Resumen segundo parcial TD IV

Juan Ignacio Elosegui

CAPA DE RED

La tarea de la capa de red es mover paquetes desde un host emisor hacia un host receptor. Para hacer esto, toma segmentos de la capa de transporte del host emisor, los convierte en paquetes, y los envía a un router cercano. El host receptor recibe estos paquetes mediante un router, extrae los segmentos para la capa de transporte y los envía para su capa de transporte.

¿Qué hay adentro de un router?
A nivel hardware, tenemos los puertos de entrada, los puertos de salida y el *switch fabric*.

Puerto de entrada: termina un enlace físico entrante al router. Acá la tabla de forwarding es consultada para saber por qué puerto de salida debe irse el paquete.

Puerto de salida: almacena los paquetes recibidos de la switch fabric y los transmite por el enlace saliente.

Switch fabric: conecta los puertos de entrada con los de salida.

A nivel software está el procesador de ruteo, que ejecuta los protocolos de ruteo y guarda la información de las tablas.

La capa de red puede ofrecer los siguientes servicios:

- Entrega garantizada y en menos de cierto tiempo (delay)
- Entrega en orden de los paquetes
- Uso de ancho de banda mínimo
- Seguridad

Sin embargo, provee sólo esto la capa de red de la Internet:

- Servicio best-effort: no se garantiza nada de lo que nombré arriba, si no que hace todo lo posible para que se cumplan medianamente bien.

CAPA DE RED: PLANO DE DATOS

Recibe los paquetes por sus puertos de entrada y los reenvía (forwarding) por los puertos de salida. Se realiza mediante hardware.

Un elemento clave del forwarding es la tabla de forwarding. Un router reenvía un paquete mirando los valores de uno o más headers de los paquetes, y en base a ese valor se determina por qué enlace de salida (interfaz) debe irse.

Una tabla de forwarding típicamente tiene rangos de direcciones IP asociadas con un puerto de salida (o interfaz). Si las direcciones IP tienen 32 bits de largo, es totalmente imposible que una tabla de ruteo tenga 2^32 entradas. Por lo tanto, el router intenta matchear (en el puerto de entrada) un prefijo de la dirección IP destino del paquete con alguna de las entradas de la tabla. Si matchea alguna, se reenvía el paquete por la interfaz que mejor matcheó.
Si hay varios matches, el router usa el matcheo más largo del prefijo.
Los paquetes entran por los puertos de entrada, se averigua por este método de matcheo por cuál interfaz deben irse, pero antes deben pasar por la switching fabric. En algunos casos, estos paquetes pueden verse bloqueados en caso de que algún otro paquete esté usando la switching fabric.

El proceso de switching (direccionar puertos de entrada con puertos de salida) se puede dar de varias maneras: por memoria, por un bus o por una red de interconexión.
- Por memoria: el paquete entrante es copiado al puerto de salida adecuado luego de que el procesador de ruteo vea la dirección en el header del paquete.
- Por un bus: el paquete va derecho al puerto de salida sin que intervenga el procesador. Todos los puertos de salida reciben el paquete, pero sólo el adecuado se lo queda.
- Por red de interconexión: usando un crossbar (interconecta buses) los paquetes no pueden quedarse trabados siempre y cuando no esté más de uno usando un puerto de salida.

El queuing (cuando se traban los paquetes) puede pasar en los puertos de entrada y/o de salida, y su importancia depende del tráfico, la velocidad del switch fabric y de la velocidad de la línea (enlace físico).

Queuing en el input: sucede cuando la velocidad del switch fabric no es suficiente para direccionar todos los paquetes que llegan. Muchos paquetes pueden ser transferidos en simultáneo, siempre y cuando los puertos de salida que requieran sean distintos. Pero ya cuando dos paquetes al frente del puerto de entrada quieren salir por el mismo puerto de salida, uno de ellos se quedará bloqueado esperando. Este fenómeno se llama **head-of-line blocking (HOL)**.

Queuing en el output: es cuando los paquetes se quedan trabados justamente en los puertos de salida. Esto puede resultar en un problema si es que la tasa de llegada de paquetes a los puertos es más rápida que la tasa en la que se van. Cuando pasa esto, se descartan paquetes de manera aleatoria (RED) o se asignan probabilidades de descarte según su prioridad (WRED), todo esto para evitar la congestión. Si se congestionan los puertos de salida, se demoran las transmisiones de paquetes.

Principio de neutralidad de red:
- Quienes proveen servicios de acceso a internet no deben bloquear contenido legítimo, ni aplicaciones, ni servicios, ni contenidos no dañinos.
- No a la prioridad paga
- No perjudicar al acceso de contenidos legítimos

Este plano opera en términos de nanosegundos.

Internet Protocol (IP)

Hay dos versiones de IP en uso, IPv4 e IPv6. Recordemos que el paquete, en la capa de red, se llama datagrama. El datagrama sigue una semántica bien estructurada, más allá de que sean sólo bits ordenados. Tiene un formato que reúne campos como versión, longitud del header, tipo de servicio, longitud del datagrama (en bytes), identificador de 16 bits, flags, TTL, dirección IP origen y destino de 32 bits, opciones, datos, etcétera.
Los datagramas tienen un total de 20 bytes de header (si es que no tiene opciones preconfiguradas), pero si contiene un segmento TCP adentro, entonces cada datagrama tiene un total de 40 bytes de header (20 de header IP y 20 de header TCP) junto al mensaje para la capa de aplicación.

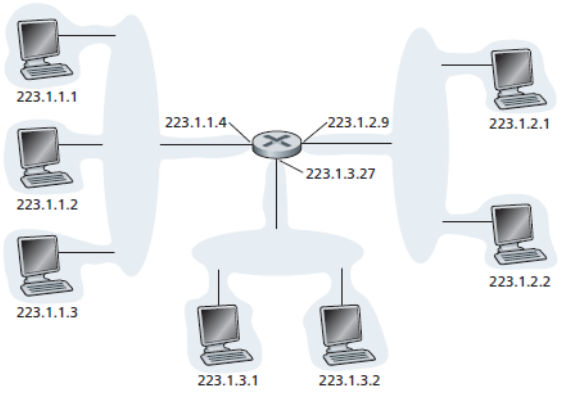
En cuanto al direccionamiento de IPv4, cabe aclarar cómo los hosts y los routers se conectan a la Internet: un host comúnmente tiene un sólo enlace a la red, y cuando el protocolo IP quiere enviar un datagrama, lo hace por este enlace. Lo que divide al host ó router del enlace físico es la interfaz. Como el trabajo del router es recibir datagramas por un enlace y

reenviarlos por otro, entonces un router va a tener múltiples interfaces, una para cada enlace. Cada interfaz que tenga cada router ó host, debe tener su dirección IP. Así, las direcciones IP están asociadas con las interfaces, más allá del sistema que contenga esa interfaz.

Las direcciones IP contienen 32 bits (4 bytes), por lo que hay 2^32 direcciones IP posibles. Estas direcciones están escritas en decimales con puntos, esto es que cada byte se escribe en su formato decimal separado con puntos. Por ejemplo: 193.32.216.9, sería el 193: 11000001, el 32: 00100000, y así sucesivamente. Notar que el límite por cada byte es 255.

Así, cada interfaz debe tener una dirección IP globalmente única (a excepción de NAT – que vamos a ver más adelante-), pero no se elige esta dirección de manera arbitraria, si no que una porción de esta dirección IP se determina por la subnet a la cual está conectado el router/host. Este método de asignación se llama CIDR.

Se puede dar un caso en el cual haya tres hosts interconectados a una sola interfaz de un router (parecen no dar los números), por ejemplo. En términos de IP, a esto se le llama subnet.



En la foto de la izquierda, el direccionamiento IP le asigna una dirección a esta subnet, 223.1.1.0/24, donde el "/24" representa la máscara de la subnet, que sirve para definir que los 24 bits más a la izquierda de los 32 que tienen las direccioens definen la dirección de la subnet.

Entonces, nos queda que hay tres interfaces de host conectadas a una sola interfaz de router. Cualquier otro host que se quiera sumar a esa subnet deberá tener una dirección como 223.1.1.xxx (porque los últimos 8 bits -xxx- se reservan para los integrantes de la subnet)

En total tenemos tres subnets: 223.1.1.0/24 (izquierda), 223.1.2.0/24 (derecha) y 223.1.3.0/24 (abajo).

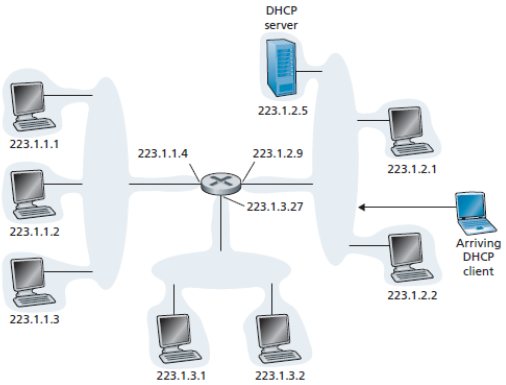
Para conseguir un bloque de direcciones IP para usar en una subnet de una organización, por ejemplo, el administrador de esta debe contactar a su ISP, quien le va a dar un bloque reducido de todo el bloque completo que le dieron al ISP.

Dynamic Host Configuration Protocol (DHCP)

Cuando la organización recibe este bloque de direcciones, puede asignar direcciones IP individuales a las interfaces de los hosts y los routers que haya dentro de la organización. Esto se puede hacer de manera manual (con un administrador) o de manera automática usando el protocolo DHCP (Dynamic Host Configuration Protocol).
El administrador de la subnet puede configurar este protocolo para que cada vez que se conecte un host a la subnet, se le asigne siempre la misma dirección IP o una dirección IP temporaria distinta. Además, este protocolo le permite a un host saber su máscara de subnet, su router first-hop (el gateway por defecto) y la dirección de su servidor DNS, si quisiera.

Este protocolo suele ser plug-and-play y no necesita ser configurado.
DHCP es un protocolo cliente-servidor.

Un cliente es un host nuevo que se suma a la subnet queriendo saber la información de la configuración de la red, además de querer una IP exclusiva para él. En un caso simple y hermético, todas las subnets suelen tener su servidor DHCP. Si no lo tiene, se incluye un agente de retransmisión DHCP (generalmente un router) que sirve para justamente enviar y recibir requests y respuestas al cliente si el servidor DHCP está en otra subnet.



NOTAR QUE CLIENTE = NUEVO HOST QUE QUIERE ENTRAR A LA SUBNET
LOS USO COMO SINÓNIMOS

Cuando un cliente (un nuevo host) se quiere sumar a la subnet, se siguen cuatro pasos:

- Buscar con qué servidor DHCP interactuar. Se hace esto mandando un mensaje tipo *discovery* DHCP via UDP. Pero el nuevo host no sabe la dirección IP a la cual se está conectando ni mucho menos la dirección IP del servidor DHCP, por lo tanto, este mensaje -luego de haberse pasado a la capa de enlace- les llega a todos los nodos en la subnet (usando la dirección IP destino 255.255.255.255 y la IP origen de 0.0.0.0 para referirse al host que quiere ingresar).
- Ofertas de servidores DHCP. Cuando un servidor DHCP recibe el discovery DHCP, hace un mensaje de oferta (que también reciben todos los nodos con IP dst 255.255.255.255) mostrándole al nuevo host qué dirección IP puede usar, cuál es su tiempo de validez y cuál es la máscara de subnet.
- DHCP request. Como puede haber varios servidores DHCP en una subnet, el nuevo host puede elegir entre una o más ofertas que se le hicieron. Para decidirse, le manda un request DHCP al servidor DHCP que más le gustó.
- DHCP ACK. El servidor DHCP le responde al mensaje request DHCP que hizo el cliente interesado con un mensaje ACK DHCP.

Cuando se recibe el ACK DHCP, se completa la interacción.

Network Address Translation (NAT)

Como dijimos antes, sabemos que cada dispositivo que corra el protocolo IP justamente necesita una dirección IP. Pero hay que tener en cuenta que las subnets, como puede ser el caso del internet de tu casa, pueden tener cuatro dispositivos conectados fácilmente. Esto quiere decir que siempre tu ISP va a tener que designar bloques de direcciones IP para cubrir direcciones IP de subnets domésticas.

Para acceder a Internet, se necesita una sola dirección IP pública, pero podemos usar varias direcciones IP privadas en nuestra subnet. Lo que propone NAT es permitir que muchísimos dispositivos accedan a Internet bajo una única dirección IP pública. Para conseguir esto, se debe hacer una traducción de la dirección privada a pública y viceversa, además de la traducción de puertos.

Generalmente, el router de la frontera (gateway) se configura para NAT. Cuando un paquete quiere salir de la red local (subnet), lo que hace NAT es convertir la dirección IP privada del host que mandó ese paquete hacia una dirección IP pública. Cuando un paquete quiere entrar a la red local, se hace el proceso inverso.

CAPA DE RED: PLANO DE CONTROL

Coordina las acciones de forwarding por cada router para asegurarse de que los paquetes lleguen al destino esperado, coordinando sus rutas (ruteo). Se realiza mediante software.

Las tablas de forwarding se configuran con el algoritmo de ruteo que mencioné anteriormente. El algoritmo de ruteo de un router cualquiera se comunica con el algoritmo de ruteo de los otros routers para calcular los valores de esta tabla.
También, pueden existir controladores remotos físicos que calculan y distribuyen las tablas de ruteo para todos los routers.

Este plano opera en milisegundos o segundos.

Los nodos son los routers o hosts en una red.

Los algoritmos de ruteo se encargan de determinar buenas rutas que relacionen al emisor y al receptor. Generalmente, una "buena ruta" es aquella que tiene menos costo.
Los algoritmos de ruteo se clasifican en:

- Centralizado: calcula el camino de menor costo desde un emisor hacia un receptor usando un conocimiento completo de la red (esto es, conocer todos los enlaces y sus costos). Típicamente, se dice que se conoce la *topología* completa. Pero, para hacer esto, el algoritmo debe obtener de alguna manera toda esta información. A los algoritmos de ruteo centralizados se los denomina algoritmos de link-state.
- Descentralizados: cada nodo sabe el costo de sus enlaces a nodos adyacentes, por lo que el cálculo de la ruta de menor costo la hacen todos los routers de manera iterativa. Estos algoritmos de ruteo se denominan algoritmos distance-vector porque cada nodo tiene un vector de estimaciones de costos a todos los otros nodos de la red.

En un algoritmo link-state, se conoce la topología de la red y los costos de los enlaces. Esta información la tienen todos los nodos, y es posible porque ellos mismos broadcastean (emiten información) paquetes a todos los otros nodos en la red, con los paquetes teniendo identificadores y costos de sus enlaces.

Algoritmo Dijkstra

Un algoritmo link-state es el Dijkstra. Dijkstra computa la ruta de menor costo de manera iterativa desde un nodo u hacia todos los otros nodos de la red. Una propiedad importante es que, luego de la iteración k del algoritmo, las rutas de menor costo son conocidas por k nodos, consiguiendo así que k nodos obtengan los k costos menores.

-Tenemos que D(v) es el costo de la ruta de menor costo desde el nodo origen (u) hacia el nodo destino (v).
- Sea p(v) el nodo anterior, vecino de v, que esté en la ruta del menor costo.
- Sea N', un subconjunto de nodos, v está en N' si la ruta de menor costo desde u hasta v es conocida.

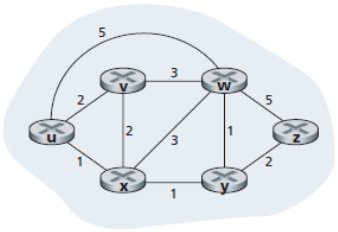
Pseudocódigo:

El subconjunto N' está sólo incluido por u.
El costo entre dos nodos no adyacentes (no vecinos) es infinito.
De manera iterativa, se buscan enlaces que sean más baratos en términos de costo, y se agregan a N' por cada nodo que se consulte.
Se repite el loop hasta que se consulten todos los nodos buscando mejores alternativas.

```
Initialization:
 N' = {u}
 for all nodes v
   if v is a neighbor of u
     then D(v) = c(u,v)
   else D(v) = ∞

Loop
 find w not in N' such that D(w) is a minimum
 add w to N'
 update D(v) for each neighbor v of w and not in N':
       D(v) = min(D(v), D(w)+ c(w,v) )
  /* new cost to v is either old cost to v or known
   least path cost to w plus cost from w to v */
until N' = N
```

En el paso 0, partimos desde el nodo u, que es el primer nodo consultado (por eso se agrega a N').
Se calculan los costos de cada enlace hacia los vecinos de u. Notar que v, x y w son vecinos de u, por lo que el costo se puede calcular directamente. Como y y z no son vecinos de u, se pone su costo como infinito. Siempre que u tenga un nodo vecino, por ejemplo j, se cumplirá que p(j) = u.



step	N'	D(v), p(v)	D(w), p(w)	D(x), p(x)	D(y), p(y)	D(z), p(z)
0	u	2, u	5, u	1,u	∞	∞
1	ux	2, u	4, x		2, x	∞
2	uxy	2, u	3, y			4, y
3	uxyv		3, y			4, y
4	uxyvw					4, y
5	uxyvwz					

En el paso 1, consultamos por x, por lo que se agrega al subconjunto N'. Ahora hay que entender que estamos parados en el nodo x. Sigue siendo cierto que el costo desde u hasta v es 2, y que, como v es vecino de u, entonces p(v) = u. Pero para que conectemos a u con w, lo ideal sería pasar por x, dado que es menos costoso. Entonces, D(w) = C(u, x) + C(x, w). Como estamos parados en x, no tiene sentido computar D(x) ni p(x). x es vecino de y, por lo que D(y) = C(u, x) + C(x, y). Como x no es vecino de z, dejamos su costo como infinito.
En el paso 2, pasamos a analizar el nodo y. La ruta desde u hacia w, es decir, D(w), cambió, porque la que encontramos (que pasa por x e y) es más barata. Notar que la manera más barata de acceder a z es mediante x e y también.
En el paso 3, analizando el nodo v, no encontramos mejores rutas.
En el paso 4 pasa lo mismo.
En el paso 5 se termina el algoritmo de ruteo, porque ya analizamos todos los nodos.

Como se puede ver, para cada nodo tenemos su nodo anterior -su predecesor- que nos lleva para atrás en la mejor ruta hacia el nodo origen. Esto se cumple para los algoritmos link-state.
Al mismo tiempo, se va configurando la tabla de forwarding para el nodo u, indicando el *next hop* (próximo nodo a visitar) hacia el destino.

Algoritmo Distance-Vector

También es un algoritmo link-state. Este es un algoritmo iterativo, asincrónico y distribuido.
Es iterativo porque este proceso de cálculos sigue hasta que ya no haya más información nueva para ofrecerle a los vecinos (termina solo).
Es asincrónico porque no es necesario que todos los nodos estén calculando en simultáneo con los otros.
Es distribuido porque todos los nodos reciben alguna información de uno o más vecinos, se hacen cálculos en base a eso, y los resultados se comparten con los vecinos.

La explicación que dio Jim Kurose es muy profesional, al menos para la materia. Se pone a hablar de ecuaciones matemáticas y lo único que quiero yo es entender este algoritmo, así que lo voy a explicar a mi manera.

La tarea que tiene este algoritmo es la misma que la de cualquier algoritmo de ruteo: hallar la mejor y más barata ruta desde un nodo origen a uno destino.

Al final voy a tener que explicar más o menos de manera matemática la ecuación de Bellman-Ford. No se puede entender de otra manera si no, una lástima.

La distancia más corta entre los routers es un vector de distancia. Estos vectores se guardan en una tabla por cada router. Los vectores estos se calculan usando la información de los vectores de los vecinos.

El proceso es el siguiente: un nodo transmite su vector de distancia a sus vecinos. Cada nodo recibe y almacena el vector de distancia más reciente de cada vecino. Un nodo va a recalcular su vector de distancia si recibe un vector con información distinta que el de antes, o si descubre que un enlace a un nodo vecino se cayó.

La parte matemática es esta.
Suponer que tenemos que Dx(y) es el costo mínimo desde x a y.
x tiene un vecino, v, por lo tanto conoce su costo: C(x, v).
La ecuación de Bellman-Ford establece una relación entre los costos de las rutas de menor costo: Dx(y) = minv{ C(x, v) + Dv(y) }. Esto es como decir que la ruta más barata desde Núñez (x) a Palermo (y) es la misma ruta que si tomara el camino más rápido desde Núñez hasta Belgrano (v) y de Belgrano a Palermo.
Ruta más rápida de Núñez a Palermo = Dx(y)
Camino más rápido de Núñez a Belgrano (son vecinos) = C(x, v)
Ruta más rápida de Belgrano a Palermo = Dv(y)

Lo importante de esta ecuación extraña es que justamente sus soluciones son las que terminan apareciendo en la tabla de forwarding del nodo x.

Ruteo Intra-AS

En los algoritmos que recién dije se asumen que todos los routers son iguales, en el sentido que todos ejecutan el mismo algoritmo de ruteo para calcular las rutas a través de toda la red. Esto no existe.

Hoy por hoy debe haber cientos de millones de routers en el mundo. Almacenar la información de todos los destinos posibles requeriría una cantidad impresionante de memoria, además, un algoritmo de ruteo común y corriente no llegaría posiblemente a ningún resultado jamás entre tantos nodos.

Además, la Internet es una red de ISPs. Cada ISP tiene su propia red de routers. Un ISP generalmente desea operar su red como desee (como elegir un algoritmo de ruteo) o bien puede querer esconder aspectos de su organización interna de la red hacia los otros ISPs, pero lo importante es que todos los ISPs se sigan comunicando entre ellos.

Estos problemas se resuelven agrupando routers en sistemas autónomos (AS) bajo el mismo control administrativo. A veces, los routers de un ISP y sus respectivos enlaces constituyen un único AS, o también los mismos ISPs particionan su red en muchos ASs.

En definitiva, los routers se organizan en ASs para tener el mismo algoritmo de ruteo y para tener información acerca de ellos entre ellos.

Open Shortest Path First (OSPF)

Es un protocolo link-state que usa el *flooding* de información link-state y el algoritmo de ruteo de menor costo Dijkstra.

Notar que no estoy hablando de algoritmo de ruteo y protocolo de ruteo como si fueran lo mismo. No lo son.

Con OSPF, cada router construye un mapa topológico (una vista completa del mapa de routers) del sistema autónomo. Luego, cada router ejecuta Dijkstra para determinar el camino más corto a todas las subnets que pueda haber dentro del sistema autónomo.
Un router hace un mensaje broadcast no sólo a sus routers vecinos, si no que a todos los que estén dentro del sistema. Va a hacer esto de manera periódica o si hay un cambio en el costo o si se cayó un router.

Ruteo Inter-AS

Si tuviera que enviar un paquete hacia un destino que está dentro del mismo AS que yo, la ruta que va a tomar el paquete va a estar 100% determinada por el protocolo Intra-AS. Pero, si quiero mandar un paquete desde Viedma hasta Rusia, vamos a necesitar un protocolo de ruteo Inter-AS.

Si bien un protocolo de ruteo Inter-AS exige coordinación entre varios AS, esos sistemas autónomos también deben tener el mismo protocolo de ruteo. Todos los AS corren el mismo: Border Gateway Protocol (BGP).

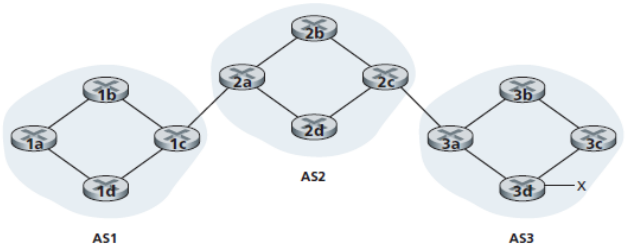
Border Gateway Protocol (BGP)

Es el pegamento entre los miles de ISPs en Internet. Funciona de manera descentralizada y asincrónica.

Como dije, necesitamos enviar un paquete desde un AS a otro AS distinto. No había problemas si el paquete tiene como destinatario alguien que está dentro del mismo AS que nosotros. Si queremos que nuestro paquete salga de nuestro AS y se meta en otro AS, en BGP, los paquetes no están direccionados a una dirección destino específica, si no que sigue prefijos, que puede ser una subnet o varias.

BGP le permite a las subnets anunciar su existencia para el resto de la Internet. Si no fuera por esto, cada subnet sería como una isla a la cual nadie conoce ni se puede acceder. También, si un router consigue más de una ruta posible para un prefijo cualquiera, BGP puede ayudarle a seleccionar la mejor ruta.

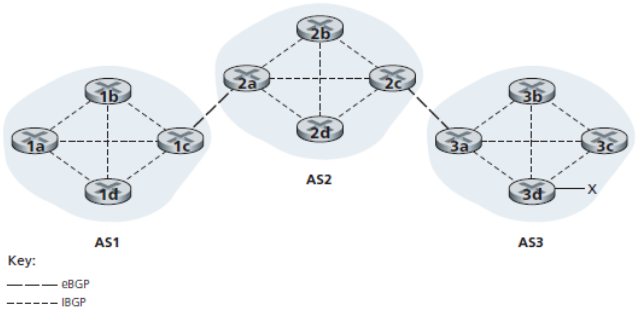
¿Cómo anuncia la subnet su existencia?
Todos los AS tienen dos tipos de routers: gateway o interno. Un router gateway está al borde del AS y se comunica con uno o más routers en otro AS. Un router interno se conecta a los hosts y a otros routers dentro del AS.



Sea x un prefijo en el gráfico.
AS3 le manda un mensaje BGP a AS2, diciendo que x existe, y que está dentro de AS3.

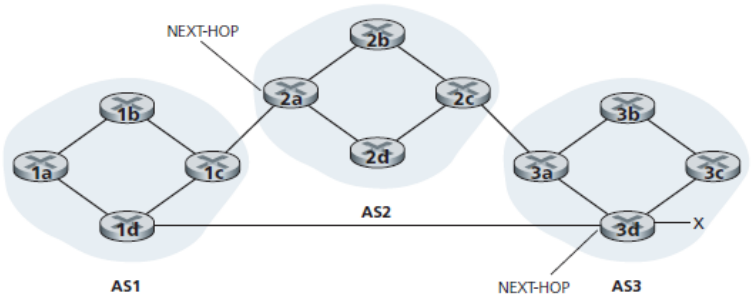
De la misma manera, AS2 le manda un mensaje BGP a AS1, diciendo "x existe, y podés llegar a x pasando por AS2 y después por AS3".
Este mensaje es algo como "AS2 AS3 x"
No olvidarse que sistema autónomo = AS
Así, todos los sistemas autónomos no sólo van a enterarse que existe x, si no que también existe un camino de sistemas autónomos que te van a llevar a x.

Pero recordemos que estos mensajes no se los mandan técnicamente los sistemas autónomos, si no los routers que están dentro de ellos. Una conexión BGP incluye una conexión semipermanente TCP junto a los mensajes BGP.



Una conexión que involucra dos ASs se la llama *external BGP* (eBGP), y una conexión entre los routers en el mismo AS es una *internal BGP* (iBGP).
La conexión eBGP se da entre los routers gateway de dos AS distintos, mientras que la conexión iBGP se da entre todos los routers (incluyendo al gateway) de un AS.

Para cerrar con BGP, cabe aclarar un algoritmo de ruteo que funciona en BGP: Hot Potato.
Tenemos el siguiente esquema:



Supongamos que estamos parados en el router 1b. Este router aprendió dos rutas BGP posibles para el prefijo x. En este algoritmo de ruteo, la ruta elegida será aquella con el mínimo costo al router next-hop que comience esa ruta.
Acá, 1b va a consultar por su información dentro de AS1 para hallar el next-hop menos costoso para llegar a 2a (de manera intra-AS), pero también va a querer saber acerca del next-hop menos costoso para llegar a 3d (de manera inter-AS).

Si el costo estuviera definido como la cantidad de enlaces que se deben cruzar, entonces el costo de viajar desde 1b hacia 2a es 2, y si quisiéramos ir desde 1b hacia 2c será 4. Entonces el router 2a será elegido.

Ahora, ¿por qué no se viaja derecho para AS3, si se cruzan menos enlaces? Esto es porque se trata de literalmente una papa caliente, ya que 1b se quiere sacar de encima sus paquetes la manera más rápida posible. Se apunta a elegir el router gateway más barato para acceder y sacar el paquete del AS.
Si el costo fuera la cantidad de enlaces cruzados, entonces es más barato ir desde 1b hacia 2a que desde 1b hacia 3d, porque se atraviesan 2 enlaces en vez de 3.

CAPA DE ENLACE

Introducción

La pregunta que se va a intentar responder en esta parte es, ¿cómo se mandan los paquetes a través de los enlaces individuales que intervienen en una ruta completa end-to-end? Vamos a ver que existen dos tipos de canales en la capa de enlace: los canales de broadcast y los enlaces de comunicación point-to-point.

Tener en cuenta que, en la capa de enlace, cada dispositivo que corra un protocolo de capa de enlace se le va a decir nodo.

Los nodos pueden ser hosts, routers, switches y puntos de acceso WiFi.
Los enlaces serán los canales de comunicación que conectan nodos vecinos a través del camino de la comunicación. En un enlace cualquiera, el nodo transmisor encapsula al datagrama en un *frame*, el cual viajará por ese canal.

Para ponerse en contexto cabe narrar un ejemplo.
Supongamos que estoy en mi casa en Viedma y quiero viajar a Madrid, entonces para eso contrato a una agencia de viajes. La agencia de viajes me dice que es conveniente que me tome un taxi desde mi casa a la terminal de Viedma, viaje a Ezeiza en colectivo, y desde Ezeiza voy derecho a Madrid en avión.
Cuando la agencia de viaje me hizo las tres reservas (taxi, colectivo y avión), el taxista ahora tiene la responsabilidad de llevarme al aeropuerto, el chofer del colectivo debe llevarme a Ezeiza y el piloto de avión debe dejarme en Madrid.
Ahora, los tres tramos de transporte están manejados por tres personas distintas, y usan tres métodos de transporte distintos. Más allá de eso, me pueden llevar desde un lado a otro.
En esta analogía yo sería el datagrama, los tramos de transporte son los enlaces y la agencia de viajes es el protocolo de ruteo.

Servicios

- Framing. Un frame es un campo de datos en el cual se inserta el datagrama proveniente de la capa de red junto con headers. La estructura del frame está especificada por el protocolo de la capa de enlace.
- Acceso al enlace. Un protocolo *medium access control* (MAC) pone las reglas por las cuales el frame se transmite hacia el enlace. Los enlaces point-to-point que tienen un único emisor en un punto y también un único receptor en el otro punto del enlace, el protocolo MAC puede enviar un frame cuando el enlace esté en desuso.
- Entrega confiable. Garantiza que los datagramas de la capa de red se muevan por los enlaces sin errores. Al igual que TCP en la capa de transporte, se puede conseguir esto mediante ACKs y retransmisiones. Esto sirve mucho para los enlaces de tipo *wireless*, pero no tanto para enlaces que incluyan cables.
- Detección y corrección de errores. El hardware de la capa de enlace de un nodo receptor puede tranquilamente decidir que un bit en un frame es un 0 en vez de un 1 o viceversa, dado al ruido electromagnético que producen los componentes de los nodos. Para que no se transmitan datagramas corruptos, en el frame se incluyen bits de detección de errores.

Además de detectar los errores, puede saber exactamente en qué bit están los errores, y los corrige.

Implementación

La capa de enlace en un host, ¿está implementada en hardware, software, o se implementa en un chip por separado? ¿cómo interactúa con los otros componentes de hardware de un host?

La capa de enlace se implementa en un chip llamado adaptador de red. Este controlador puede hacer varias de las cosas que nombré antes: framing, acceso al enlace, detección de errores... por lo que la mayoría de las cosas se hacen via hardware.

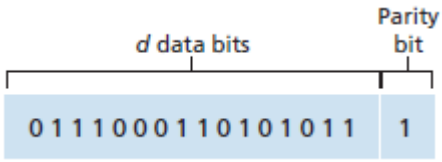
En el lado del nodo emisor, este chip, o controlador, toma el datagrama que creó la capa de red, lo encapsula en un frame, y lo envía por el enlace siguiendo el protocolo de acceso de enlace.

Por el lado del nodo receptor, el controlador recibe este frame y extrae los datagramas para la capa de red.

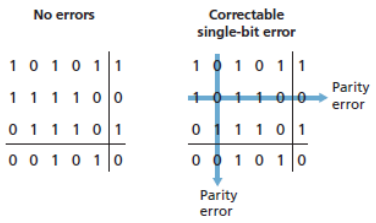
Cabe aclarar que, si el enlace tiene habilitada la detección de errores, el controlador del emisor pone los bits de detección de errores en el header del frame y el controlador del receptor es quien hace la detección de errores en sí.

Métodos de detección y corrección de errores

Control de paridad: supongamos que la información que queremos enviar es D, con d bits.
En un esquema de paridad par, el emisor incluye un solo bit de paridad adicional. El valor de ese bit agregado va a ser tal que la cantidad de 1s en los d+1 bits sea par.



En un esquema de paridad impar, el bit de paridad agregado va a ser tal que la cantidad de 1s en los d+1 bits sea impar.
Este es un esquema de paridad par. Notar que tenemos nueve 1s. Como queremos que haya una cantidad par de 1s (porque así lo pide el esquema), agregamos un bit de paridad 1, que nos deja con diez 1s (número par). De manera análoga, se puede hacer lo mismo con los 0s, agregando un bit de paridad que tenga valor 0.
El receptor, cuando le llegan estos d+1 bits, lo único que tiene que hacer es contar la cantidad de 1s. Si se cuentan una cantidad de 1s impar, en un esquema de paridad par, se va a enterar que hubo al menos un error en los bits.
Más precisamente, va a saber que hubo una cantidad impar de errores en los bits, porque si hubiera una cantidad par, no se detecta el error.



Para no sufrir este fenómeno, se pueden usar los esquemas de paridad par de dos

dimensiones, que organiza los bits en columnas y filas. Esta modificación del control de paridad puede detectar exactamente en qué bit está el error y también lo puede corregir.

Métodos de checksum: los d bits de datos se ven como una secuencia de enteros. Se suman estos números y se usa su resultado para detectar errores en los bits (así funciona el checksum del Internet). El receptor chequea el checksum tomando el complemento a uno de los datos recibidos (incluyendo al checksum) y se fija que el resultado de todo es 0 bits. Si algún bit da 1, se indica un error.
Recordar que cuando yo sumo un número con su complemento a uno debe dar cero.

Cyclic Redundancy Check (CRC): no es tan importante la explicación matemática, pero básicamente lo que hace es ver a los de bits como una función polinómica, y en base al resto de dividir este polinomio por otro polinomio generado se pueden detectar errores.

Protocolos y enlaces de acceso múltiple

Recordemos que había dos tipos de enlaces de red: los enlaces point-to-point y los enlaces de broadcast.
Un enlace point-to-point (o punto a punto) tiene un único emisor en un extremo de la red y un único receptor al otro extremo de la red.
Un enlace broadcast es aquel que tiene muchos nodos emisores y receptores, todos conectados por el mismo y único canal de broadcast compartido. El término "broadcast" se usa acá porque cuando algún nodo transmite un frame, todos los otros nodos reciben una copia del mismo frame. Un ejemplo de canal broadcast puede ser Ethernet o una LAN wireless.

El problema que se nos presenta es cómo se puede coordinar el acceso de múltiples emisores y receptores en un canal de broadcast compartido.

Para poner un ejemplo, si estoy en un aula, los profesores y los alumnos comparten el mismo canal de broadcast, que vendría a ser el aire.
El problema radica en cómo se determina quién habla y cuándo, o sea, quién transmite en el canal.

Entonces, las redes de computadoras tienen protocolos para regular la transmisión de los nodos por el canal.

Como todos los nodos pueden transmitir frames, puede tranquilamente pasar que dos o más quieran usar el canal al mismo tiempo para transmitir. Cuando pasa esto, todos los nodos van a recibir dos o más frames al mismo tiempo. Este fenómeno se llama colisión, y cuando sucede, los nodos no pueden entender los frames que le llegan, por lo tanto, estos frames se pierden y se malgasta el uso del canal mientras se da la colisión.
Como dije, para que esto no suceda, necesitamos coordinar el uso del canal de los nodos activos. Esto es lo que hacen los protocolos.

Protocolo de partición de canal: se usan dos técnicas de multiplexación basada en división de frecuencia y en división de tiempo para particionar el ancho de banda de un canal de broadcast entre todos los nodos que estén conectados a él.
Protocolo de acceso aleatorio: un nodo que esté transmitiendo va a usar el ancho de banda

completo del canal. Cuando haya una colisión, cada nodo involucrado en esa colisión retransmite su frame de manera insistente hasta que no se dé una colisión. Esta retransmisión insistente no es inmediata luego de una colisión, si no que espera un ratito (un delay aleatorio).

Slotted ALOHA: todos los frames tienen la misma cantidad L de bits, el tiempo se divide en slots de L/R segundos (R es la tasa de transmisión del canal), los nodos transmiten frames sólo al comienzo de los slots, los nodos están sincronizados para que cada uno sepa cuándo comienza un slot; y si dos o más frames colisionan, todos los nodos detectan este hecho antes que el slot termine.

Cuando un nodo quiere enviar un frame, espera a que arranque el slot y lo intenta transmitir completo. Si no hay colisión es un éxito, nos olvidamos de ese frame y no hay que retransmitirlo porque justamente viajó por el canal. Si hay una colisión, el nodo detecta la colisión antes que termine el slot, y se va a retransmitir más adelante con probabilidad p hasta que se pueda transmitir por el canal sin colisiones.

Carrier Sense Multiple Access (CSMA): sigue los lineamientos de las conversaciones entre los humanos, como "escuchar antes de hablar" o "si alguien empieza a hablar al mismo tiempo que vos, hacé silencio".

Si alguien habla, esperá a que terminen de hablar. Esto se llama *carrier sensing*, un nodo escucha el canal mientras está transmitiendo. El nodo que quiere transmitir espera hasta que no escuche ninguna transmisión en el canal (CSMA).

Si alguien habla al mismo tiempo que vos, hacé silencio. Esto se llama *collision detection*, un nodo que está transmitiendo escucha al canal mientras transmite. Si detecta que otro nodo está transmitiendo un frame que colisione, para de transmitir, deja que hable el que está molestando en el canal y espera un rato para seguir transmitiendo (CSMA/CD).

Direccionamiento al nivel de enlace

Ya vimos que los routers y los hosts tienen direcciones de capa de red. Pero también tienen direcciones de capa de enlace, ¿para qué necesitamos también tener direcciones en las dos capas?

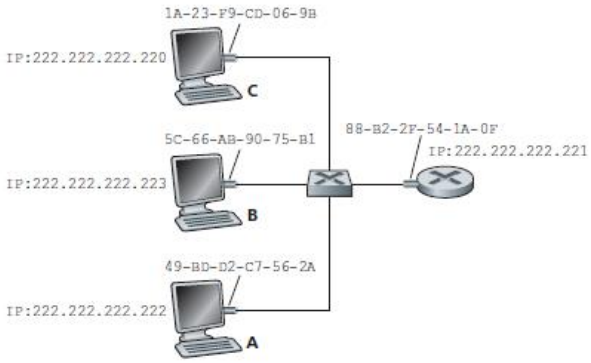
En realidad, no es que los hosts y los routers tienen direcciones de capa de enlace, si no sus interfaces de red. Por lo que, si un dispositivo tiene muchas interfaces de red, tendrá muchas direcciones de capa de enlace y red (IP) asociadas.

Recordar que los switches no tienen direcciones en su capa de enlace, dado que su trabajo es llevar datagramas entre los hosts y los routers de manera transparente.
Aparte, un adaptador es el componente que permite que un dispositivo se conecte a una red.

Una dirección de capa de enlace se llama dirección LAN, dirección física o dirección MAC. De ahora en más le vamos a decir dirección MAC. Las direcciones MAC tienen 6 bytes de largo -48 bits-, por lo que nos provee 2^48 direcciones MAC posibles. Están representadas de manera hexadecimal.
La dirección MAC de un adaptador no cambia por ninguna razón en el mundo, cosa que no es así con las direcciones IP. La dirección MAC es como nuestro DNI, que es el mismo siempre, y la dirección IP es como la dirección de tu casa o tu número de teléfono, que no necesariamente son fijos.

Cuando un adaptador emisor quiere enviar un frame a otro adaptador destino, el adaptador emisor inserta en el frame la dirección MAC del adaptador destino. Luego, lo manda a la LAN. Ahora vamos a ver en mejor detalle que los switches broadcastean el frame que reciben a todas sus interfaces, lo que implica que un adaptador puede recibir un frame que no es realmente para él, si no para otro adaptador. Entonces, todos chequean la dirección MAC dentro del frame entrante para ver si coincide con la dirección MAC suya, para hacerse cargo. A veces, un adaptador quiere mandar un frame para que lo reciban todos los adaptadores en la LAN intencionalmente, para hacer eso, usa una dirección MAC de broadcast FF-FF-FF-FF-FF-FF.

Address Resolution Protocol (ARP)



Como tenemos direcciones de capa de red y también direcciones de capa de enlace, es necesario traducirlas entre ellas.
En este esquema, todos los hosts y routers tienen una sola dirección IP y una sola dirección MAC.
Vamos a asumir que cada vez que el switch reciba un frame por una interfaz, lo va a broadcastear por todas las otras interfaces.

Supongamos que el host C con IP 222.222.222.220 quiere enviar un datagrama IP al host A con IP 222.222.222.222.
Para enviarlo, C debe darle a su adaptador no sólo el datagrama que quiere enviar, si no que también debe ofrecerle la dirección MAC del host A. Por lo que se termina enviando un frame con el datagrama y la dirección MAC destino.

Ahora, ¿cómo sabe el host C la dirección MAC de A? Usa ARP. Un módulo ARP en el host emisor toma cualquier dirección IP en la misma LAN como input, y devuelve la dirección MAC correspondiente.
En el ejemplo, el host C con IP 222.222.222.220 le da a su módulo ARP la dirección IP del host A, 222.222.222.222, entonces el módulo ARP le da la dirección MAC correspondiente, 49:BD:D2:C7:56:2A.

Por lo visto, ARP resuelve una dirección IP a una dirección MAC (algo parecido que hace DNS). Es importante saber que ARP sólo funciona en las LAN (o subnets). No funcionaría si un nodo en Viedma quiere usar ARP para resolver (obtener MAC) una dirección IP de un nodo en Córdoba.

Así funciona ARP, pero ¿cómo funciona? Cada host y router tiene una tabla ARP en su memoria, que contiene los mapeos de direcciones IP a direcciones MAC, además de un TTL para mostrar cuándo se van a borrar los mapeos. Estas tablas no suelen tener una entrada por cada host y router en la subnet.
Supongamos que el host C quiere enviarle un datagrama al host B. Puede parecer una tarea

fácil si es que su tabla de ARP tiene al host B registrado, pero, si no está, el host C debe usar el protocolo ARP para resolver la dirección.
Para hacer esto, C debe armar un paquete ARP y enviarlo a todos los otros hosts y routers en la subnet para determinar la dirección MAC correspondiente a la dirección IP de B que se pasó como parámetro.
Entonces, C le dice a su adaptador que la dirección MAC destino es FF:FF:FF:FF:FF:FF, es decir, le tiene que llegar a todos este frame en la subnet a través de este broadcast.
Cuando todos los adaptadores de la subnet reciben esta query ARP, cada uno le entrega esta query a su módulo ARP. Cada módulo se fija si su dirección IP coincide con la dirección IP de la query, que es la que busca el host C. Si se encuentra una coincidencia, le manda un response ARP al host C con su traducción correspondiente de dirección IP a dirección MAC.

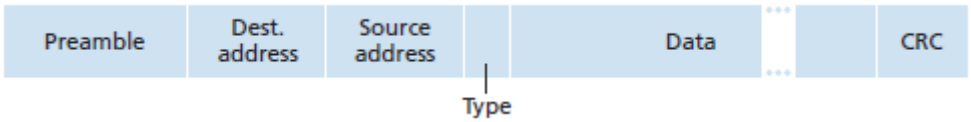
Las queries ARP se envían en un frame de tipo broadcast (se enteran todos) y las responses ARP se envían en un frame estándar. También, ARP es plug-and-play, por lo que se configura automáticamente.

Ethernet

Es una tecnología de LANs cableadas. Tiene dos topologías posibles: una puede ser mediante un bus (cosa que no se sigue usando) o mediante un switch, la cual es libre de colisiones.

Supongamos que existen dos nodos en una LAN Ethernet que quieren mandarse un datagrama IP. El adaptador del nodo emisor encapsula el datagrama IP en un frame Ethernet y se lo pasa a la capa física. El adaptador del nodo receptor recibe este frame Ethernet mediante la capa física, extrae el datagrama IP, y se lo pasa a la capa de red.

Un frame Ethernet tiene esta estructura:



- Preámbulo: despierta a los adaptadores que reciben el frame.
- Dirección destino: contiene la dirección MAC del adaptador que recibe.
- Dirección origen: contiene la dirección MAC del adaptador que envió este frame.
- Tipo: le permite a Ethernet elegir los protocolos de la capa de red (porque no siempre es IP).
- Datos: contiene el payload, que sería el datagrama IP.
- CRC: le permite al adaptador que recibe detectar errores en los bits del frame.

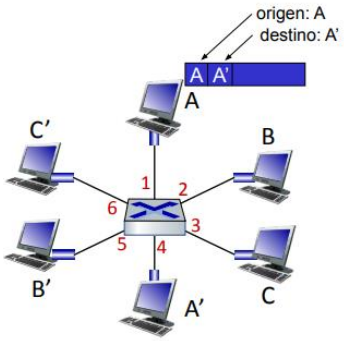
Ethernet es un servicio que no está orientado a conexión (porque no existe el handshaking) y es un servicio no confiable (porque no hay ACKs ni NAKs).

Switches (en Ethernet)

Reciben frames Ethernet de la capa de enlace y los reenvían por sus interfaces. Libre de colisiones y full dúplex (bidireccional).

El switch es transparente para la subnet, ya que ni los hosts ni los routers se enteran si un switch reenvía sus frames. Como es de esperar, la tasa a la que llegan los frames al switch puede ser más alta que la tasa a la cual son reenviados, por lo que tienen buffers en sus interfaces.

Tienen una tabla de switching, por la cual se guían para reenviar los frames por las interfaces.
Esta tabla tiene entradas para algunos hosts y routers en la subnet. Para cada entrada hay una dirección MAC del host/router, la interfaz del switch que nos lleva a esa dirección, y la hora en la cual se incorporó esta información.
Esta tabla se construye de una manera distinta a como lo hacen los routers con sus tablas de forwarding. Se construyen ellas mismas con el autoaprendizaje: los switches pueden adivinar más o menos qué hosts o routers se pueden alcanzar por cuáles interfaces. Cuando un switch recibe un frame, aprende la ubicación del emisor y guarda la dirección MAC del emisor y mediante qué interfaz se recibió el frame.



MAC	interfaz	TTL
A	*1*	*60*

tabla del switch (inicialmente vacía)

El forwarding se hace de la siguiente manera: cuando se recibe un frame, se registra la dirección MAC del emisor que le envió ese frame y la interfaz por el que vino en una entrada de la tabla del switch.
Si esa entrada ya existe y su destino está por la misma interfaz, se descarta el frame. Si la entrada ya existe, pero su destino está por otra interfaz, se reenvía por allí.
Si la entrada no existe, se hace el flooding (reenviar el frame por todas las interfaces).

Los switches se pueden interconectar, y funciona de la misma manera que si tuviéramos un solo switch en una subnet.

Los switches y los routers son store-and-forward y tienen tablas de forwarding.
Pero uno es un dispositivo de nivel de enlace y otro es un dispositivo de red.
Uno aprende las tablas por flooding, autoaprendizaje y direcciones MAC, mientras que otro las calcula con algoritmos de ruteo y direcciones IP.

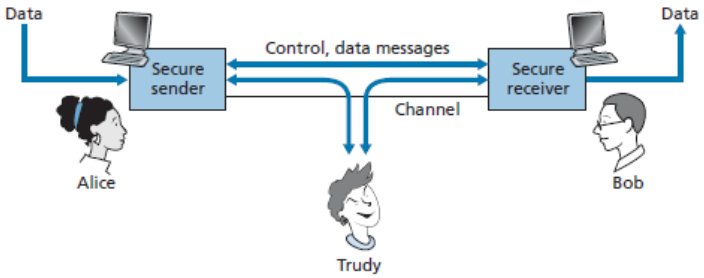
SEGURIDAD

Introducción

Sean Alice y Bob dos personas que se quieren comunicar de forma segura. Alice y Bob pueden ser dos routers que se quieren intercambiar las tablas de ruteo, pueden ser un cliente y un servidor que quieren establecer una conexión de transporte segura o pueden ser dos aplicaciones de e-mail que quieren intercambiarse mails.
"De forma segura" implica que Alice y Bob quieren que su comunicación se mantenga secreta de un tercero (Trudy), implica que quieren verificar sus identidades, e implica que si hay algún tercero escuchando, que esto se detecte.

Queremos que la comunicación segura entre Alice y Bob tenga las siguientes propiedades:
- Confidencialidad. Sólo Alice y Bob deben entender el contenido del mensaje transmitido.
- Integridad de los mensajes. Alice y Bob quieren que el contenido del mensaje no se altere, ya sea de manera accidental o con intenciones maliciosas.
- Autenticación end-to-end. Se deben confirmar las identidades entre ellos dos.
- Seguridad operacional. Las organizaciones tienen redes que están conectadas a la Internet, las cuales pueden estar prestadas para que un atacante pueda comprometer su seguridad y, por lo tanto, molestar a quienes están bajo esa red.

Trudy puede:
- Modificar, insertar o borrar mensajes o contenidos del mensaje.
- Escuchar el canal (sniffing)

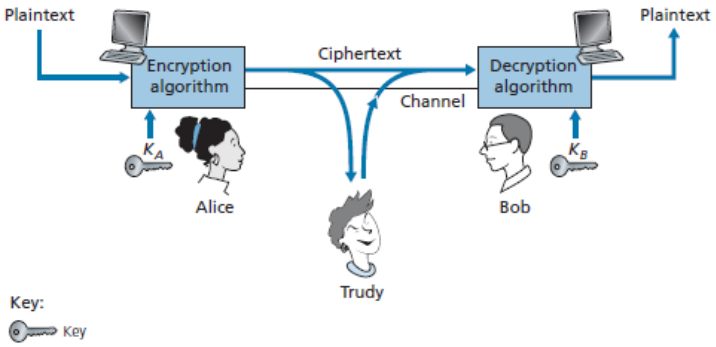


Criptografía

Provee confidencialidad, pero también nos ayudará con la autenticación y la integridad. Permite al emisor "disfrazar" datos para que el intruso no pueda entenderlos. El receptor, claramente, debe poder recuperar los datos originales de los datos "disfrazados".

Supongamos que Alice le quiere mandar un mensaje a Bob.
El mensaje original de Alice es "`¡Bob, comprá pan, por favor!`", el cual es un mensaje de texto plano. Es peligrosísimo mandarse mensajes de texto plano por el canal inseguro, porque Trudy va a saber que Alice lo mandó a comprar pan a Bob.
Alice encripta su mensaje de texto plano con un algoritmo de encriptación, para convertir el mensaje pelado en un texto cifrado, y así consiguiendo que Trudy no lo pueda entender. ¡Pero los algoritmos de encriptación son conocidos, hasta Trudy puede llegar a conocerlo!
Claramente, si todos conocen los algoritmos de encriptación, tiene que haber otra información secreta para evitar que Trudy desencripte los mensajes.

A esta información secreta se la llama clave. Alice provee una clave, K_A, la cual usa como parámetro -junto al mensaje de texto plano (m)- el algoritmo de encriptación, devolviendo el texto cifrado como $K_A(m)$.
A su vez, Bob le va a proveer otra clave, K_B, a su algoritmo de desencriptación para obtener el mensaje de texto plano original haciendo $K_B(K_A(m)) = m$.

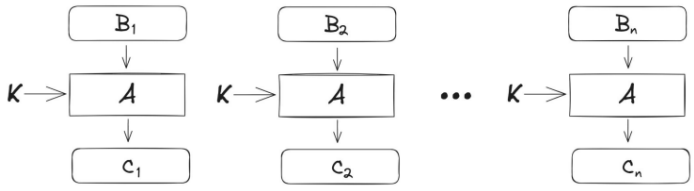


Criptografía de clave simétrica

En un sistema de claves simétrico, Alice y Bob usan la misma clave secreta. En un sistema de claves público, se usa un par de claves: una de ellas la conoce todo el mundo, mientras que la otra la conoce o Alice o Bob, pero ambos no.

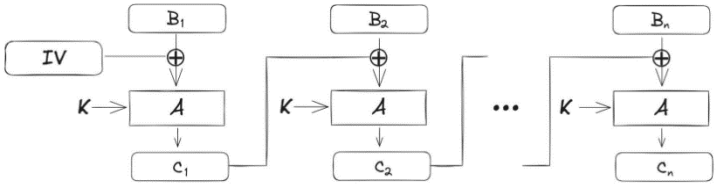
Todos los algoritmos criptográficos consisten en sustituir una cosa por otra. Si se hace un análisis estadístico acerca del uso de las letras en el idioma español (esto es, saber que la "e" es la letra más usada, sigue la "a"...), se facilita descifrar el mensaje, aún más si sabemos qué contenido esperarnos dentro del mensaje. La facilidad de Trudy para descifrar el mensaje depende de qué información él tiene.
- Ataque de texto cifrado: Trudy tiene acceso al texto cifrado que interceptó.
- Ataque de texto plano conocido: cuando conoce alguna información que puede contener el texto cifrado.
- Ataque de texto plano elegido: Trudy puede obtener el texto cifrado de un texto plano cualquiera.

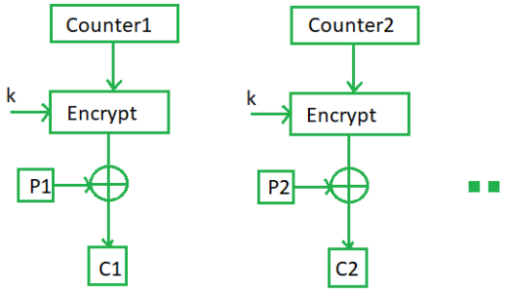
El cifrado en bloques consiste en encriptar un mensaje en bloques de k bits. Si k=64 bits, el mensaje se divide en bloques de 64 bits de largo, con cada bloque encriptado independientemente. Para codificar cada bloque, el cifrado en bloques mapea por sustitución a cada bloque, por lo que hay un output distinto para cada bloque que tome como input. Hay 2^k inputs posibles.
Cada mapeo se puede ver como una clave, por lo que Alice y Bob la pueden saber.
Los cifrados por bloques conocidos son DES, 3DES y AES.



Cifrado por bloques en modo Electronic CodeBook (ECB). Es peligroso porque se pueden adivinar los mensajes dado que dos bloques pueden ser iguales. El atacante puede ver cuáles bloques son iguales y sabrá perfectamente cómo traducirlos.

Cifrado por bloques en modo Cipher Block Chaining (CBC). Se erradica el problema de ECB porque relacionamos los bloques haciendo un XOR entre el bloque cifrado anterior y el bloque actual. IV es el Initialization Vector, que es una secuencia de bits random para arrancar el proceso.





Cifrado por bloques en modo Counter (CTR). Se cifra sucesivamente el valor de una secuencia incremental y se calcula el XOR del valor cifrado obtenido con el bloque correspondiente.

Criptografía de clave pública

La criptografía de clave pública surge porque se necesita que Alice y Bob compartan un "secreto" que es la clave simétrica, pero antes, Alice y Bob se tienen que poner de acuerdo para usar la misma clave. Para ello, se necesita hacer también sobre una comunicación segura.

El uso de las claves públicas consiste en que un interlocutor tenga dos claves: una pública y una privada.
Como siempre, suponemos que Alice se quiere comunicar con Bob. En vez de tener los dos la misma clave, Bob (el receptor) va a tener una clave pública -que todos pueden acceder- y una clave privada -que sólo él la conoce-.
Sea K_B^+ y K_B^- las claves pública y privada de Bob.

Para que Alice se comunique con Bob, ella busca la clave pública de Bob, K_B^+, y luego encripta el mensaje m con un algoritmo de encriptación conocido. Es decir, ella hace $K_B^+(m)$.
Bob recibe el mensaje encriptado de Alice, y usa su clave privada junto a un algoritmo de desencriptación para conseguir el mensaje, es decir, hace $K_B^-(K_B^+(m))$.

Notar que $K_B^-(K_B^+(m)) = m$.

El problema con la criptografía de clave pública es que cualquiera puede mandarle un mensaje a Bob, ya sea Alice o alguien que dice ser Alice.

Un algoritmo que se ocupa de este problema es Rivest Shamir Adleman (RSA), que hace cálculos aritméticos con módulos, ya que los bits que se mandan se pueden ver como números enteros. Muy a cuentagotas, este algoritmo nos va a ayudar con las firmas digitales y también es bastante seguro.

Firmas digitales

Una firma deja en claro que vos estás de acuerdo con los contenidos de un documento. Una firma digital hace lo mismo, y también es verificable y no adulterable.

Supongamos que Bob quiere firmar un documento de manera virtual, siendo el documento el mensaje *m*. Para hacer esto, Bob simplemente usa su clave privada, K_B^-, para computar $K_B^-(m)$, su firma digital.

Supongamos que Alice tiene *m* y $K_B^-(m)$. Quiere verificar que Bob firmó el mensaje y que fue el único que tuvo acceso al mensaje.
Para esto, Alice toma la clave pública de Bob, K_B^+ y se la aplica a la firma digital $K_B^-(m)$. Es decir, hace $K_B^+(K_B^-(m))$. Como output de esta operación, consigue *m*.

Alice está convencida que sólo Bob pudo haber firmado el documento porque sea quien sea que haya firmado el documento, necesitó usar la clave privada K_B^- para hacer la firma $K_B^-(m)$. El único que tiene esa clave privada es Bob.

Certificados de clave pública

Vinculan una clave pública con una identidad. Una autoridad de certificación (CA) genera un certificado haciendo esta vinculación y luego firma este certificado digitalmente.

Autenticación

Alice quiere demostrar su identidad. Para hacer esto, hay protocolos de autenticación.
- Protocolo ap1.0: Alice le puede decir a Bob "Soy Alice", pero Trudy también puede decirle "Soy Alice". Bob no puede verificar esto.
- Protocolo ap2.0: Alice le dice a Bob "Soy Alice" en un datagrama IP con su dirección. Trudy puede armar un datagrama IP con la dirección de Alice diciendo "Soy Alice".
- Protocolo ap3.0: Alice le dice a Bob "Soy Alice" en un datagrama IP con su dirección y una contraseña. Trudy puede sniffear este datagrama y reenviar en un futuro un datagrama como estos exactamente igual.
- Protocolo ap3.1: Sucede lo mismo que el anterior, sólo que la contraseña está encriptada. Trudy puede hacer lo mismo que con ap3.0.
- Protocolo ap4.0: Alice le dice a Bob "Soy Alice", Bob le manda un *nonce* R (valor único), y Alice debe cifrar este número R con una clave compartida entre ellos dos.
- Protocolo ap5.0: Igual al anterior, sólo que Alice encripta este nonce con K_A^-. Cuando a Bob le llega $K_A^-(R)$, le pide a Alice su clave pública para desencriptar y verificar que es Alice. Se presenta la falla de *man-in-the-middle*, donde Trudy estuvo como intermediario todo este tiempo, pero si se pueden verificar los certificados, se erradica por completo.

Transport Layer Security (TLS)

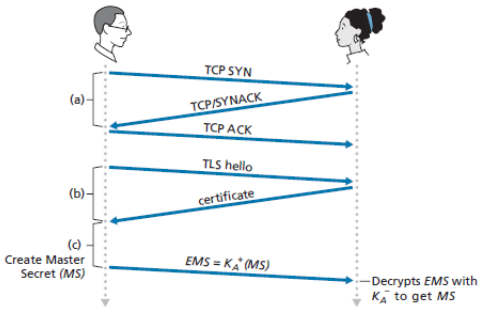
Vamos a ver cómo las técnicas de criptografía ofrecen seguridad, integridad y autenticación para una aplicación en específico.

Se usa mucho en sitios web de comercios, y es necesaria esta medida de seguridad. Si no hubiera confidencialidad, alguien te podría robar información de las tarjetas de crédito; si no hubiera integridad, alguien te puede modificar un pedido; y si no hubiera autenticación, podríamos estar comprando en una página trucha.

Se usa TLS para términos de seguridad en transacciones que corren sobre HTTP. Como TLS corre sobre TCP, se puede usar por cualquier aplicación que corra sobre TCP.

Van a darse tres fases en TLS:

- Handshake. Bob necesita establecer una conexión TCP con Alice, necesita verificar que Alice es realmente Alice y le va a mandar una master key secreta. Cuando se establece la conexión TCP, Bob saluda a Alice, y ella le manda su certificado (el cual está certificado por la CA) con su clave pública. Finalmente, Bob genera un master secret (MS) que es encriptado por la clave pública de Alice.



- Derivación de claves: el master secret MS, ahora compartido entre Alice y Bob, se puede usar como una clave de sesión simétrica. Con este MS, Alice y Bob generan cuatro claves: E_B, M_B, E_A y M_A. E_A es la clave de la encripción de la sesión de los datos que le manda Bob a Alice y M_B es la clave de sesión HMAC para los datos que le manda Bob a Alice.

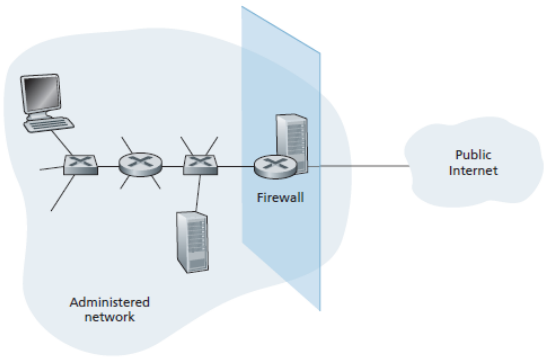
HMAC es el código de autenticación de mensajes hasheado.

- Transferencia de datos: ahora que Alice y Bob comparten las mismas cuatro claves de sesión, se pueden empezar a mandar datos seguros sobre una conexión TCP.

Firewall

Es una combinación de hardware y software que aísla una LAN o subnet del resto de Internet, haciendo pasar a algunos paquetes y bloqueando otros, para que se controle el acceso y se protejan los recursos manejando el tráfico.

Un firewall es un peaje entre la subnet y la Internet, si quiere pasar un paquete de un lado a otro, tiene que pasar por el firewall necesariamente. También, sólo el tráfico autorizado por el firewall tiene permitido pasar, y este es inmune a la penetración.



Los firewalls se pueden clasificar en tres categorías:

- Filtrado sin estado (stateless): el firewall chequea paquete a paquete sus parámetros. Si no hay nada raro en un paquete, lo deja pasar.
- Filtrado con estado (stateful): se fija en todas las conexiones TCP si los paquetes que vengan por ahí pueden ser permitidos o no. El firewall se fija esto en una lista de control de acceso, registra el establecimiento y el fin de las conexiones y puede determinar si los paquetes intercambiados son consistentes
- Application gateway.

TEORÍA DE LA INFORMACIÓN

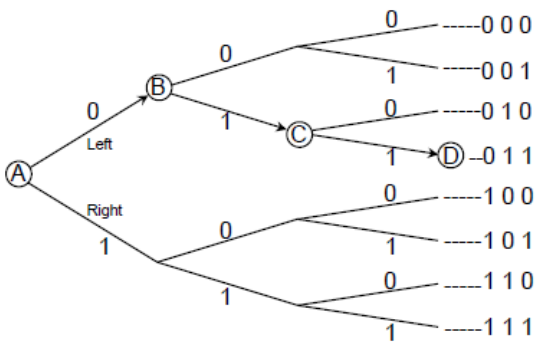
Directamente voy a resumir la bibliografía, ya fue. Lo que resumí es lo que mandó el profesor Marcelo de *James V. Stone – Information Theory: A Tutorial Introduction*.

¿Qué es la información?

El universo está descrito en temas de cantidades físicas como masa y velocidad, pero una cantidad no menos importante es la información. Estamos convencidos que el comportamiento de las computadoras, la evolución, la inteligencia artificial o los cerebros están determinados por cómo procesan la información.

Claude Shannon le puso una definición matemática a la información, y describe cuánta información puede ser comunicada entre dos elementos distintos en cualquier sistema. Esto establece nuestra relación entre cómo se relacionan los ruidos y las señales, y por qué la tasa a la que se comunica la información entre cualquier sistema (animal o hecho por el hombre) tiene un límite definido.
Cuando buscamos una pregunta en Google, los resultados que salen son información útil para nosotros, pero estos resultados que buscamos están entre muchísimas cosas que no nos interesan. Así, notamos la diferencia entre información y datos: la información es la "señal" que nos interesa, y el resto es solamente "ruido".
La habilidad de separar señal de ruido, o sea, extraer información a partir de datos, es importantísima para las telecomunicaciones hoy. No sólo eso, si no que nuestros sentidos también hacen algo parecido cuando hablamos con alguien, ya que nos dedicamos exclusivamente a escuchar a la otra persona, aislando su voz de todo el barullo que puede haber.

La información se mide en bits, y un bit de información te permite elegir entre dos alternativas.
Supongamos que estamos parados en una ruta con varias bifurcaciones (la ruta se divide en dos) en cadena. Si no tenemos información a priori acerca de cómo llegar desde el punto A al punto D.
Si nos dicen que vayamos a la izquierda, nos llegó un bit de información (si sabemos que 0 = "izquierda" y 1 = "derecha"). Cualquier dirección que nos digan para tomar estará representada en un bit.



La decisión tomada en el punto A excluye a la mitad de los ocho posibles destinos a los que podíamos llegar en un principio. De igual manera, por cada decisión que tomemos en cada punto, va a reducir los destinos posibles a la mitad.
Ahora, imaginen que estamos en el punto B cuando nos dieron un bit de información (que, en este caso fue 0 = izquierda). Si nos dan un bit más de información, sea un 1, llegaríamos al punto C. Si nos dan otro bit más de información, un 1, nos vamos al punto D.
Con tres bits en nuestras manos, sabemos que podríamos haber elegido ocho destinos posibles partiendo de A, porque 2 x 2 x 2 = 8, por lo que la cantidad de caminos posibles es

2^cantidad de bits.
Se podrían haber numerado los ocho destinos posibles con números decimales que vayan del 0 al 7 o con su número binario correspondiente. La representación binaria representa una ventaja en este caso, porque nos dice las instrucciones (izquierda/derecha) necesarias para llegar a nuestro destino. Esto aplica a cualquier problema que incluya un dilema.
Aparte, podemos calcular cuántas instrucciones se necesitan para llegar a ocho destinos posibles mediante un logaritmo base 2: $\log_2(8) = 3$ bits de información. Si tenemos n bits de información, podés elegir entre $m = 2^n$ alternativas equiprobables, y si querés elegir entre m alternativas posibles, necesitás $n = \log_2 m$ bits de información.

Si bien la palabra *bit* viene de *binary digit*, no son la misma cosa. Un dígito binario puede ser un 0 o un 1, pero no es información de por sí. En contraste, un bit es una cantidad de información definida, que puede representarse como un 0/1 o un Sí/No.

Denle bola a esto de los bits y dígitos binarios. No son lo mismo, y los usé bastante quisquillosamente. Atención absoluta señores.

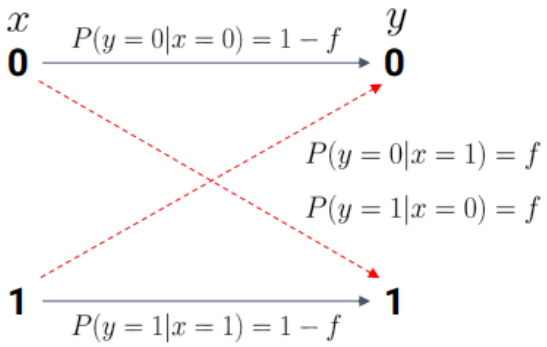
Un bit es una unidad de información, pero no es la cantidad mínima de información que puede existir: puede darse el caso que, si estamos en el punto A, ya sabemos que hay que ir por la izquierda. Por más que nos digan "hay que girar a la izquierda" dándonos un dígito binario 0, ya sabíamos esto, entonces no ganamos información.
Pero puede darse el caso en el que sepamos que hay una probabilidad del 71% de que haya que girar a la izquierda (en contraste con el 100% de probabilidad que teníamos en el caso anterior). Esto quiere decir que tenemos algo de información no más. Si tuviéramos un bit solo de información ya sabríamos qué camino tomar, pero ahora tenemos menos de un bit de información. Si nos muestran un dígito binario 0, como ya tenemos algo de información, ganamos menos de un bit de información, es decir, casi un tercio de bit (29%).
Acordarse de que un dígito binario puede adoptar un valor de dos valores posibles (0 ó 1), mientras que un bit es una cantidad de información que puede adoptar cualquier valor entre 0 y 1.
Dígito binario: [0; 1] para todo número entero
Bit: [0, 1] para todo número real

Podría tranquilamente explicar los ejemplos de telegrafía con código Morse, imágenes binarias e imágenes de niveles grises. No creo que aporten tanto en cuanto al contenido que vemos en TD IV, pero los voy a leer de todas maneras porque no lo escribió James porque estaba al pedo.

Acá empieza el resumen de *Thomas M. Cover, Joy A. Thomas – Information Theory*. Debe ser la bibliografía más chota que leí en mi vida.

La información, cuando viaja por un medio, puede verse afectada por el ruido. Una comunicación ruidosa puede ser la línea telefónica.

Supongamos que un canal transmite correctamente bit a bit con una probabilidad (1-f), siendo f la probabilidad de que falle.



En la columna izquierda tenemos un emisor, x, que le manda información al receptor, y.
La probabilidad de que a y le llegue un 0, P(y=0), si es que x le mandó un 0, P(x=0) es P(y=0 | x=0) = 1-f. Esto quiere decir que se transmitió correctamente el bit por el canal.
Si x le transmite un 0 a y, y por alguna razón a y le llegó un 1, quiere decir que hubo un error: P(y=1 | x=0) = f.
Esto sigue valiendo si x transmite un 1 y a y le puede llegar un 0 (incorrecto) o un 1 (correcto).

Esa línea que conecta x con y es un canal binario simétrico (BSC), el cual está prestado a que el ruido cause que las cosas lleguen corrompidas al otro lado del canal. Los bits recibidos no muestran en dónde ocurrió el error. Hay dos soluciones a esto: una es una solución física, ya que si mejoramos los componentes del canal el ruido va a ser menor; y la otra es la solución lógica, que propone aceptar este canal ruidoso, pero adoptando técnicas de detección y corrección de errores con redundancia. Este es el problema central de la comunicación.

Recordar que un código es un sistema de símbolos y reglas utilizados para representar información, mientras que un mensaje codificado es un mensaje transformado mediante un proceso de codificación.

Un mensaje se puede codificar con una función llamada tabla de "look-up".
La distancia de Hamming de un código es el número de posiciones en las que dos palabras de código (dos secuencias de bits) son distintas. Por ejemplo: la distancia de Hamming entre 1001 y 1111 es dos, porque basta con cambiar de valores a dos bits.
La distancia mínima (d_{min}) es la mínima distancia de Hamming entre cualquier par de palabras distintas en un código. Puede detectar hasta d_{min}-1 errores, y corregir hasta (d_{min}-1)/2 errores.

Podrías llegar a insinuar que, para que la probabilidad de que se encuentre un error sea la mínima posible, se podría mandar el mensaje varias veces. Y tenés razón: se incorpora un esquema de repetición llamado R3, porque se manda cada bit tres veces.
- Primero, el receptor utiliza una regla de mayoría para decidir el valor del bit original: si al menos dos de los tres bits que recibe son iguales, asume que los dos bits iguales es el bit correcto que quiso mandar el emisor.
- Luego, sabemos que la probabilidad de error (sin redundancia), era como dije al principio, la probabilidad de que un bit transmitido sea recibido incorrectamente, denominado como f.
- Por lo tanto, para un esquema R3, queremos saber la probabilidad de que la mayoría de los tres bits recibidos sean incorrectos. Sólo hay dos casos: si los tres bits son incorrectos o si dos de los tres son incorrectos. Entonces nos deja con una probabilidad de que los tres estén

incorrectos de f * f * f = f^3, y la probabilidad de que dos estén incorrectos es f * f * (1-f) = f^2(1-f). Con este esquema es muy fácil (de)codificar mensajes y podríamos reducir la probabilidad de error cuanto queramos si le agregamos muchos más bits de redundancia, pero agregar bits de redundancia implica disminuir la tasa de transmisión en el canal, ya que tiene más trabajo para mandar por cada mensaje.

Ahora, queremos tener la probabilidad de error más baja posible, pero teniendo una tasa de transmisión lo más alta posible. Para esto, queremos agregar redundancia por bloques de datos en lugar de codificar un bit a la vez. El código de Hamming hace esto: cada 4 bits de mensaje codificado, se le agregan 3 bits de redundancia, por lo que termina siendo 7 bits transmitidos.

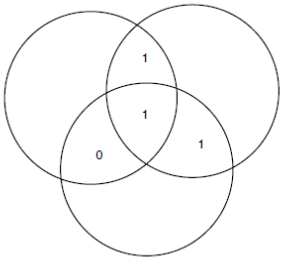


FIGURE 7.10. Venn diagram with information bits.

Si queremos transmitir la secuencia de información 1101, agregamos estos bits de información en las intersecciones en el diagrama de Venn.

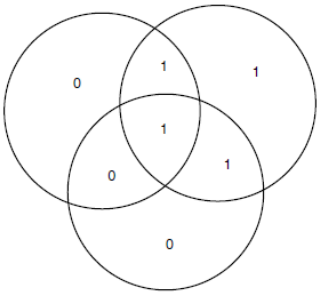


FIGURE 7.11. Venn diagram with information bits and parity bits with even parity for each circle.

Cuando ponemos un bit de paridad en cada región que falta, el círculo es par (hay una cantidad par de unos en el círculo).

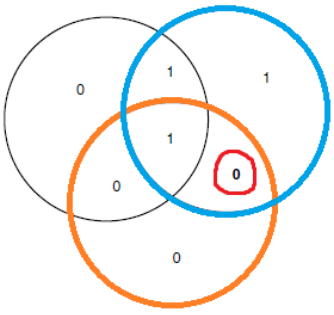
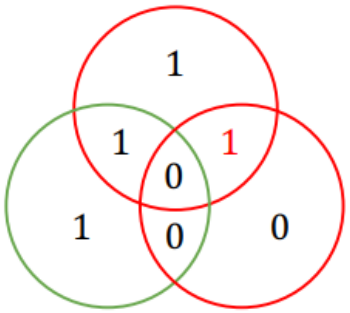


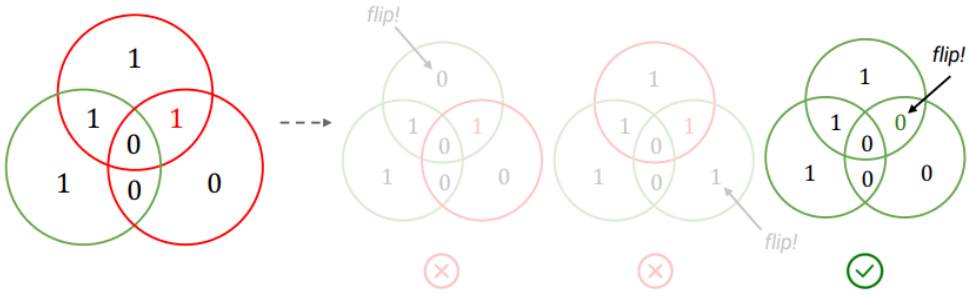
FIGURE 7.12. Venn diagram with one of the information bits changed.

Supongamos que se dio un error en algún bit. Como se puede ver, se violó la paridad de dos círculos: el círculo anaranjado y el azul. El único error posible está en el bit que se marcó con

rojo. Esta es una manera en la cual se puede hallar el error en el código de Hamming.



Otra manera posible es marcar los círculos sin paridad par. Como los círculos rojos no tienen paridad par, sólo basta con cambiar el 1 para que tenga sentido, y así corregir el error.



Entropía

Es la medida de las "sorpresas" promedio en una variable aleatoria. Es el número de bits promedio que se necesitan para describir una variable aleatoria. La entropía de una variable aleatoria X con una función de masa p(x) es:

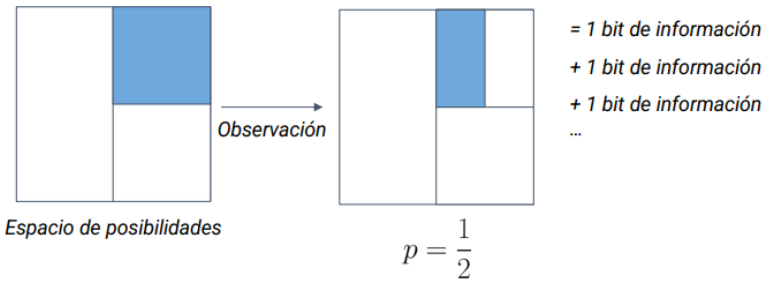
$$H(X) = \sum_{i=1}^{m} p(x_i) \log_2 \frac{1}{p(x_i)} = -\sum_{i=1}^{m} p(x_i) \log_2 p(x_i) \text{ bits}$$

Por ejemplo, consideremos una variable aleatoria *X* con distribución uniforme de 32 resultados posibles. Para identificar los 32 resultados posibles, necesitamos una secuencia que tome 32 valores distintos. Con una secuencia de 5 bits es suficiente. La entropía de esta variable aleatoria es:

$$H(X) = -\sum_{i=1}^{32} p(i) \log p(i) = -\sum_{i=1}^{32} \frac{1}{32} \log \frac{1}{32} = \log 32 = 5 \text{ bits},$$

que es el número de bits necesitado para describir la variable *X*.

La entropía es la cantidad mínima de preguntas binarias (asumiendo que trabajamos siempre con logaritmos de base 2) para identificar el valor de una variable aleatoria discreta.



El valor de la entropía depende sólo de la distribución de probabilidades.

Primer teorema de Shannon

Si queremos transmitir los resultados de alguna distribución, en promedio vamos a necesitar tantos símbolos como la entropía de esa distribución. Esto quiere decir, que si queremos transmitir que cuando tiramos una moneda al aire y salió cara, vamos a necesitar un solo bit, ya que $\log_2(2) = 1$ bit.
La entropía es el límite de cuánto podemos llegar a comprimir nuestra información.

Técnicamente, dice que es posible codificar una fuente de información con entropía *H* y un canal de capacidad *C* de manera que se pueda transmitir a más o menos *C*/*H*-e símbolos por segundo.
La tasa máxima de transmisión de un canal se puede calcular como R = *C*/*H*. Si tenemos un canal de capacidad *C* de 2 bits por segundo, y tenemos una entropía *H* de 2 bits por símbolo, entonces la capacidad máxima R del canal es 1 símbolo por segundo. Éste es un código eficiente.

Si tenemos un dado de seis caras, con capacidad del canal C de 1 bit por segundo y una entropía H de 2,58 bits por símbolo.
Tasa máxima de transmisión:

$$R = \frac{C}{H} = \frac{1}{2.58}\ \frac{\text{bits/s}}{\text{bits/símbolo}}$$
$$= 0.387\ \text{símbolos/s}$$

Eficiencia:

$$\frac{H(S)}{L(X)} = \frac{2.58}{3} = 0.86\ \text{bits/dígito binario}$$

69

Tasa efectiva:

$$R = \frac{C}{H} = \frac{0.86}{2.58}\ \frac{\text{bits/s}}{\text{bits/símbolo}}$$
$$= 0.333\ \text{símbolos/s}$$

La compresión sirve para que los símbolos más frecuentes (en probabilidad) no ocupen tanto espacio como los pocos frecuentes. Si tiramos dos dados, la probabilidad de que la suma de los dos resultados dé 2 ó 12 no es la misma que la suma dé 7.
El primer caso implica que las dos veces sean 1 o 6. En el segundo tenemos más combinaciones posibles.
Asumimos que son variables iid.

La información de Shannon es una forma de expresar las "sorpresas" en cantidades, en función de la probabilidad de una variable aleatoria. Nos da más información saber que

$$h(x) = \log_2 \frac{1}{p(x)}$$
$$= -\log_2 p(x)$$

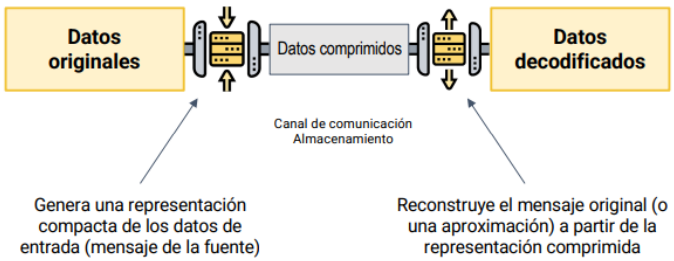
ocurrió un evento improbable, que conocer que sucedió un evento común.
Nos dará un valor más alto (más información de Shannon) si la probabilidad de X = x es baja.
Hagan las cuentas ustedes...

Una secuencia con letras aleatorias va a tener mucha entropía porque se asumen que las apariciones de las letras (las variables) son independientes unas de otras, pero si sabemos que un mensaje va a tener sólo la letra "a" repetida va a tener entropía muy baja, porque no tienen ocurrencias independientes.
De igual manera, un mensaje con contenido que no conocemos va a tener mucha información de Shannon, pero un mensaje con un contenido predecible va a tener baja información de Shannon.
Ambas cosas relacionan la incertidumbre presente en la información y usan la misma fórmula matemática.

Por otro lado, la información de Shannon cuantifica las "sorpresas", mientras que la entropía mide la incertidumbre promedio (no saber si será sorpresa o no). También, la información de Shannon se usa para medir la eficiencia de codificación y transmisión de datos, mientras que la entropía sirve para comprimir datos y ver la capacidad de un canal.

Compresión

Consiste en asignarles códigos más cortos a los símbolos más frecuentes, y códigos más largos a los menos frecuentes (hace una compactación del mensaje y se espera que lo reconstruya el receptor adecuadamente). Esto requiere conocer las probabilidades de que se den esos símbolos. Para esto, se asumen eventos independientes.



Hay dos tipos de compresión:
- Sin pérdida. Se mantiene la integridad de la información - cuando se descomprime el mensaje se consigue el mensaje original.

$$H(S) \leq L(X)$$

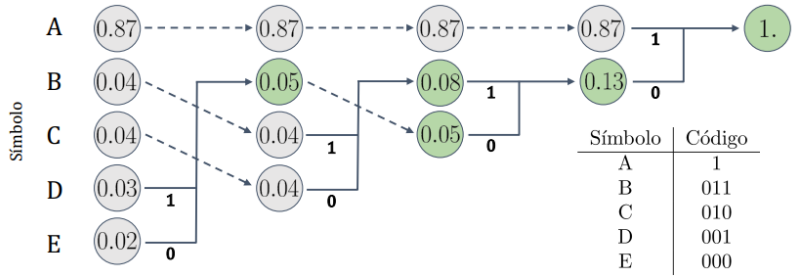
- Con pérdida. No se mantiene la integridad del mensaje - cuando se descomprime se obtiene una aproximación del mensaje. No tienen a la entropía como límite.

$$L(X) < H(S)$$

Código de Huffman

Consiste en encontrar los dos símbolos menos probables y luego combinarlos para hacer un símbolo compuesto imaginario. Estos dos símbolos menos comunes se reemplazan por el nuevo símbolo compuesto.
Este proceso se repite hasta que sólo quede un símbolo imaginario, compuesto por otros símbolos imaginarios, compuestos por otros símbolos imaginarios...
Cada vez que se combinan dos símbolos, los dos caminos entre esos dos símbolos y el nuevo

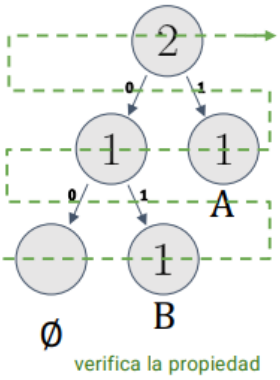
símbolo se etiquetan con un 1 para cada camino superior y con un 0 para cada camino inferior, formando un árbol que va de izquierda a derecha.
Cuando se completa este árbol, el código de cada símbolo se obtiene pegando los dígitos binarios (como el primer ejemplo para llegar de A hacia D)

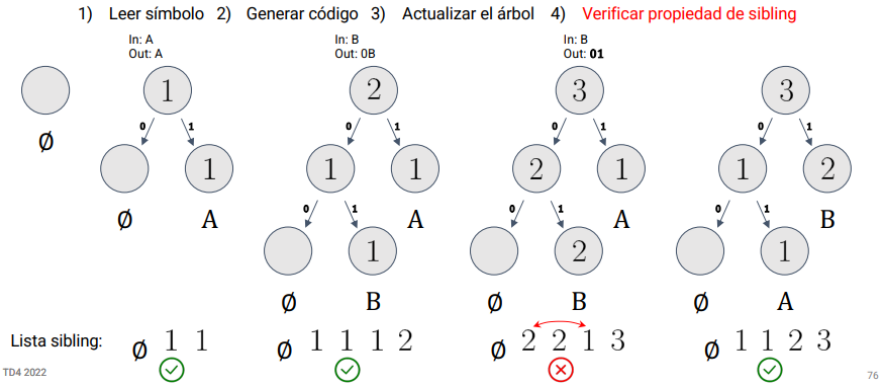


Primero, analizar cada probabilidad por separado de ocurrencia de los símbolos en el mensaje. Nos da como resultado las hojas.
Segundo, se construye el árbol hasta llegar a la raíz.
Finalmente, se ven las rutas que se tomaron para conseguir la codificación. Además, se le agregan la cabecera con las probabilidades de que aparezca cada símbolo para que el decodificador del receptor pueda reconstruir el mensaje.

Código de Huffman Adaptativo (FGK)

No se conocen las probabilidades de los símbolos. Se comienza con un árbol vacío y cada nodo del árbol almacena un valor entero de frecuencia de aparición. Con esto, las hojas tendrán un contador de símbolos observados hasta el momento y su frecuente. Los nodos internos contienen la suma de las frecuencias de sus hijos.
Se usa un nodo especial, el nodo "cero" Ø para identificar el punto de inserción. Los árboles de Huffman cumplen la misma prioridad que la de los árboles de búsqueda de TD III. A la izquierda va siempre un número menor (propiedad de sibling).

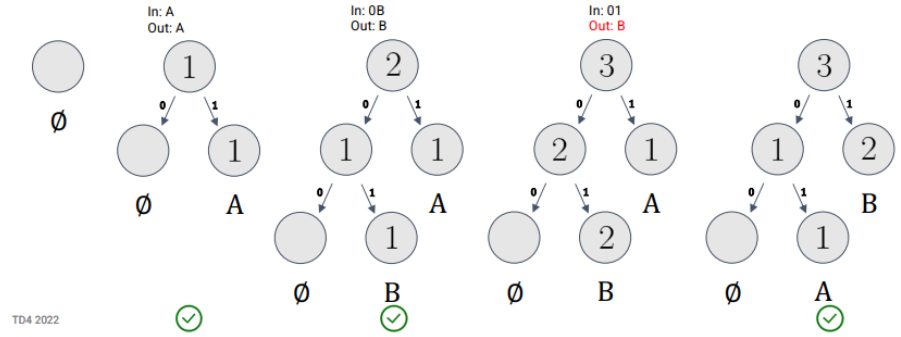




En el paso 1) se lee el símbolo "A", como "A" no aparece en el árbol, se transmite el código Ø más el símbolo literal "A" porque no hay ninguna ocurrencia previa de este símbolo. Se cumple la propiedad de sibling.
En el paso 2) se lee el símbolo "B", como no apareció nunca en el árbol, se transmite el código Ø más el símbolo literal "B" (porque no hay ocurrencias previas). Aparte, se divide el nodo que contenía al Ø, dejándolo en 1 y que de él salgan dos ramas: una con el Ø y la otra con la apariencia de B: 1. El código generado para "B" es 0B. Se cumple la propiedad de sibling.

En el paso 3) se lee el símbolo "B", como apareció en el paso anterior, se transmite el código asociado (0B) y se incrementa la frecuencia del nodo y se propaga todo hacia arriba (ahora el código de "B" es 01). No se cumple la propiedad de sibling, por lo que en el paso 4) se reemplazan las ramas para que queden bien acomodadas.
Finalmente, conseguimos la codificación total de A0B01.

El descompresor es simétrico: se recibe la codificación y se recrea el mismo árbol.



Métodos con diccionarios

Se reemplazan las secuencias de los símbolos que se repiten por un índice a alguna ocurrencia anterior, ya que un índice ocupa menos que una palabra. Comprimir por este método es más caro que descomprimirlo.

Run Length Encoding (RLE)

Reemplaza secuencias de símbolos consecutivos iguales por un par ordenado (símbolo, repeticiones) (*S*, *R*).
Si tenemos "AAAAFJGGGGF", 12 bytes, se puede comprimir como A4F1J1G41F (10 bytes).
Este algoritmo no nos conviene usarlo cuando tenemos un texto donde casi no aparezcan repeticiones: "JUANIGNACIOELOSEGUI", 19 bytes, terminaría quedando J1U1A1N1I1G1N1A1C1O1E1L1O1S1E1G1U1I1, con 36 bytes.

Compresión Lempel & Ziv (LZ)

Usa dos buffers, uno sirve como ventana de búsqueda con los símbolos a codificar, y el otro tiene la secuencia ya procesada.
El método LZW es un método universal de compresión de datos con un diccionario precargado (como 256 ASCII). Para todos los símbolos:
- Se busca la frase de mayor coincidencia en el diccionario y codificar.
- Se agrega al diccionario una nueva entrada: frase codificada + siguiente símbolo no coincidente.

símbolo/s	código	diccionario	
diccionario precargado		a	0
		b	1
a	0	aa	2
a	0	ab	3
b	1	ba	4
ab	3	aba	5
aba	5	abaa	6
aa	2		

Por ejemplo: tenemos la secuencia "aababababaaa", con un diccionario precargado de a = 0, y b = 1.
Como "aa" aparece dos veces seguidas, no es necesario codificar esto como 00, si no que se agrega una entrada nueva al diccionario con "aa" y símbolo 2.
"b" aparece una sola vez sin repetirse, por lo que significa un 1.
Y así...

Si queremos descomprimir: para todos los símbolos se busca la entrada en el diccionario, se decodifica la frase asociada y se avanza, luego, se agrega al diccionario una nueva entrada: decodificación anterior + primer símbolo de la decodificación actual.

código	símbolo/s	diccionario	
diccionario precargado		0	a
		1	b
0	a	2	aa
0	a	3	ab
1	b	4	ba
3	ab	5	aba
5	aba	6	abaa
2	aa		

Si el mensaje que recibió el decodificador es 001352, con un diccionario precargado de 0 = "a" y 1 = "b".
Como el 0 aparece dos veces consecutivas, habrá que integrarlo al diccionario y elegimos decodificarlo como "aa".
El 1 aparece una sola vez, ya sabemos que 1 = "b".
El 3 aparece una sola vez consecutiva, y decidimos decodificarlo como "ab".
El 5 aparece, por lo tanto se usa la decodificación anterior ("ab") y se le agrega el primer símbolo, que es "a", por lo que su codificación se agrega al diccionario como 5 = "aba".