quien puede tener la IP de todo dominio que termine con el expresado.

Ejemplo: utn.edu.ar, NS, dns.utn.edu.ar

Además, como yo necesito obtener la IP para el dominio establecido, va a existir otro registro que contenga dicha información. Por ejemplo: dns.utn.edu.ar, A, 201.1.2.3

 **MX (Mail Exchange – Intercambio de Correo):** Hay muchas empresas que en el mismo dominio tienen un servidor de correo, un servidor ftp, servidor web etc. Por lo tanto para un dominio determinado puedo obtener cual es el servidor de dominio de correo asociado. Ejemplo: utn.edu.ar, MX, mail.utn.edu.ar

 **CNAME (Connonical name):** Para un dominio obtengo el nombre real del dispositivo o maquina.

Ejemplo: utn.edu.ar, CNAME, colorado.utn.edu.ar

 **SRV (Service):** me permite expandir la base de datos para otro tipo de servicio (servidor de correo, servidor FTP, servidor Web, etc.)

 **HINFO (Host Information – Información del Host):** Me permite obtener que sistema operativo se ejecuta en un determinado servidor, quien es el administrador de la maquina, cual es el mail para contactar al administrador, etc.

 **AAAA:** Este campo se creó para reservar en la base de datos 128 bits en el caso de que haya una dirección de IPv6 relacionada con un dominio.

 **PTR (Pointer)**

En cada registro puedo almacenar mucha información diferente de un mismo dominio. Es decir que para un dominio pueden existir muchos registros asociados. Cada vez que a un servidor le llega una petición de consulta, el DNS consulta en su base de datos y devuelve todos los registros relacionados con ese dominio.

Para poder visualizar todos estos datos podemos ejecutar el comando **NSLOOKUP** que permite consultar los registros de la base de datos de los servidores DNS. Si nosotros ejecutamos **NSLOOKUP <dominio>** nos va a devolver las IP que tiene ese dominio. Al ejecutar NSLOOKUP Enter entramos en el modo de la aplicación NSLOOKUP y a partir de ahí si escribimos **set type=tipo** y en tipo ponemos el tipo de registro que queremos saber podemos averiguar los distintos registros para un dominio. Por ejemplo, para saber la direccion IPv6 del dominio de google:

**C:\> nslookup <Enter>**

**>set type=AAAA**

**>www.gmail.com**

CLASE 15 DE OCTUBRE

**CAPA DE APLICACIÓN:**

**FTP (File Transfer Protocol – Protocolo de Transferencia de Archivos)**

Es un protocolo que se usa para transferir archivos en ambos sentidos, es decir que permite subir o descargar archivos a un servidor.

**Características:**

 **Corre sobre TCP:** Esto se debe a que cuando se transfiere un archivo es necesario que el mismo llegue completo al destino.

 **Usuario y Contraseña (Privado):** Al conectarse contra un servidor FTP se solicita un usuario y una contraseña, de manera que solamente se puede acceder a un servidor FTP al que se pertenezca.

 **FTP:**

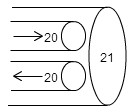
* **Anónimo:** Algunas empresas tienen servidores en donde se puede subir material en general, entonces cuando alguien que no pertenece a esa empresa quiere acceder a ver dicho material se loguea indicando como nombre de usuario “anonymous” y como contraseña se coloca la dirección de correo electrónico de esa persona. Esto implica que el servidor me va a permitir acceder y me va a dejar descargar aquellos archivos de público acceso a los que se tiene permiso de acceder en este estado.
* **Privado:** Si una empresa tiene un servidor FTP, solamente podrán ingresar aquellas personas que estén registradas para hacerlo. Esto se controla a través del usuario y la contraseña.

 **Mantiene el estado:** Cuando se conecta un usuario a un servidor debe loguearse, y luego el servidor está atento a que hace el usuario durante toda su sesión hasta que se desconecta, lo que se conoce como mantener el estado. Esto implica que el servidor va a reservar muchos recursos mientras el usuario este logueado en ese servidor.

 **Conexiones:** Tiene dos tipos de conexiones:

* **Persistentes (Control):** También conocida como conexión de control. Cuando el usuario se loguea, la conexión queda abierta todo el tiempo hasta que decida cerrar sesión. El servidor escucha en el puerto 21. Si el usuario quiere pedirle algo servidor ejecuta un comando, que va a viajar por la conexión persistente. Es decir que la conexión persistente significa que se establece una conexión TCP entre el cliente y el servidor, y todo lo que el usuario tipea le va a llegar al servidor.
* **No persistente (Datos):** Cuando el usuario le solicita un archivo al servidor, el servidor para enviarlo abre otra conexión TCP que es de datos o no persistente, envía el archivo y libera la conexión (por eso recibe el nombre de no persistente). Aunque se cierre esta conexión, siempre se mantiene abierta la conexión persistente. Si se requiere descargar otro archivo de nuevo se vuelve a abrir una conexión no persistente, es decir que esta conexión únicamente se utiliza para enviar datos.

Gráficamente: se abre una conexión grande por el puerto 21, y dentro de esta conexión se van abriendo y cerrando conexiones más chicas en el puerto 20 en donde se hacen las transferencias de archivos en cualquier sentido (se pueden subir y bajar archivos).



**Operaciones (Comandos Básicos):**

**User:** Se especifica el nombre de usuario con el que se quiere conectar.



**Pass:** Contraseña asociada al usuario que quiere loguearse.

**List:** muestra todos los archivos que se encuentran en el directorio actual en el que esta posicionado el usuario.

**Retr:** Se utiliza para descargar un archivo.



**Stor:** Se utiliza para cargar un archivo.

**?:** También conocido como Help, nombra y describe todos los comandos que se pueden utilizar. **Quit:** Para cerrar sesión.

**Cwd:** Permite cambiar de directorio.

**TFTP (Trivial File Transfer Protocol – Protocolo de Transferencia de Archivos Liviano) Características:** Permite transferir archivos (subir o descargar) de manera muy rápida.

**Corre sobre UDP**



**No autentica:** No solicita ingresar a través de usuario y contraseña, lo que no lo hace seguro ya que no se puede determinar quien accede y quién no.

 **No mantiene estados de las conexiones:** El servidor no está atento a lo que hace un usuario cuando ingresa. Por ello se conoce como ua protocolo de transferencia de archivos trivial, liviano. Decir que es liviano hace referencia a que la cabecera es simple y que el protocolo no controla muchas variables.

 **Segmentos de 512 bytes:** Cuando se realiza una transferencia el protocolo garantiza que los archivos lleguen completos y ordenados. Para ello, desde la aplicación el propio TFTP envía bloques de datos de 512 bytes con un temporizador para cada uno de ellos. Supongamos que tenemos que transmitir tres segmentos, lo que hace el protocolo es enviar el primer bloque de datos, que cuando llega al destino éste estará encargado de enviar un acuse de recibo (ACK). Cuando el origen recibe el acuse de recibo envía el segundo paquete, y así sucesivamente. En caso de que el origen no reciba el acuse de recibo luego del tiempo determinado por el temporizador especificado al lanzar el segmento, lo que hace es retransmitir ese bloque de datos, y de esta manera logra garantizar que los archivos lleguen completos y sus segmentos ordenados.

 **Se usa para actualizar software:** Estos protocolos se usa adentro de las empresas para realizar actualizaciones de Software. Supongamos que tenemos 50 puestos de trabajo y tenemos que instalar un sistema operativo en ellos, no tiene sentido ir uno por uno realizando las instalaciones, sino que lo

que se hace es instalarlo solo una vez en un servidor TFTP y de ahí se transfiere a todos los puestos de trabajo.

Respaldar la configuración de un router en un servidor TFTP: Otra funcionalidad está relacionada con los routers: se sabe que estos dispositivos guardan su configuración en una memoria NVRAM que no se borra, pero si alguien llegara a borrar ese archivo de configuración se perdería la misma. Por lo tanto, para evitar dicha situación, lo que se hace es guardar un respaldo del archivo de configuración en un servidor TFTP. De esta manera, el administrador de la red puede tener toda la configuración de los routers centralizada en un servidor (también se pueden resguardar los sistemas operativos de los routers). Si ejecutamos los comandos:

**#copy running-config tftp** decimos que copie la configuración actual en un servidor TFTP.

**Ip-servidor** El sistema solicita se ingrese la IP del servidor

**Nombre\_del\_archivo:** El sistema solicita se ingrese el nombre que queremos que tenga el archivo de configuración en el servidor TFTP.

**!!!!** Luego aparecen varios signos de pregunta donde cada uno de ellos representa a un segmento de 512 bytes que fue enviado.

**#**

**GESTION DE RED:**

**Gestión:** administrar, controlar, analizar, informar recursos de Hardware y Software para que la red funcione según las políticas definidas por la empresa, a un cierto costo. **Áreas de gestión de red:** Fueron definidas por la ISO

1. **Gestión de rendimiento:** Las empresas definen políticas con respecto a la red (por ejemplo que los usuarios tengan un ancho de banda determinado) que luego el administrador debe controlar que se cumplan, y de ser posible que se superen. El administrador puede medir el rendimiento de una red haciendo un análisis del tráfico y del ancho de banda, para tratar de mantener la performance esperada de la red. Se puede controlar a qué velocidad viajan los datos, si hay mucho tráfico dirigido hacia un servidor y se puede congestionar, los tiempos que tardan en viajar los mensajes, etc. Las actividades antes mencionadas deben hacer con una frecuencia semanal o mensual y debe almacenarse en registros para poder analizarlos posteriormente. Es bueno contar con un software que apoye la gestión del rendimiento de la red.

1. **Gestión de fallas:** Un buen administrador tiene que anticiparse a las fallas, no debería actuar solamente cuando le indican que hay algún problema. El administrador cuenta con herramientas para saber si una red va a fallar, si se va a caer una interfaz, si un enlace se está por saturar, etc. Por ejemplo, si se ve que la interfaz de un router está continuamente conectando y desconectándose es necesario anticiparse y actuar rápidamente antes de que definitivamente se caiga. Para el último caso mencionado, se puede ejecutar el comando ifconfig/show interfaces y obtener información sobre las interfaces como por ejemplo si están conectadas o desconectadas, la cantidad de paquetes transmitidos, la cantidad de paquetes recibidos, cantidad de errores, etc. De manera que podemos determinar cuándo una interfaz no funciona correctamente.

1. **Gestión de configuración:** Consiste en controlar constantemente la configuración de los dispositivos de una red. La configuración no significa solamente configurar todo una vez, sino que constantemente es necesario estar controlando las configuraciones porque no siempre funciona todo correctamente (los caminos elegidos pueden ser los más largos por ejemplo). No es necesario estar cerca de los routers para hacer estas tareas, es posible hacer la configuración a distancia.

1. **Gestión de usuarios:** Consiste en gestionar los perfiles de usuarios, esto implica definir los privilegios, permisos, los recursos a los que puede acceder, etc.

1. **Gestión de seguridad:** Implica controlar la seguridad dentro de la empresa, por ejemplo a nivel de antivirus por si los empleados conectan dispositivos de almacenamiento con virus, tener sistemas operativos actualizados, configurar firewalls (controla que paquetes salen y que paquetes entran desde y hacia la empresa), etc. También se puede implementar un IDS, que consiste en un sistema de detección de intrusos, de esta manera el administrador puede darse cuenta si hay ataques a la red o si hay intrusos que quieren penetrar en la misma. Además, es necesario controlar la seguridad en los bordes de la empresa.

**Protocolos de gestión de red:** Existen dos protocolos de gestión de red:

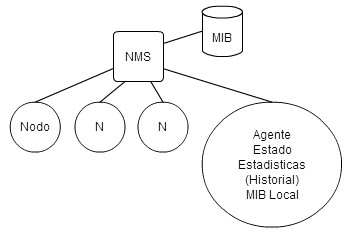
 **ISO:** El protocolo definido por la ISO se llama **CMIP (Common Management Information Protocol – Protocolo de Administración de Información Común)**.

 **IETF:** El protocolo definido por la IETF se llama **SNMP (Simple Network Management Protocol – Protocolo Simple de Gestión de Red)**. Al comprar un dispositivo tenemos que estar seguros que como mínimo soporte este protocolo, a estos se los conoce como dispositivos administrables o gestionables

**Modelo de gestión de red:**

Cuando nació Internet no existía la gestión de red debido a que existían muy pocos dispositivos y con el usp de tan solo algunos comandos, como el ping, tracerout, o netstat era suficiente para gestionar la red. Pero una vez que Internet comenzó a crecer cada vez eran más los dispositivos que gestionar y comenzó a ser inmanejable. Así es como surge el modelo de gestión de red. En la actualidad existe software que permiten gestionar redes completas, en donde se muestran en distintas pantallas con distintos colores cual es el estado de la red para saber si algún router esta congestionado, si se cayó alguna interfaz, etc. Estos software pueden ser propietarios (solo gestionan dispositivos de una determinada marca) o heterogéneos (gestionan dispositivos de muchas marcas). Se gestiona tanto el Hardware como el Software.

Los componentes que forman parte del modelo son:

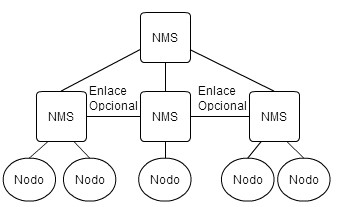


1. **Dispositivos gestionados (nodos):** Son los dispositivos que se van a gestionar. Están formados por un software agente y una MIB local.

1. **Estaciones administradoras de red (NMS – Network Management Station):** Son las computadoras que van a tener instalado el software de gestión de red. Para enterarse cuál es el estado de los nodos a gestionar, las NMS realizan periódicamente un sondeo que consiste en enviar preguntas sobre ciertos parámetros hacia todos los nodos de la red. Los dispositivos envían la respuesta a las NMS con los datos requeridos, a partir de lo cual se crea un mapa o grafo de la situación relevada.

Otra forma para conocer el estado de la red consiste en que los dispositivos gestionados le envíen a las NMS un mensaje de notificación cuando perciban algún problema, lo que se conoce como tráfico. Por ejemplo, si a un router se le cae una interfaz, dicho dispositivo deberá enviar un mensaje notificando tal situación a la NMS.

Existen diferentes arquitecturas de las estaciones administradoras en función de la envergadura de las empresas. Para una empresa chica con una sola estación administradora es suficiente, mientras que para una empresa grande con muchas sucursales se utiliza una arquitectura distribuida de NMS. De esta manera se evita centralizar toda la información en un solo NMS lo que supondría mucho tráfico de datos. Entonces, la arquitectura distribuida de las estaciones administradoras busca tener una NMS por cada sucursal de manera que se encuentren más próximas a los dispositivos que se quieren gestionar (es más rápido porque los enlaces son más cortos). La arquitectura es distribuida pero también jerárquica, ya que cada estación administradora releva todas las estadísticas de sus dispositivos y cada cierto tiempo se lo informa a una NMS central. Otra alternativa es que exista un único nivel de NMS y entre ellas se intercambien información para tener conocimiento de lo que pasa en otras sucursales. Se pueden configurar valores umbrales, que especifican que una variable puede crecer hasta alcanzar el valor umbral, si lo supera es necesario notificar a las estaciones administradoras dicha situación para que actúen en consecuencia.



1. **Agente (software):** Los dispositivos gestionados para poder recibir y responder a las peticiones, o notificar cuando tienen algún problema, deben tener instalado un agente que es un software de pequeño tamaño (liviano) que está almacenado en el propio dispositivo que se está gestionando. Es decir que los agentes van a estar en los nodos, y se van a encargar de informar sobre el estado de los mismos, así como mantener estadísticas o historial. Esto último hace referencia a que el dispositivo tiene la obligación de ir guardando información de un periodo de tiempo corto, por ejemplo de los últimos 5 minutos, debido a su corta memoria; y los NMS tienen la obligación de consultarlos periódicamente.

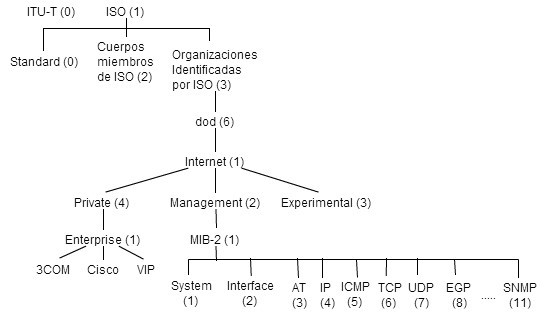
1. **MIB (Management Information Base – Base de Información Gestionada):** Cada dispositivo tienen distintas variables que gestionar, por lo que reciben distintos tipos de preguntas por parte de los NMS en función de las características que presente cada nodo. Para saber qué tipo de pregunta debe realizar un NMS a los objetos gestionados se utiliza una MIB, que es una base de datos que contiene todas las variables que se pueden gestionar de todos los objetos (esta BD es de gran tamaño). Cada dispositivo tiene su propia MIB interna o local que consiste en una pequeña base de datos en donde se definen las variables que se le van a poder gestionar al dispositivo que almacena dicha MIB.

1. **Protocolo SNMP**

**SMI (Structure Management Information – Estructura de la Informacion Gestionable):**

La MIB está definida a través de una SMI que es la estructura de la información que gestiona. Esto especifica que es necesario conocer el nombre de lo que se va a gestionar, el tipo de dato que es (entero, coma flotante, string, etc.), y el valor que puede adquirir.

**Jerarquía de la MIB-II:**



En el primer nivel se pueden apreciar las grandes organizaciones de estandarización (ITU-T e ISO). A cada una se le va asignando un número, y la numeración es por nivel. En el segundo nivel se definen todas las organizaciones que están debajo de la ISO, y la numeración comienza de nuevo desde cero. En el tercer nivel se encuentra el departamento de defensa de los Estados Unidos, mientras que en el cuarto esta internet. Debajo de internet esta todo lo referido a gestión, experimental o empresas privadas. Dentro de las empresas privadas esta Enterprise y debajo de este todas las empresas privadas. En la parte de gestión, debajo de esta más específicamente, tenemos la MIB-II en donde se listan todos los objetos que podemos gestionar.

* Objetos system: se puede gestionar la versión de los sistemas operativos de los objetos.
* Interface: me dice cuantas interfaces tiene activas.
* AT (traducción de direcciones): me dice cuantas traducciones de direcciones implemento el dispositivo (cuantas veces leyó sus tablas).
* IP: Se pueden pedir muchos parámetros con respecto a IP, por ejemplo cuantos datagrama con TTL=0 envió en los últimos 10 minutos.
* ICMP: cuantos paquetes del tipo Dato se enviaron.
* TCP: se pueden preguntar cosas relacionadas con las cabeceras.
* SNMP: También se pueden preguntar cosas referidas al mismo SMNP, por ejemplo cuantas peticiones de variables recibió.

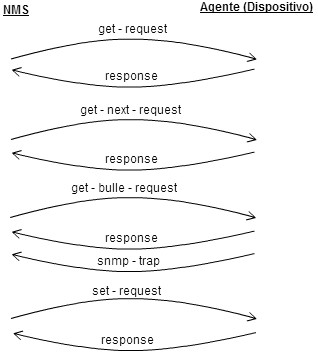
Para pedirle información a un dispositivo hay que ejecutar un get y al lado especificar que dato queremos de ese dispositivo, lo que lo hacemos ingresando 1.3.6.1.2.1.1.objeto. Cada dispositivo tiene preconfigurado que se le puede consultar, y se hace con números en lugar de nombre para ocupar menos lugar en la base de datos.

**Protocolo SNMP (Simple Network Management Protocol – Protocolo Simple de Gestión de Red):**

Este protocolo se usa para ver cómo se va a establecer la comunicación entre la estación administradora y el dispositivo gestionado. Se definen que tipos de mensajes se van a enviar para poder gestionar el dispositivo.

Esta aplicación corre sobre el protocolo UDP en los puertos 161 y 162. Cuando se le manda a preguntar al agente se utiliza el puerto 161, mientras que cuando el agente avisa que tuvo algún error lo informa al puerto 162.

**Mensajes SNMPv3:**



Dijimos que el agente tiene su propia MIB local, por lo tanto tiene un conjunto de objetos que le puede gestionar la estación administradora.

Para hacer un sondeo la NMS debe solicitar el valor de alguna variable y lo hace a través de la operación o mensaje **get-request**. El software agente que recibe esta petición consulta en su MIB interna el valor de la variable y envía un **response** como respuesta. Si la estación administradora quiere conocer el valor de 10 variables deberá enviar 10 mensajes al dispositivo. Pero, para evitar que la NMS este constantemente entrado en su MIB (que es bastante grande) para consultar que variables puede gestionar del agente, lo que hace es enviar el mensaje **get-request** para la primer variable y luego envía **get-next-request** en donde solicita que le devuelvan el valor de la siguiente variable. De esta manera se puede hacer un pedido de muchas variables pero sin la necesidad de especificar que variable, ya que la NMS va a saber de cual se trata porque el agente va a responder en siempre en el mismo orden. Por cada get-next-request hay un response. Para evitar que haya tanto trafico de mensajes la versión 3 incorporo un nuevo mensaje que permite solicitar todas las variables de una sola vez, y que el agente responda con el valor de todas la variables solicitadas en un solo mensaje. Esto se logra con el mensaje **get-bull-request**, y de esta manera solo se necesita 2 mensajes para obtener todos los datos.

Si un dispositivo supera el umbral que le fue definido tiene la obligación de informarle al NMS, y esto se hace con el mensaje **snmp-track**. Entonces cuando el agente tiene un inconveniente tiene la obligación de enviar este mensaje.

Como este protocolo corre sobre UDP es posible que algún mensaje se pierda, porque no hay retransmisión. Por lo tanto, para superar este inconveniente, la estación administradora al enviar un mensaje le asigna un temporizador. En caso de que se venza el tiempo de dicho temporizador y no obtuvo respuesta, la NMS retransmite el mensaje. Como vemos, el control lo ejerce la estación administradora y no el protocolo UDP.

Dijimos previamente que las NMS permiten hacer configuración a distancia, por lo que pueden cambiar el valor de una variable de un dispositivo, esto se logra mandando el mensaje **set-request** (debe indicarse que variable se va a cambiar y que valor tendrá). Esto requiere que el dispositivo envíe un **response** para que la estación administradora sepa que se pudo realizar la modificación.

En las arquitecturas distribuidas y jerárquicas del modelo de gestión de red, una vez que la estación administradora recabo toda la información de todos los dispositivos, tiene que enviar esa información a la NMS central. Este mensaje se intercambia entre estaciones administradoras para enviarse los reportes.



# CLASE 22 DE OCTUBRE

**CORREO ELECTRONICO:**

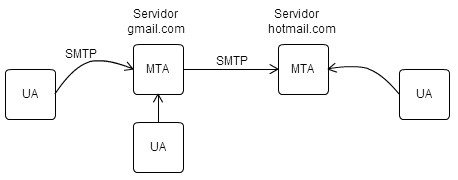
En un comienzo se escribían los mails en un editor de texto, se utilizaba código ASCII y no se podían mandar archivos adjuntos.

**Arquitectura:**

**Componentes:**

**UA (User Agent - Agente de Usuario):** El agente de usuario se ejecuta en una maquina y me permite escribir y leer correos.

**MTA (Main Transfer Agent – Agente de Transferencia de Correo):** Generalmente son los servidores de correo.



Al escribir un correo y enviarlo es necesario que el mismo vaya a un agente de transferencia de correo, que normalmente es el servidor de correo. Si el mail va dirigido a un usuario que tiene el mismo servidor de correo que la persona que lo envía, el mismo servidor de correo se encarga de almacenarlo en el buzón del usuario (que es un trozo de disco del servidor). El mensaje es guardado en el buzón hasta que el usuario tenga ganas de leerlo. Para poder hacerlo debe autenticarse ante el servidor de correo a través de un usuario y contraseña (para demostrar que es quien dice ser), y una vez esto hecho se muestran todos los correos que están almacenados en el buzón.

En caso de que la persona que mande el mail no pertenezca al mismo servidor de correo que el receptor, ya que por ejemplo uno usa gmail y el otro yahoo, va a ser necesario que exista una comunicación entre los dos MTA de los usuarios. Entonces, el MTA del emisor le manda el correo al MTA del receptor que tiene un demonio que cada cierto tiempo se fija si le va a llegar algo, cuando detecta esta situación se establece la conexión y recibe el correo que es almacenado en el buzón del usuario. La comunicación entre MTAs se realiza a través del protocolo SMTP, que es un protocolo de subida, de entrega de correo (no de descarga). El protocolo SMTP corre sobre TCP en el puerto 25. Hay veces que hay varios saltos o comunicaciones entre MTA hasta que el mail llega al receptor.

El correo electrónico es asincrónico, esto significa que cuando uno escribe un mail no es necesario que la otra persona esté esperando a ver si se entrego el mismo.

**Servicios básicos:**

**Composición:** Tiene que haber un software que permita escribir el mail.



**Transferencia:** El mensaje sale del agente de usuario y es entregado al servidor para que el mismo llegue al buzón del destinatario.

 **Generación del informe:** Debe haber un Software que diga si el mensaje fue entregado, si fue rebotado, si no existe el destinatario, etc.

 **Visualización:** tiene que haber un Software que permita visualizar todos los mails que tengo en el buzón.

 **Disposición:** Se debe permitir decidir al usuario que va a hacer con el mail, si borrar, guardar, reenviar, responder, etc.

**Servicios avanzados:**

Si me voy de viaje puedo redireccionar el mail a otra cuenta o configurar una respuesta automática.



Listas de correo: antes si se quería mandar el mail a 5 personas había que enviarle el mismo correo a las 5 personas, ahora puedo escribir un grupo y el agente se encarga de replicarlo muchas veces.

Seguridad: puede cifrarse el mensaje de correo.



Puedo enterarme si la otra persona abrió o leyó el correo.

Un mensaje de correo electrónico consta de dos partes: Cuerpo (o datos) y la cabecera.

**Campos de la cabecera:**

 **To:** Hace referencia al/los destinatarios primarios a los que se quiere que le llegue el correo. Los destinatarios primarios ven a los destinatarios secundarios.

 **Cc:** Hace referencia a que el mail va a ir con copia a los destinatarios secundarios. Los destinatarios secundarios ven a los destinatarios primarios.

 **Cco:** Con copia oculta, significa que el mail va a llegar a un destinatario oculto, que no va a poder ser visto por el destinatario primario (ni los secundarios de CC).

**Date:** Fecha en la cual se envía el mensaje.



**Asunto (Subject):** pequeño resumen de lo que contiene el mail.

**Return-Path:** Indica a quien tiene que devolverse el correo electrónico. En caso de que haya algún problema indica a quien hay que notificarle dicha situación. Normalmente es el remitente.

 **Received:** Los mensaje a veces atraviesan varios agentes de transferencia (servidores de correos). Cada vez que un MTA recibe un mensaje le estampa una línea que dice Receive en donde especifica su nombre, su IP, la fecha y la hora. Si un mensaje atraviesa por ejemplo 3 MTA, quedaran registradas en el mensaje 3 líneas de Receive. Se usa para detectar fallas (porque un mensaje no se encamino correctamente).

 **Messaje-id:** Es un identificador único de mensaje, son números y/o letras que se antepone a la dirección de correo.

 **Priority:** Los mensaje generalmente tienen prioridad normal, pero si llevan prioridad alta se van a entregar más rápido.

**From:** Quien escribió el mensaje



**Sender:** Quien envió el mensaje. No siempre la persona que escribe el mensaje es la persona que lo envía.

 **Reply to:** Indica que la respuesta al correo enviado vaya a la dirección que está indicada en este campo.

Los agentes de transferencia de correo son quienes administran estos campos para que tratar de que la entrega del mensaje llegue correctamente al destinatario.

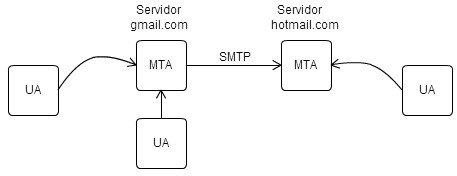
**PROTOCOLO SMTP (Simple Mail Transfer Protocol – Protocolo para la Transferencia Simple de Correo)**

La función de este protocolo es que el agente de usuario que envíe el correo al servidor de correo electrónico, es un protocolo de entrega (no es para descargar correos). Solamente se establece la conexión cuando hay que transferir un correo de un punto a otro, y se establece entre el agente de usuario y el MTA, y entre los servidores de correo. El primer de los casos ocurre cuando la comunicación se da entre usuarios que pertenecen a un mismo servidor de correo (por ejemplo cuando los dos mails son de gmail) ya que el mail va a permanecer almacenado en el mismo servidor. El segundo caso ocurre cuando la comunicación se da entre usuarios que pertenecen a distintos servidores de correos (por ejemplo cuando un mail es de gmail y el otro de yahoo).

Este protocolo corre sobre el protocolo de transporte TCP porque hay que garantizar que todo el mensaje llegue completo al destinatario, y utiliza el puerto 25 (es decir que hay un servidor de correo que va a estar escuchando en el puerto 25).

El protocolo tiene una arquitectura cliente-servidor, en este caso el agente de usuario va a funcionar como cliente y va a establecer una conexión TCP con el servidor MTA y le va a transferir el correo. En general los servidores funcionan con demonios, por eso los correos no se entregan al instante sino que a veces demoran un tiempo en llegar. Esto se debe porque cuando los demonios intentan establecer una conexión con los servidores y estos justo están ocupados, vuelven a intentar de nuevo mas tarde. Si intentan varias veces y no pueden establecer la conexión, es ahí donde envían una notificación de que el correo no pudo ser entregado.

**Funcionamiento:**



Al poner una dirección de correo dice el nombre@dominio. El agente de usuario necesita obtener la dirección IP del servidor de correo para establecer un socket de conexión, pero antes necesita conocer cuál es el dominio de dicho servidor para poder realizar una traducción a través del protocolo DNS. Por lo tanto primero averigua cual es el registro MX, ya que es el que nos va poder brindar el nombre del servidor de correo, y una vez obtenido procede a averiguar la dirección IP del servidor de correo haciendo una traducción DNS del tipo dominio/IP. Recién ahora se establece la conexión TCP y el agente de usuario transfiere todos los correos que tiene. En ocasiones, el MTA espera a tener varios correos para enviar al servidor de yahoo o Hotmail y transfiere todos los mails en una misma conexión TCP. Por este motivo cuando nos comunicamos con un usuario de otro servidor no siempre llega instantáneamente el correo, en cambio cuando la comunicación se da con un usuario que pertenece al mismo MTA el mail llega instantáneamente porque el servidor simplemente lo tiene que guardar en el buzón del usuario receptor. Una vez que se transfieren todos los mails se cierra la conexión TCP y el servidor de correo le almacena en su disco rígido.

**Descarga de correo:**

Como la comunicación de correo es asíncrona, el agente de usuario del receptor no tiene que estar esperando a ver si le escribieron. El AU cuando quiera leerlo se va a conectar con su servidor de correo, quien va a pedir una autenticación del usuario debido a que gestiona los correos de muchos usuarios y solamente debe mostrar los mails pertenecientes a la persona que intenta ingresar. Una vez hecho esto permite ver un resumen de los correos que tiene el usuario, lo que se conoce como visualización.

Para ver (descargar) los correos electrónicos almacenados en el buzón de usuario es necesario utilizar los protocolos IMAP o POP3. Estos protocolos no son iguales, primero surgió el POP3 y luego surgió IMAP **POP3 (Post Office Protocol – Protocolo de Oficina de Correo):**

Este protocolo me permite hacer una descarga de correo en el agente de usuario y almacenar ese correo en el disco rígido. Es decir que una persona podía conectarse al servidor de correo, descargar todos sus mails y almacenarlos en el disco rígido. Una vez almacenados se desconecta del servidor de correo. Esto se debe a que se utilizaban conexiones de Dial Up y era caro estar conectado, ya que se cobraba en función del tiempo que uno pasaba conectado. La ventaja que presentaba es que una vez que ya había descargado todos los mails podía liberar la conexión a internet y podía usar la línea telefónica para hablar por teléfono. Además, los mails podían leerse todas las veces que se deseen porque ya estaban en el disco rígido de la máquina.

Ventajas:

 Los discos rígidos de los servidores de correo no se llenaban tanto como en la actualidad (que requieren mucho tamaño). Cabe recalcar que en esa época no había mucha capacidad de almacenamiento y los usuarios tenían poco espacio para guardar sus correos.

Desventaja:

 Siempre había que consultar el correo desde una misma Pc. Si te ibas de viaje no era posible ver los correos a menos que lleves la computadora.

**IMAPv4 (Internet Message Access Protocol – Protocolo de Acceso a Mensajes de Internet)**

Este protocolo es mucho más completo que el POP3 porque permite conectarse al servidor de correo y ver los mensajes que están en el servidor sin necesidad de descargarlos. Es decir que los mails van a estar siempre almacenados en el disco rígido del servidor. Además, permite crear carpetas para organizar los correos, realizar búsquedas, etc. La ventaja que tiene es que permite movilidad a los usuarios, es decir que pueden ver sus correos desde cualquier computadora y no solamente tienen que usar una sola. La desventaja que tiene es que para poder consultar los correos es necesario estar todo el tiempo conectado a internet (conexión permanente).

Otra forma de consultar los correos es a través de un Correo Web, que permite visualizar los mails por medio de un navegador (no es necesario tener una aplicación de correo en la maquina).

**MIME (Mail Internet Multipurpose Extensions – Extensiones Multipropósito de Correo de Internet)**

Originalmente en los correos únicamente se transmitía textos en ASCII, así es como surge el protocolo SMTP. Como internet fue evolucionando y comenzó la necesidad de transmitir archivos de multimedia se creó el protocolo MIME. Para no tener que cambiar toda la estructura manejada por el protocolo SMTP (agentes de usuario y MTA) se decidió que MIME iba a ser encapsulado por el SMTP. De esta manera el SMTP puede llevar y traer correos sin darse cuenta que hay adentro. Por lo tanto el que va a manejar la complejidad de la información no va a ser el servicio de correo (protocolo SMTP) sino que van a ser los extremos (agentes de usuario) quienes deben tener la inteligencia para poder interpretar los mensajes.

El protocolo MIME me permite enviar cualquier tipo de datos (letras negritas, subrayadas, imágenes, videos, etc.) a través de correo electrónico, ya sea como archivo adjunto o formando parte del mismo mail. Para ello es necesario agregar una cabecera más, que será encapsulada en un mensaje SMTP, luego encapsulado en TCP, luego en IP y por ultimo en una TRAMA. De esta manera los agentes de transferencia de correo para encaminar el mail tienen que ver la cabecera SMTP, no tienen por qué meterse en la cabecera MIME.

**Campos de la cabecera:**

 **Mime version:** Se establece una campo versión para que el protocolo pueda evolucionar. En el caso en el que se mande un texto ASCII únicamente, no va a ser necesaria una cabecera MIME, por lo que en este campo no habría nada.

 **Content – description:** Tiene información sobre el contenido del mensaje. Es lo que se ve cuando viene un archivo adjunto. Nos permite elegir si queremos abrir o no el archivo adjunto en función de esa descripción.

 **Content – id:** Es una identificación única de la cabecera MIME.

 **Content – transfer – encoding:** Es un tipo de codificación. Cuando se transfiere audio, video, o imagen, la información va codificada. Esta información se va trasladando por los servidores de correos quienes no tienen noción de lo que transmiten, pero cuando llega al agente de usuario receptor, para poder visualizar lo que se está mandando, tiene que fijarse de que manera vino codificada para poder decodificarla y mostrarla al usuario.

 **Content – type:** Indica que tipo de contenido tiene el mensaje MIME (si es un archivo de Word, una imagen, un video, etc.).

|  |  |
| --- | --- |
| **Tipo** | **subtipo** |
| ***Texto*** | Plano, Enriquecido (con negrita, subrayado, etc.), html, xtml |
| ***Imagen*** | Gif, jpeg |
| ***Audio*** | Basic, mp3 |
| ***Video*** | Mpeg |
| ***Aplicación*** | Word, pdf |
| ***Mensaje*** | Parcial (si el mensaje es largo se divide en partes), externo (si queremos pasar una peli, en lugar de pasarla ponemos una url de donde descargarla) |
| ***Multiparte*** | Mezclado (Mixed, cuando hay muchos tipos en un mismo correo), alternativo (el agente de usuario puede ver el mensaje de distintas aneras, elijo uno o el otro), paralelo (mando dos tipos de mensaje distintos pero exijo que se visualicen en paralelo, por ejemplo video y música) |

**Telefonía en internet (VOIP – Voice Over IP – Voz Sobre IP):**

La red telefónica requería que haya un doble cableado: uno para transmitir el audio y otro para transmitir los datos. Con voz sobre IP esto no sería necesario ya que sobre un solo utp se puede transmitir tanto voz como datos. Esto es posible porque se crearon estándares que permiten digitalizar la señal, es decir que las señales analógicas como la de la voz son muestreadas, cuantificadas y codificadas y pasadas a unos y ceros. Una vez digitalizada, la voz es empaquetada y viaja por la red de datos; cuando llega al destino se reciben los bits, se decodifica el mensaje y se transforma nuevamente en una señal analógica para ser escuchada. De esta manera las empresas ahorran mucho dinero porque no deben gastar en un doble cableado, y los usuarios también porque solo deben pagar por el plan de datos y pueden hablar todo lo que quieran con cualquier persona que se encuentre en cualquier parte del mundo.

La desventaja que tiene es que la red de IP no fue pensada para transmitir voz, y por lo tanto la calidad de la llamada puede verse afectada ya que los paquetes pueden sufrir retardos ya que pueden viajar por distintos caminos. Internet no fue pensada para transferencia de datos en tiempo real.

Para mejorar la calidad se buscan métodos alternativos como por ejemplo el almacenamiento en buffers. Por ejemplo si queremos ver una película lo que se hace es retrasar el comienzo de la reproducción para intentar tener una cierta cantidad de paquetes en el buffer. De esta manera se supera el problema de la fluctuación, que afecta mucho a la voz sobre IP.

A pesar del problema de la calidad se está implementando mucho debido a que se abaratan los costos.

Una forma de utilizar este medio de comunicación es comprando los teléfonos sobre. Pero existe una alternativa que es utilizar la computadora como medio de comunicación, lo que se conoce como SoftPhone, y consiste en utilizar un software que simula ser un teléfono IP.

**ITU:** Definieron una pila de protocolos llamado H323.

**IETF:** Definieron para la transferencia de voz un protocolo llamado SIP (Protocolo de Inicio de Sesión).

**RTP (Real-Time Transport Protocol – Protocolo de Transporte en Tiempo Real):**

Este protocolo está definido por la IETF y permite transmitir datos digitalizados, los cuales pueden ser voz o video. El protocolo no se encarga de la codificación ni de la digitalización de los datos, para eso están los codecs. Entonces, es el códec quien toma la voz o video y la muestrea, cuantifica y codifica, y una vez que esta digitalizada se la pasa al RTP.

Este protocolo corre sobre UDP, lo que significa que no se establece control alguno sobre el envío de los paquetes. Por lo tanto, el control quedara en manos de la aplicación que en este caso es el protocolo RTP.

Para ello, lo que va a hacer el protocolo es agregarle una cabecera a los datos codificados.

**Cabecera:** La cabecera presenta los siguientes campos:

Cada trozo de voz digitalizada tiene 160 bytes, son pequeños porque se muestrea cada 20 milisegundos, y la idea es que mejor que lleguen muchos paquetes chicos que pocos paquetes de gran tamaño.

 **Numero de secuencia:** Por si los paquetes se desordenan se les coloca un numero de secuencia. Se enumera cada paquete, es decir que cada vez que sale un paquete de 160 bytes se le coloca el número de secuencia. Esto funciona así ya que si los paquetes van llegando en orden al destino, éste lo que hace es mantenerlos en orden para mostrarlos, mientras que si hay algún paquete que llega desfasado en el tiempo es eliminado. En el caso de que todavía estén en el buffer los paquetes, el que llego tarde puede ser reacomodado, pero si ya fueron entregados esto no es posible porque se perdería el contexto de la comunicación.

 **Marca de tiempo (Time Stamp):** Sirve para que se sincronice el origen y el destino. Supongamos que en el origen los paquetes salen cada un milisegundo, pero en el destino los paquetes llegan cada 1, 2, 4, 5, 3, etc. Milisegundos produciendo variabilidad de retardo. En este caso el destino le avisa al origen que baje su tasa de transferencia porque los paquetes están llegando con mucho retardo.

 **Tipo de codificación:** Se utiliza para notificar como esta codificada la voz, que tipo de códec uso.

El RTP se encapsula en un segmento UDP, que se encapsula en un paquete IP, que se encapsula en una trama. Además, el protocolo para poder establecer una conexión abre un socket en un sentido y otro socket en otro sentido (porque cuando hablamos por teléfono la comunicación es fullduplex). Pero además, agrega otro canal para establecer un protocolo de control, el RTCP La cabecera es liviana, tiene un tamaño de 12 bytes.

**RTCP (Real-Time Transport Control Protocol – Protocolo de Control de Transporte en Tiempo Real):**

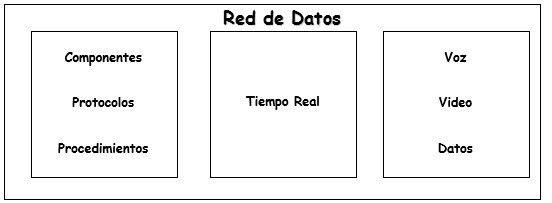
Este protocolo se usa para sacar estadísticas de la transmisión. Entonces, por un canal utilizo el RTP para transmitir la voz, y por otro canal se envía información de control para ver cuántos paquetes se descartaron, cuantos llegaron correctamente, cuantos llegaron con retardo, etc.

# CLASE 29 DE OCTUBRE

**PROTOCOLO H.323**

El estándar H.323 es el primer intento tecnológico para lograr la transmisión de audio, video y datos en tiempo real en una red datos. Son muchos protocolos y componentes donde cada uno cumple una función específica.

Está divido en dos partes, por un lado están todos los componentes necesarios a nivel de dispositivos y protocolos, y por otro lado los datos que se envían. Dichos datos deben ser muestreados y codificados, H323 define específicamente que codec’s son necesarios para cada situación.



H323 fue definido por la ITU, en contrapartida de SIP que fue definido por la IETF.

Los componentes que se definen son:

 **Terminal:** Puede ser una PC o un dispositivo stand-alone corriendo aplicaciones H323 y multimedia. Se exige que los terminales cumplan con ciertas características, como mínimo deben permitir transmitir voz, por lo que debe ejecutar códecs determinados.

 **Gateway:** Es el dispositivo que conecta dos redes distintas, permite unir los “dos mundos”, VoIP y RTPC (Red Telefónica Pública Conmutada). Esto lo realiza traduciendo protocolos. Recibe una señal de la red y la traduce a un porotocolo que pasa del otro lado. Por ejemplo, H323 trabaja protocolos de señalización como el Q931 o el H225, y del lado de la RTPC se utiliza el sistema de señalización N°7, el Gateway traduce dichas señales para que se puedan comunicar.

 **Gatekeeper:** Es el dispositivo considerado como el cerebro de la red H.323, si tengo una LAN necesito un gatekeeper, que administra todas las llamadas a través del control de acceso y controlando el ancho de banda. Además ejecuta el protocolo RAS (Registrer Admision State), que es el protocolo que administra las admisiones, es decir cuando una computadora/teléfono accede a la red, se registra con el Gatekeeper para que éste la administre.

 **MCU (Unidad de control multipunto):**Es el dispositivo que provee el soporte de conferencias de tres o más terminales H.323.

**Terminales:**

Características:

Debe soportar comunicaciones de audio y opcionalmente de video y datos



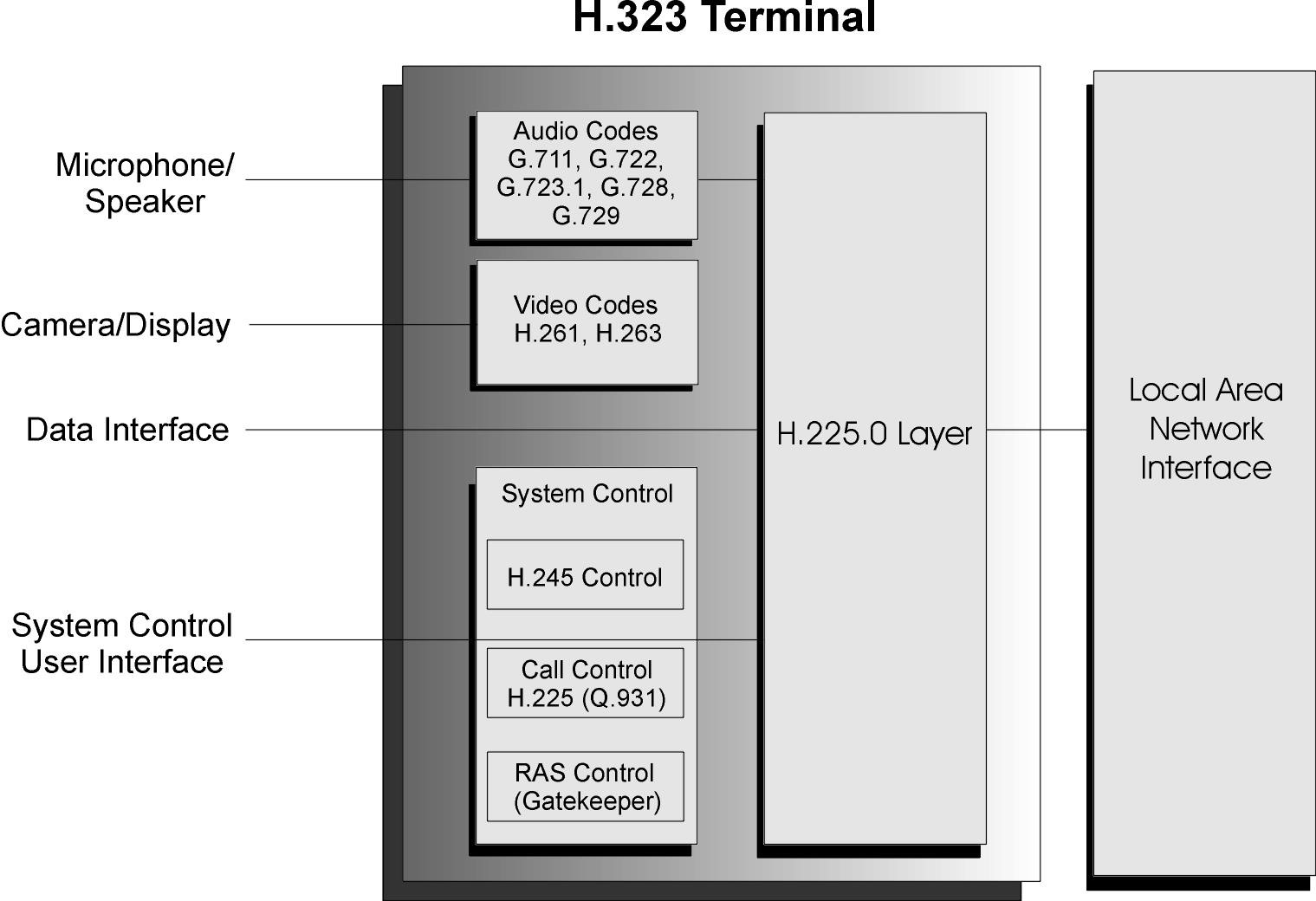
Puede usarse en conferencias multipunto

Se usa en comunicaciones multimedia bidireccionales

Como mínimo debe soporta los siguientes protocolos:

* H.245: Mantiene el control de toda la llamada.
* H.225: Sse utiliza para señalización de la llamada (encaminamiento, sonido de llamada, etc)
* RAS: Protocolo que se utiliza para registrar la terminal en el gatekeeper.  RTP / RTCP: Sirve para transferir y controlar la transferencia de datos.

A continuación se muestra la pila de protocolos:



Puede apreciarse que si bien se incluyen los protocolos de video (Video Codes H.261, H.263), las terminales no están obligadas a soportarlos, es opcional.

**Gateway:**

El Gateway debe proveer traducción de protocolos para el establecimiento y la liberación de la llamada, convertir los formatos entre diferentes redes y transferir la información entre las redes H.323 y las que no lo son.

Las principales Características son:

 Debe soportar los siguientes protocolos: H.245, H.225, RAS y RTP / RTCP sobre el lado de la red H.323

 Debe soportar sobre el lado que no es H.323 los protocolos que se ejecutan en dicha red, ej : SS7 e ISDN

 Los terminales se conectan a él usando H.245 y H.225

**Gatekeeper:**

Como ya dijimos la función del gatekeeper es aceptar peticiones de conexiones, es el cerebro de la red LAN.

Las principales Funciones son:

 **Traducción de Direcciones**: Asocia un Nro de teléfono a una dirección IP, para mantener la comunicación.

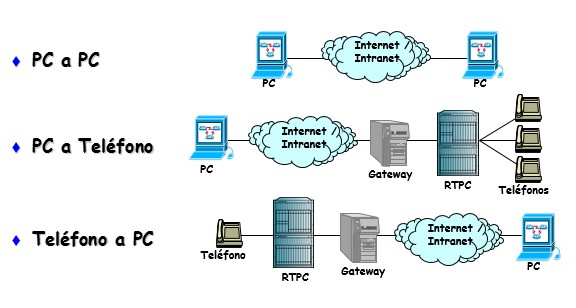
**Control de Admisión.**



**Control de Ancho de Banda.**

**Modalidades de conexión:**

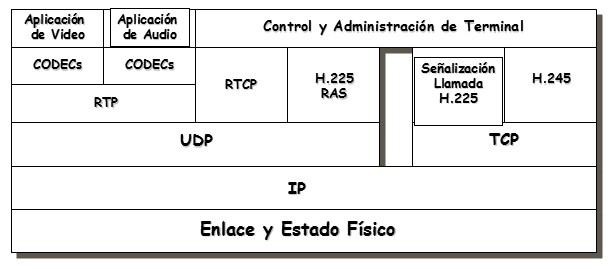
Las distintas modalidades de conexión que se pueden presentar, pueden ser:



Donde se puede apreciar que solo en la comunicación con la RTPC es necesario un Gateway.

**Arquitectura de protocolos**

A continuación se detalla la pila de protocolos:



Codificación de datos

Codificación de Control

No hay solo un socket abierto en una conexión, sino que existen varios sockets abiertos, en donde cada uno puede servir para:

* Transmitir voz
* Recibir voz
* Canal de control
* Etc.

En la parte de **Codificación de los Datos** se puede apreciar que debajo de la capa “Aplicación de video” se ejecutarián los codecs como H.261, H263 y debajo de la capa “Aplicación de audio” codecs como G.711, G.722, etc.

Una vez que se codifican los datos (ya sea audio o video) se los pasa al protocolo RTP que se encuentra por debajo, éste recibe la voz digitalizada, la empaqueta, le coloca una cabecera de 12 Bytes y se la envía al protocolo UDP, para seguir su camino hasta salir a la red.

Por otro lado en la parte de **Codificación de Control** están los protocolos de señalización y control. Los protocolos de señalización de llamada H.225 y el protocolo H.245 corren sobre TCP, aunque generalmente se los suele encapsular sobre UDP ya que es más rápido.

Esta metodología de separar la señalización en otro canal distinto al que se utiliza para transmitir los datos se lo conoce como “Señalización Fuera de Banda”.

**Protocolo h.225 (ras)**

La función principal de este protocolo es ayudar a una terminal a descubrir la ubicación del Gatekeeper, para que este lo registre y pueda realizar llamadas. Este protocolo trabaja sobre el puerto 1720.

**PROTOCOLO SIP (Session Initiation Protocol o Protocolo de Inicio de Sesión):**

Este protocolo intenta ser más simple. Es un protocolo de señalización, se usa para establecer llamadas, mantenerlas y finalizarlas, para hacer que el teléfono suene, etc.

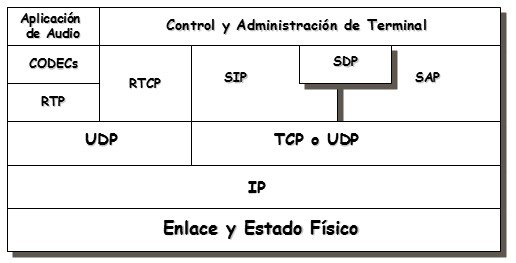
Este protocolo puede correr tanto sobre el protocolo TCP como en el UDP, aunque preferentemente se utiliza el segundo.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | | **SIP** | | | **TCP** | **UDP** | | **IP** | | | **ATM /SDH/WDM** | | |  |

**Port ( TCP ) = 5060Port ( UDP ) = 5060 Protocolo = 6Protocolo = 17**

De esta forma, vamos a tener un canal SIP para cada sentido de la comunicación. Una de las ventajas que tiene es que permite transmitir voz, audio, video, etc. con el propósito de transmitir multimedia en general, además es fuera de banda por lo que se abren otros canales para SIP y para RTCP.

La estructura del SIP es:



Sobre CODEC está definido solo el audio, pero también puede codificar video. Se puede utilizar cualquier CODEC, al SIP no le interesa, lo decidimos nosotros.

En la parte de control, tenemos el protocolo SIP que además se apoya en dos protocolos, el protocolo SDP (Protocolo de descripción de Sesión) que se encarga de la negociación de las capacidades de la comunicación a establecer y el SAP (Protocolo de acceso de sesión) que se encarga de anunciar e invitar a otros terminales a conferencias multimedia, esto corre en TCP o UDP y luego se encapsula en IP.

**Componentes SIP:**

 **Agente de usuario:** Es un software que se va a ejecutar en el cliente (teléfonos IP o Computadoras).Éste software está compuesto por el UAC (Agente de usuario Cliente ) y UAS ( Agente de usuario Servidor), esto se debe a que la terminal debe funcionar como Cliente-Servidor. Este agente va a ejecutar el protocolo SIP, y trabaja con los siguientes mensajes:

* **INVITE –** Establece una sesión, se ejecuta cuando un usuario se quiere comunicar con otro, es una invitación y contiene:
  + Dirección SIP del destinatario (nombre@dominio)
  + Dirección SIP del remitente u origen, para que conozca quien está llamando.
  + IP del origen
  + Método de codificación a utilizar (códec a utilizar)
  + Puerto que se utilizará para datos a través de RTP
* **ACK –** Confirma una solicitud INVITE
* **BYE –** Finaliza una sesión
* **CANCEL –** cancela el establecimiento de una sesión
* **REGISTER –** comunica la localización de un usuario, este mensaje lo envía la maquina/terminal del usuario al servidor de registro para que éste registre una asociación entre la IP del cliente y su nombre SIP.
* **OPTIONS –** comunica información de las capacidades de envío y recepción de teléfonos SIP

 **Servidores de Registro:** La funcionalidad es similar a la del Gatekeeper en H.323. Al encender una máquina si se quiere que tenga posibilidad de realizar llamadas telefónicas, ésta se debe registrar en el Servidor de Registro. Por ende, permiten registrar los usuarios para su posterior localización (método REGISTER). El servidor realiza una asociación (o binding) entre la dirección lógica o dirección SIP o dominio SIP y la IP actual del usuario. Dicha relación se realiza debido a la asignación dinámica de IP (la IP del cliente cambia al desconectarse/reconectarse), entonces cuando un dispositivo SIP necesita comunicarse con otro, solo con conocer su dominio SIP basta con consultar al servidor DNS correspondiente para conocer la IP del mismo, y poder realizar la llamada.

 **Proxy Server:** contacta uno o más clientes o el server próximo y posteriormente pasa los requerimientos de llamada. Encamina el mensaje hacia el destino y queda formando parte del camino. El funcionamiento sería similar a las consultas recursivas en DNS.

 **Redirect Server :** acepta las consultas SIP de los clientes y envía de vuelta una respuesta al mismo la cual contiene la dirección del próximo server y no forma parte del camino. Se compara con las consultas iterativas de DNS.

**SEGURIDAD**

**Políticas de Seguridad:**

Se definen procedimientos que permiten delimitar lo que se puede hacer y lo que no se puede hacer dentro de la empresa, longitudes que deben tener las contraseñas, permisos de acceso que tienen los distintos usuarios a los distintos recursos, en qué momento se puede loguear un usuario, etc.

Todo esto se plasma en un documento técnico, que no es estático, si no que va cambiando para adaptarse a los cambios de tecnología que sufre a empresa.

Una vez que se realizan dichas políticas de seguridad, se debe garantizar que se cumple con las mismas.

Con respecto a los usuarios existen dos métodos para definir los permisos:

 **Política Restrictiva:** Se le niega el acceso al usuario a toda la organización, salvo a lo que se le quiere dar permiso. Esta se prefiere sobre la permisiva.

 **Política Permisiva:** El usuario puede acceder donde quiera, menos a los lugares que no se quiere que tenga acceso.

El encargado de definir las políticas de seguridad de la organización tiene que tener en cuenta los siguientes aspectos:

Aspectos a tener en cuenta en el desarrollo de las políticas:

* Necesidades de la empresa: que es lo que la empresa necesita en función de qué información quiera obtener cualquier persona externa a la misma.
* Identificación de amenazas
* Análisis de riesgos (relación costo-beneficio)
* Necesidades de seguridad
* Prácticas recomendadas por la industria
* Operaciones de seguridad

**Requisitos de una Comunicación segura:**

 **Confidencialidad**  la información debe ser accedida solamente por las partes autorizadas

 **Integridad**  los elementos pueden ser modificados sólo por las partes autorizadas

 **Disponibilidad**  los elementos deben estar disponibles para las partes autorizadas

  **Autenticación** verificar la identidad de un usuario o equipo

 **No repudio** no poder negar algo que se hizo, debe haber una forma para validarse.

**Tipos de Amenazas a la seguridad:**

 Interrupción: Por Ej: cuando se interrumpe un servidor mediante DoS ataque a la **disponibilidad**

 Interceptación: es un intruso que lee la información pero no la modifica ataque a la **confidencialidad**

 Modificación: de los datos por un tercero (man-in-the-middle)  ataque a la **integridad**

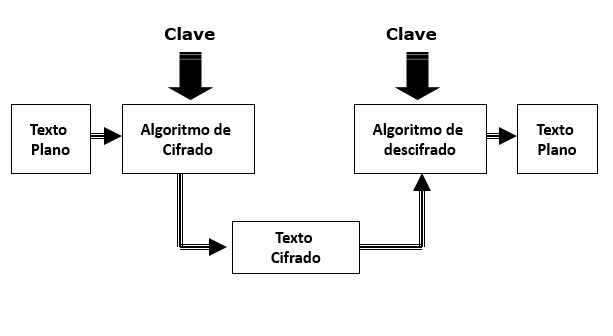
 Invención: Es cuando alguien se hace pasar por un usuario, que no lo es  ataque a la **autenticidad**

**Confidencialidad:**

Concepto: asegurar que la información no se revele a personas no autorizadas y que se protejan los datos transmitidos contra ataques pasivos, como la interceptación.

Solución: cifrar la información, datos, voz, video o su combinación. Esto requiere un algoritmo de encriptación y un esquema de administración de claves.

El proceso de encriptación es el siguiente:



Donde se puede apreciar que el texto plano pasa por un proceso de cifrado y es transmitido, luego en el destino se aplica la inversa del proceso que se aplicó en el origen, y se obtiene el texto plano que se envió. La “fuerza” de un proceso de encriptación no está en el propio algoritmo si no en la clave que se utiliza para cifrar los datos. Por ello los algoritmos son públicos, y cuanto más larga sea la clave, más segura es la encriptación.

Existen distintos tipos de sistemas criptográficos, y pueden ser clasificados según distintos aspectos:

 Según la cantidad de claves utilizadas:

* Simétricos o de clave privada: Emisor y receptor utilizan la misma clave cifrar y para descifrar.

* Asimétricos o de clave pública: Emisor y receptor utilizan diferente clave, una para cifrar y otra para descifrar.

**Sistema Criptográfico de Clave Pública (Asimetrico):**

En los algoritmos asimétricos o de clave pública, se define una clave pública y una privada para una entidad de red determinada (a una persona o a una entidad de red) y a su vez, existe una relación matemática que establece una relación entre ambas claves. La clave pública, como la palabra lo dice, es vista por todo el mundo es decir que si yo quiero conocer la clave pública de alguna entidad, basta con consultarla.

Existen distintas formas de encriptación-desencriptación, según lo que se desea garantizar:

 **Confidencialidad:** Para garantizar confidencialidad, si el emisor A quiere enviar un mensaje a B, lo que realiza A es cifrar el mensaje con la clave PÚBLICA de B, donde dicho mensaje solamente puede ser descifrado con la clave PRIVADA de B. De esta forma todos pueden ver el mensaje pero el único que puede interpretarlo es B (haciendo uso de su clave privada).

𝐴 𝐶𝐼𝐹𝑅𝐴 𝐸𝐿 𝑀𝐸𝑁𝑆𝐴𝐽𝐸 (𝐶𝐿𝐴𝑉𝐸 𝑃𝑈𝐵𝐿𝐼𝐶𝐴 𝐷𝐸 𝐵) + 𝐵 𝐷𝐸𝑆𝐶𝐼𝐹𝑅𝐴 𝐸𝐿 𝑀𝐸𝑁𝑆𝐴𝐽𝐸 (𝐶𝐿𝐴𝑉𝐸 𝑃𝑅𝐼𝑉𝐴𝐷𝐴 𝐷𝐸 𝐵)

* **Autenticidad:** De forma distinta si A desea garantizar a B que el mensaje que éste recibe, realmente proviene de A lo que hace es cifrar el mensaje con su propia clave privada, pero que puede ser descifrado con su clave pública (la clave pública de A), de esta forma B comprueba que el único que puede haber cifrado dicho mensaje es A (haciendo uso de su clave privada). Hay que destacar que un mensaje enviado de esta manera, puede ser descifrado por cualquiera, con lo que no se garantiza la confidencialidad.

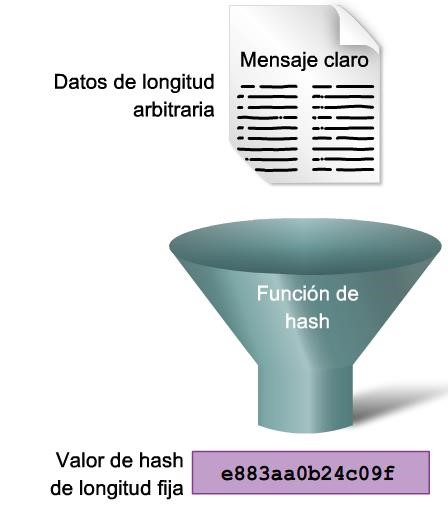
𝐴 𝐶𝐼𝐹𝑅𝐴 𝐸𝐿 𝑀𝐸𝑁𝑆𝐴𝐽𝐸 (𝐶𝐿𝐴𝑉𝐸 𝑃𝐼𝑉𝐴𝐷𝐴 𝐷𝐸 𝐴) + 𝐵 𝐷𝐸𝑆𝐶𝐼𝐹𝑅𝐴 𝐸𝐿 𝑀𝐸𝑁𝑆𝐴𝐽𝐸 (𝐶𝐿𝐴𝑉𝐸 𝑃𝑈𝐵𝐿𝐼𝐶𝐴 𝐷𝐸 𝐴)

 **Confidencialidad + Autenticidad:** Si se quiere garantizar autenticidad y confidencialidad, lo que se realiza conjuntamente los procedimientos anteriores. Es decir, supongamos nuevamente que A envía un mensaje B y necesita autenticidad y confidencialidad, lo primero que hace es cifrar el mensaje con su propia clave privada, y luego al resultado lo vuelve a cifrar con la clave pública de B:

𝐴 𝐶𝐼𝐹𝑅𝐴 𝐸𝐿 𝑀𝐸𝑁𝑆𝐴𝐽𝐸 {[𝐴 𝐶𝐼𝐹𝑅𝐴 𝐸𝐿 𝑀𝐸𝑁𝑆𝐴𝐽𝐸(𝐶𝐿𝐴𝑉𝐸 𝑃𝑅𝐼𝑉𝐴𝐷𝐴 𝐷𝐸 𝐴)]𝐶𝐿𝐴𝑉𝐸 𝑃𝑈𝐵𝐿𝐼𝐶𝐴 𝐷𝐸 𝐵}

De esta forma el mensaje viaja por la red, y solamente B puede descifrarlo haciendo uso de su clave pública, garantizando la confidencialidad. Una vez que B lo descifra, obtiene la clave pública de A, y nuevamente descifra el mensaje obteniendo el original, y garantizando que A fue el que lo envió.

 **Integridad:** Para lograr integridad lo que se hace es calcular un HASH. Un hash es un resumen que se calclua en función de toda la información que se quiere transmitir. Se realiza mediante una función matemática de una sola vía (es decir que se obtiene un valor de hash a partir de un documento, pero no se puede obtener el documento a partir del hash y haciendo la inversa del algoritmo).



Los protocolos que se utilizan generalmente para HASH son:

* MD5 – Message Digest 5 con valores de hash de 128 bits
* SHA-1 – Secure Hash Algorithm con valores de hash de 160 bits
* SHA-2 – con hash de 224, 256, 384 y 512 bits

Lo que el emisor hace es cifrar el HASH con la clave privada, y enviarlo junto a los datos (en texto plano y sin confidencialidad), de esta forma el receptor descifra el hash que se le envió haciendo uso de la clave pública del emisor, y obtiene el hash descifrado. Luego lo comparar con el hash que él mismo calculó en función del mensaje que le llego, y los comprara. De esta forma garantiza autenticidad mediante la encriptación del hash e integridad mediante el hash (si se modifica el documento, el hash calculado será distinto). Por esto que se menciona, al hash se le suele llamar “huella digital”, ya que si se cambia un solo valor en el mensaje, el valor de hash se modifica.

**Firma Digital:**

Hoy en día se realizan transacciones comerciales a través de internet, y la forma de garantizar que dicha transacción la hizo una determinada persona, que nadie la modifico y que dicha persona no se arrepienta de haberlo hecho, es haciendo uso de las firmas digitales.

Estas deben garantizar:

 Que el receptor pueda verificar la identidad del transmisor

 Que el emisor no pueda repudiar después el contenido del mensaje

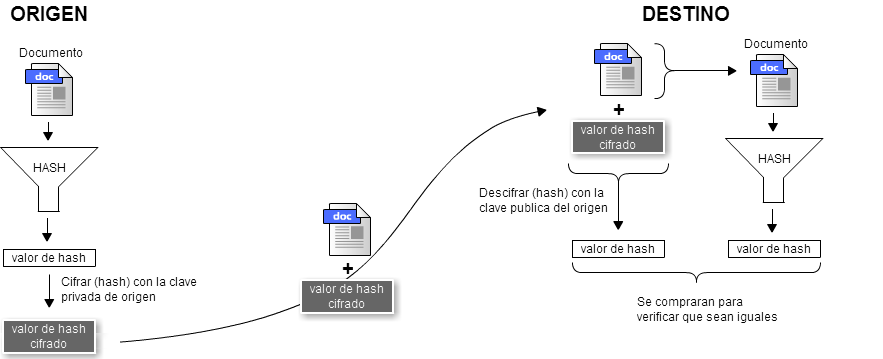
 Que el receptor no haya podido confeccionar el mensaje él mismo

* Que no se haya alterado el contenido del mensaje

La firma digital, no garantiza confidencialidad. Existen firmas de clave simétrica y firmas de clave asimétrica (o clave pública).

Firmas digitales de clave pública:

El algoritmo que se utiliza es el RSA. Analicemos el siguiente gráfico:



Se puede observar que en el origen lo primero que se realiza es aplicar el algoritmo de hash al documento o información que se desea transmitir, obteniendo un valor de hash.

A su vez cifra este valor con su propia clave privada, y junto con el documento, los envía al destino (le adosa la firma digital).

En el destino, se separan los datos trabajándolos en paralelo. Por un lado se trabaja el documento calculando nuevamente el hash y por otro lado a partir del hash cifrado, se descifra con la clave pública del origen (garantizando autenticidad y no repudio) obteniendo el valor de hash.

El valor de hash calculado a partir del documento se comprara con el valor de hash descifrado, y si son iguales significa que el documento no sufrió modificaciones por terceros en la transmisión (garantizando integridad de los datos).

De esta forma se garantiza autenticidad e integridad sin la necesidad de cifrar todo el documento, ahorrando tiempo de cifrado. Hay que aclarar que esta metodología no garantiza confidencialidad de los datos.