Sistemas Operativos - Apuntes para final

Gianfranco Zamboni

19 de octubre de 2023

Índice

1. Introducción

1.1. La semilla del internet

El telégrafo fue el antecesor del teléfono, un primer acercamiento a la comunicación de mensajes vía una codificación. Desde fines de siglo XIX hasta segu nda mitad del siglo XX, aparecen las centrales de **conmutación de circuitos** (centrales telefónicas). A estas centrales llegaban señales (cables) correspondientes a todas las casas que participacen en el sistema de teléfonos. Las operadoras conectaban dos circuitos en sus tableros para cerrar el circuito y permitir la comunicación entre las dos partes involucradas. Sin embargo, este tipo de comunicación tenía una gran falla: Si una central de conmutación de circuitos dejaba de estar disponible por algún motivo de fuerza mayor, todas las personas pertenecientes a esa zona se verían incomunicadas.

A fines de los 50 se empieza a desarrollar la **conmutación de paquetes** buscando resolver este tema, es decir se busca una **red más tolerante a fallas**, **más flexible** a la hora de conectar dos puntos distantes y que **escale más facilmente** ante un incremento en el acceso a la comunicación.

La nueva red propuesta es una red descentralizada con mútiples caminos entre dos puntos que divide los paquetes en fragmentos que pueden llegar a destino a través de distintos caminos.

El modelo OSI

En 1983 aparece una publicación de ISO para establecer un estándar que especifique la estructura de una arquitectura de red, que uniformice la forma de construir las redes de cominucación: el modelo OSI-ISO (Open Systems Interconnection).

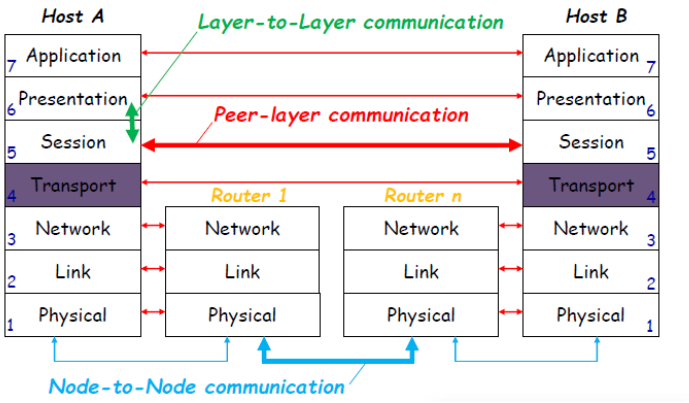

Figura 1: Modelo OSI

Este modelo está dividido en 7 capas, cada una de las cuales tiene una función definida que permitirán la comunicación coherente entre dos sistemas remotos.

1. La capa **física (Physical)** se encarga de enviar raw bits a través de los medios físicos disponibles en la red.

2. La capa de **enlace (Link)** se encarga de detectar errores en la transmisión y corregirlos, si es posible.

3. La capa de **red (Network)** se encarga de resolver problemas de congestión dentro de la red, que paquetes se aceptan y la ruta que deben tomar los paquetes que se envían por la misma.

4. La capa de **transporte (Transport)** se encarga de tomar la información provista por la capa de arriba, pasarla a la capa de red separada en pedazos más chicos (**chunks**) y se asegura que todas las partes lleguen a destino correctamente.

   Esta es la primer capa **end-to-end**, es decir que entabla una "conversación" entre la máquina emisora (**Source**) y la destinataria (**Destination**). Las capas anteriores, usan protocolos de comunicación nodo a nodo, es decir, entre una máquina y su vecino inmediato y no entre el source y el destination que podrían estar separados entre sí por varios nodos.

5. La capa de **sesión (Session)** permite establecer sesiones entre dos máquinas distintas. Estas sesiones permiten sincronizar el pasaje de información entre ambas máquinas, deciden de quien es el turno para enviar información y evitar que ambas máquinas realizen operaciones críticas de manera simultanea.

6. La capa de **presentación (Presentation)** procesa la información recibida, la estructura y la codifica de la manera necesaria para que pueda ser usada por la máquina.

7. La capa de **aplicaciones (Application)** contiene los protocolos necesarios para que los usuarios puedan ver y leer la información.

Además de estas funcionalidades, cada capa ofrece una interfaz que le permite comunicarse con las capas vecinas para hacer el pasaje de los datos entre ellas y asumen que el host en el otro extremo de la comunicación tendrá una arquitectura similar y podrá interpretar los mensajes de cada una de ellas.

1.2. Modelo TCP/IP

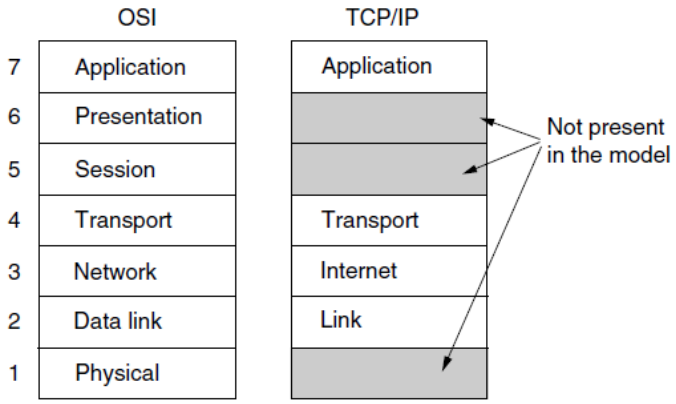

Figura 2: Modelo TCP/IP

Fue diseñado con el objetivo de mantener las conexiones intactas mientras ambos puntos finales de conexión estén funcionando, incluso si alguna de las máquinas o líneas entre ellos fuese dado de baja.

1. La capa de **enlace (Link)** describe que deben cumplir los enlaces (cable de ethernet, lineas telefónicas, etc) para poder usados como medios de trasporte para este tipo de conexión.

2. La capa de **intrared (internet)** permite al host injectar paquetes en cualquier red y se encarga de hacerlos llegar a destino. Esto se suele hacer de manera independiente para cada paquete, es decir que dos paquetes que pertenezcan al mismo mensaje pueden no seguir el mismo camino e incluso podrían llegar a destino en distinto orden. En el último caso corresponde a la capas superiores reodenarlos para que puedan ser procesados, si es necesario.

3. La capa de **transporte (Transport)** debe ser diseñada para permitir que dos entidades de la red puedan mantener una conversación. Aquí se definieron dos protocolos:

   - El **Transmission Control Protocol (TCP)** que permite enviar sin errores un stream de bytes desde una máquina a otra en la red.
   - El **User Datagram Protocol (UDP)** que permite evitar todo el flujo de conexión TCP y crear el suyo propio. En general, se usa en aplicaciones que requieren una respuesta más rápida que precisa.

4. La capa de **Aplicación (Application)** que se incluye la sesiones y funciones necesarias para codificar y procesar los paquetes enviados y recibidos. Entre los protocolos usados en esta capa se encuentran: TELNET, FTP y SMTP.

Parte I
Nível Físico

2. Teoría de la información

2.1. Información

Sea E un suceso que puede presentarse con probabilidad $P(E)$. Cuando E tiene lugar, decimos que hemos recibido

$$I(E) = \log \frac{1}{P(E)}$$

unidades de información.

Si introducimos el logaritmo de base 2, la unidad se denomina *bit*. Notemos, también, que si $P(E) = \frac{1}{2}$, $I(E) = 1$ bit. Es decir, un bit es la cantidad de información obtenida al especificar una de dos posbiles alternativas igualmente probables.

2.1.1. Fuentes de Memoria Nula

Son fuentes de información que emiten una secuencia de símbolos pertenecientes a un alfabeto finito y fijo $S = \{s_1, \ldots, s_n\}$ de manera estadísticamente independientes. Estas fuentes pueden describirse mediante el alfabeto S y las probabilidades con que los simbolos se presentan: $P(s_1), \ldots, P(s_n)$.

La información que aporta cada símbolo de la fuente

$$I(s_i) = \log_2 \frac{1}{P(s_i)} \text{ bits}$$

2.1.2. Entropía

Dada una fuente de memoria nula S con alfabeto $s_1, \ldots, s_n$, la entropía $H(S)$ es la cantidad media de información por símbolo de la fuente, es decir:

$$H(S) = \sum_{i=1}^{n} P(s_i) I(s_i) = \sum_{i=1}^{n} P(s_i) \log_2 \frac{1}{P(s_i)} = -\sum_{i=1}^{n} P(s_i) \log_2 P(s_i) \text{ bits}$$

Podemos interpretar la entropía como el **valor medio ponderado de la cantidad de información** del conjunto de mensajes posibles, como una medida de la **incertidumbre prob-medio (grado de incerteza)**acerca de una variable aleatoria o la **cantidad de información** obtenida al observar la aparición de cada nuevo símbolo.

Propiedades:

- La entropía es no negativa y se anula si y solo si la probabilidad de uno de sus símbolos es 1 y la del resto es 0.

- La entropía máxima (**mayor incertidubme del mensaje**) se logra cuando todos los símbolos que puede ser emitidos por la fuente son equiprobables.
- Si hay n símbolos equiprobables $P(s) = \frac{1}{n}$ se cumple:

$$H(S) = -\sum_{s} P(S)\log_2 P(S) = -n(\frac{1}{n}\log_2\frac{1}{n}) = -(\log_2 1 - log_2 n) = \log_2 n$$

2.1.3. Extensión de memoria nula

Si tenemos una fuente de memoria nula S, con un alfabeto $\{s_1, \ldots, s_q\}$, podemos agrupar las salidas en paquetes de n símbolos. Tendremos, pues, qn secuencias de salidas distintas.

Formalmente, sea S una fuente de información de memoria nula con un alfabeto $\{s_1, \ldots, s_q\}$. Sea P_i la probabilidad correspondiente a s_i. Entonces, la extensión de orden n de S (S^n) es una fuente de memoria nula de qn símbolos $\{\sigma_1, \ldots, \sigma_{qn}\}$ donde cada σ_i corresponde a una secuencia de simbolos de S de longitud n. La probabilidad de σ_i, P_{σ_i} es la probabilidad de la secuencia correspondiente. Es decir, si $\sigma = s_{i_1} \ldots s_{i_k}$ entonces $P(\sigma) = P_{i_1} \ldots P_{i_k}$.

Puesto que un símbolo de S^n corresponde a n símbolos de S, es de suponer que la entropía por símbolo de S^n sea n veces mayor que la de S, osea:

$$H(S^n) = nH(S)$$

2.2. Codificación

Sea $S = \{s_1, \ldots, s_q\}$ el conjunto de símbolos de un alfabeto dado. Se define un **código** como la correspondencia de todas las secuencias posibles de símbolos de S a secuencias de símbolos de algún otro alfabeto $X = \{x_1, \ldots x_r\}$. S recibe el nombre de **alfabeto fuente** y X el de **alfabeto código**.

Cuando codificamos un alfabeto fuente, buscamos lograr una representación eficiente de la información mediante la eliminación de la redundancia.q

Código bloque: Es aquel que asigna cada uno de los símbolos del alfabeto fuente S a una secuencia fija de símbolos del alfabeto código X. Esas secuencias fijas (secuencias de x_i) reciben el nombre de palabras código. Denominaremos X_i, a la palabra código que corresponde al símbolo s_i.

Código no singular: Es un código en el que todas sus palabras son distintas.

Extensión de orden n: La extensión C^n de orden n de un código bloque $C : S \to X^*$, es el código bloque que hace corresponder las secuencias de símbolos de S con las secuencias de las palabras código formadas por $C(s_i)$. Es decir: Si $s_i \ldots s_k \in S^*$, entonces $C^n(s_i \ldots s_k) = C(s_i) \ldots C(s_k)$.

Código univócamente decodificable: Es aquel en el cual ninguna tira de símbolos del alfabeto código admite más de una única decodificación. Dicho de otra forma, un código bloque se dice univocamente decodificable si, y solamente si, su extensión de orden n es no singular para, cualquier valor finito de n.

Código instantaneo: Un código unívocamente decodificable se denomina instantáneo cuando es posible decodificar las palabras de una secuencia sin precisar el conocimiento de los símbolos que las suceden.

Préfijo de una palabra: Sea $X = x_1 \dots x_m$ una palabra de un código. Se denomina prefijo de esta palabra a la secuencia de símbolos $x_1 \dots x_j$, donde $j \leq m$.

La condición necesaria y suficiente para que un código sea instantáneo es que ninguna palabra del código coincida con el prefijo de otra.

Inecuación de Kraft: Dado un alfabeto $S = \{s_1, \dots, s_n\}$ y un alfabeto de código $X = \{x_1, \dots, x_m\}$, es condición necesaria y suficiente, para exista un código instantáneo con palabras de longitud $l_1, \dots, l_n$, que se cumpla la siguiente inecuación:

$$\sum_{i=1}^{n} |X|^{-l_1} \leq 1$$

2.2.1. Codificación óptima

Buscamos codificar un alfabeto S de tal forma que máximizar la relación entre la entropía $H(S)$ y la longitud media del código L. Sea l_i la longitud de la palabra que codifica al símbolo s_i de la fuente, p_i la probabilidad de aparicion de s_i y r la cantidad de símbolos diferentes en el alfabeto del código entonces:

- $L = \sum p_i L_i$ es la longitud medía del código.
- $\log r$ es la cantidad promedio máxima de información de un símbolo del código.
- $h = \frac{H(S)}{L \log r}$ es la eficencia del código.

La máxima eficencia se logra cuando $h = 1$. En general, esto sucede cuando se asigna las palabras de código más cortas a los símbolos de fuente más probables. También se puede deducir:

$$1 \geq h \geq \frac{H(S)}{L \log r} \Rightarrow L \log r \geq H(S)$$

Primer teorema de Shannon (Teorema de la codificación sin ruido): Sea S una fuente de memoria no nula, y S^n la extensión de orden n de S. Sea $C : S^n \to X^n$ un código y L_n la longitud media de los códigos correspondientes a los símbolos de S^n:

$$L_n = \sum_{\sigma \in S^n} C(\sigma) P(\sigma)$$

Entones vale:

$$H(S) \leq \frac{L_n}{n} \leq H(S) + \frac{1}{n}$$

Esto nos dice que el número medio de símbolos de C correspendientes a un símbolo de la fuente puede hacerse tan pequeño (pero no inferior) como la entropía de la fuente. El precio que se paga por la disminución de L_n es un aumento en la complejidad de la codificación debido al gran número de símbolos de la fuente que hay que manejar.

En particular:

$$H(S) = \sum_{\sigma \in S^n} C(\sigma)P(\sigma)$$

Codificador óptimo: Es un codificador que usa la menor cantidad posible de bits para codificar un mensaje, es decir: Un codificador se dice óptimo si no existe ningún código para la misma fuente con menor longitud media.

Sea $s_i \in S$, entonces la cantidad de bits necestarios para representarlo en un codificador óptimo es $\lceil \frac{1}{P(s)} \rceil$ y la entropía de

$$H(X) = \sum_{s \in S} P(s) \log_2 \left(\frac{1}{P(s)} \right)$$

Codificación de Huffman: Es una forma de definir códigos óptimos asumiendo que se conoce la probabilidad de ocurrencia de los símbolos, que la codificación es símbolo por símbolo y la probabilidad de ocurrencia de cada símbolo es independiente.

Dado un mensaje M:

1. Se extrae del mismo la frecuencia de cada símbolo.
2. Se ordenan los símbolos en arbol dependiendo de la frecuencia de cada uno. Mientras más cerca de la raíz, más frecuente es el símbolo.
3. El código de un símbolo será entonces el camino de la raíz al nodo donde está ubicado (utilizando ceros cuando se toma la rama izquierda y 1 cuando se toma la rama derecha).

Así, los símbolos más frecuentes tendrán los códigos más cortos.

3. Señales

3.1. Fundamentos de las señales

Las señales que se envían por el canal físico para comunicar dos extremos de un canal son **ondas electromagnéticas** que se propagan a través del canal a una cierta velocidad determinada por el tipo de canal que estemos usando.

Podemos definir todas las señales con una función períodica $f(t)$ llamada **frecuencia**. Esto significa que $f(t) = f(t+T)$ para alguna constante T. Al minimo valor positivo mayor que cero de T que cumple esto, lo llamamos **período fundamental**. f se mide en Hertz o ciclos por segundo y T en segundos.

En base a a la frecuncia y el período de una onda definimos:

- **Amplitud:** Indica la cantidad de cambios en la presión del aire. Se mide en decibeles (db o volts). Por decirlo de otra forma, la amplitud es la distancia entre el eje horizontal y el punto más alto del pico de la onda, o el punto más bajo de la depresión de la onda.
- **Frecuencia angular:** $\omega = 2\pi f$ radianes por segundos.
- **Fase** ϕ: Compara el tiempo entre dos ondas y se mide en grados, de 0 a 360. Cuando dos ondas comienzan al mismo tiempo, se dice que están en fase o alineadas en fase. Cuando una onda se encuentra ligeramente retrasada en comparación con otra onda, se dice que las ondas están desfasadas.
- **Longitud de onda:** Es la distancia entre los ciclos repetitivos de una onda a una frecuencia dada. Cuanto más elevada sea la frecuencia, más corta será la longitud de onda:

$$\lambda = \frac{c}{f}$$

donde $c = 3 * 10^8 \frac{m}{s}$ es la velocidad de la luz.

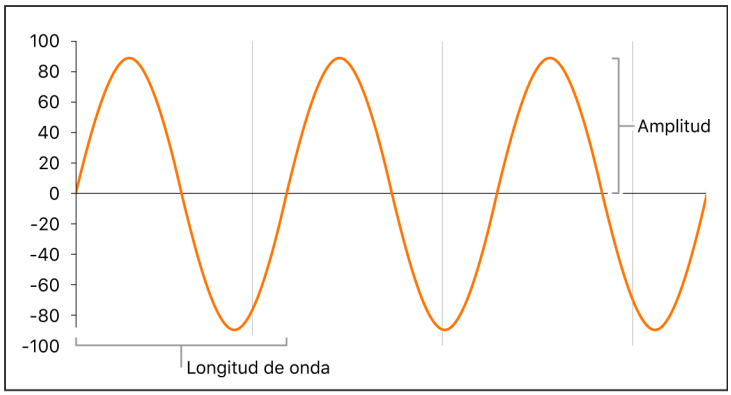

Figura 3: Onda de señal

Esta onda puede chocar con imperfecciones del material (del canal), producir reflexiones y refracciones que se traducen en **perdidas** (menos energía para la señal original en la que está codificado el mensaje)

Dado que las ondas electromagnéticas son continuas y que son modificadas a medida que se propagan por el canal, debemos encontrar una manera de mapear estas frecuencias a los 0 y 1 que componen nuestros mensajes.

Para esto, tanto el transmisor como el recibidor definen un **ancho de banda** que es el rango de frecuencias que va a ocupar las señales que van a ser transmitidas por el canal y dentro de ese rango, cuales deben ser mapeadas a un 1 y cuales a un 0.

3.2. Transformación de fourier

Todas las funciones períodicas pueden expresarse como una suma infinita de senos y cosenos:

$$f(t) = \frac{1}{2}c + \sum_{i=0}^{\infty} (a_i \operatorname{sen}(n\omega t) + b_i \cos(n\omega t))$$

done ω es la frecuencia angular y a_i es la amplitdud de la onda.

Ninguna instalación transmisora puede transmitir señales sin perder cierta potencia en el proceso. Si todos los componentes de Fourier disminuyeran en la misma proporción, la señal resultante se reduciría en amplitud, pero no se distorsionaría. Desgraciadamente, todas las instalaciones de transmisión disminuyen los distintos componentes de Fourier en diferente grado, lo que provoca distorsión. Por lo general, las amplitudes se transmiten sin ninguna disminución desde 0 hasta cierta frecuencia f_c y todas las frecuencias que se encuentren por arriba de esta frecuencia de corte serán atenuadas. El rango de frecuencias que se transmiten sin atenuarse se conoce como **ancho de banda**. En la práctica, el corte no es abrupto, por lo que, en general, el ancho de banda ofrecido va desde 0 hasta la frecuencia en la que el valor de la amplitud es atenuado a la mitad de su valor original.

El ancho de banda es una propiedad física del medio de transmisión y por lo general depende de la construcción, grosor y longitud de dicho medio. En algunos casos, se introduce un filtro en el circuito para limitar la cantidad de ancho de banda disponible para cada cliente.

3.3. Problemas de los medios de transmisión reales

Cuando enviamos un mensaje a una máquina en una red, debemos pasar ese mensajes por un **transmisor** que convertirá el mensaje en una serie de **señales** que pueden ser enviadas a través del **canal** que nos comunica con la máquina de **destino**. La máquina de destino debe tener un **recibidor** que le permita captar las señales del canal y transformarlas nuevamente en el mensaje original.

Sin embargo, los canales de transmisión físicos no son perfectos y aportan **ruido** a las señales emitidas por nuestro transmisor pudiendo llegar a destino con errores.

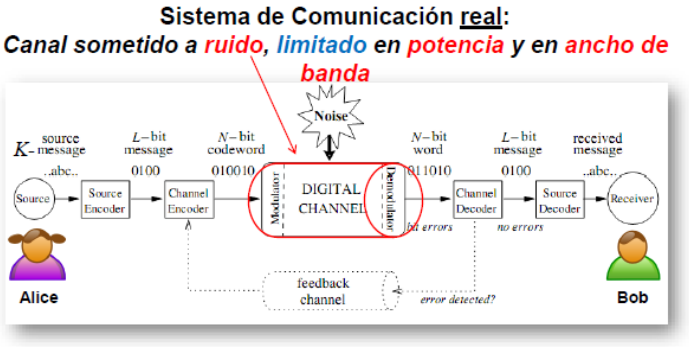

Figura 4: Esquema general de un sistema de comunicación

Una forma de tratar de resolver esto es agregar al modelo nuevas capas de actores que trabajan para reducir los efectos nocivos del ruido. Por ejemplo, podemos agregar al modelo observadores externos que sean capaces de ver lo que se transmite de un lado y se recibe del otro, deducir información a partir de las diferencias, y tener chance de enviarle correcciones al Elemento Corrector. También es posible tener dos niveles de observadores: uno que se maneja a nivel mensaje y otro que se maneja a nivel característica del canal propio.

3.3.1. Tipos de errores de los canales físicos

- **Atenuación:** En medios análogicos, la señales se degradan con la distancia recorrida lo que puede llevar a provocar errores en algunos bits recibidos. Por lo que la intensidad de la señal recibida deber ser suficiente para ser detectada y, además, debe ser suficientemente mayor al ruido del canal para que se reciba sin error. En general, las frecuencias más afectadas son las más altas por lo que se puede ecualizar estas frecuencias, es decir, amplificarlas.
- **Distorsión de retardo:** En medios guiados, la velocidad de propagación en el medio varía con la frecuencia por lo que los componentes del mensaje llegan en distintos instantes de tiempo, originando desplazamiento de fases entre las distintas frecuencias. Esto se puede deber a varios motivos.
- **Ruido:** Los canales físicos poseen ruido natural. Es decir, transmiten señales adicionales debido a agentes externos:
  - **Ruido Término ó Ruido Blanco:** Se produce debido a la agitación térmica de electrones y aumenta linealmente con la temperatura absoluta del canal. En general, está distribuido de manera uniforme a lo largo de todo el canal y para un ancho de banda B, la potencia del ruido blanco $N_b = kTB$.
  - **Ruido por intermodulación:** Son señales que son la suma o la diferencia de sus frecuencias originales producidas por una falta de linealidad en el canal. $N_I = mf_1 \pm nf_2$

- **Ruido por Diafonía:** Se produce cuando una señal de una línea interfiere en otra.
- **Ruido impulsivo:** Son impulsos irregulares o picos que se pueden producir por interferencias externas (como pueden ser interferencia electromagnéticas, tormentas, etc). Este tipo de ruido es de corta duración, tienen gran amplitud y es disruptivo.

3.3.2. Capacidad de un canal

Las perturbaciones mencionadas afectan la velocidad de transmición del canal por lo que debemos asegurarnos de no enviar más bits que la capacidad límite del mismo para no perder información durante la transmición. Para esto vamos a definir los siguientes conceptos:

- C es la capacidad del canal o tasa de datos, es decir a la cantidad de bits por segundo que podemos transmitir a través del mismo.
- B es el ancho de banda por el cual vamos a transmitir nuestros datos, va estar medido en ciclos por segundo (Hertz) y va estar limitado por el transmisor y el medio.
- N es el nivel medio o potencia del ruido del canal
- BER es la tasa de errores de bits por segundo (Bit Error Rate).
- S es la potencia o amplitud de la señal.
- $SNR = S/N$ es la cantidad de ruido térmico presente se mide por la relación entre la potencia de la señal y la potencia del ruido, llamada relación señal a ruido. Por lo general, la relación misma no se expresa; en su lugar, se da la cantidad $10 \log_{10} S/N$. Estas unidades se conocen como decibeles (dB).

En un canal sin ruido, $C = 2B \log_2 M$ donde M es la cantidad de niveles que usamos para representar los símbolos.

En un canal con ruido, Shannon propuso

$$C_{max} = B \log_2(1 + SNR)$$

En principio, si se aumentan el ancho de banda B y la potencia de señal S, aumenta la velocidad binaria. Sin embargo, un aumento de B aumenta el ruido y un aumento de S aumenta las no linealidades y el ruido de intermodulación.

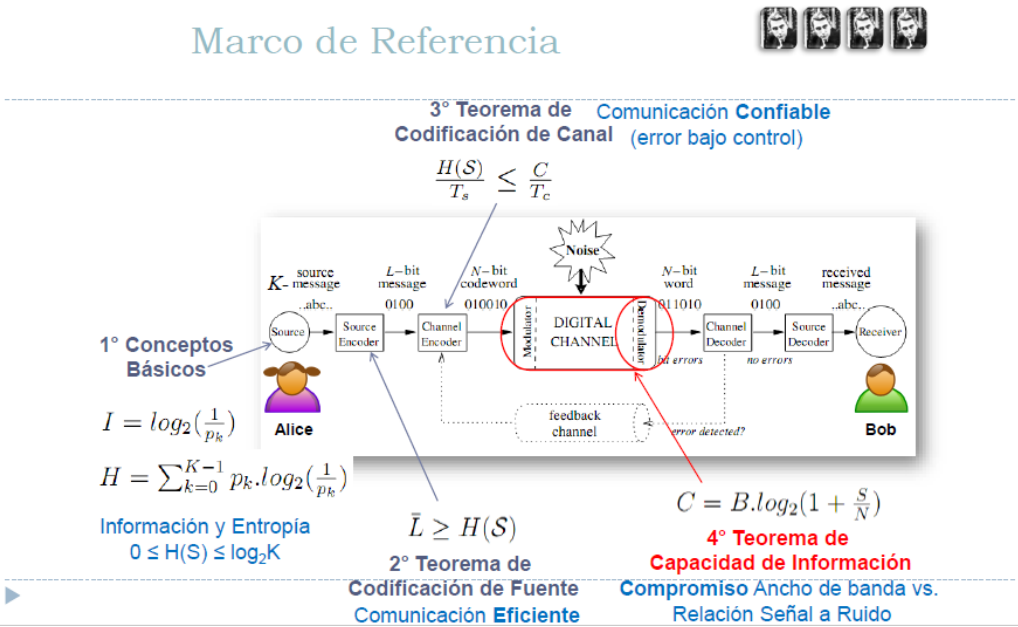

Figura 5: Esquema de un sistema de comunicación y sus conceptos asociados

Límite de eficencia: La eficiencia de ancho de banda es la máxima cantidad de bits por segundo que podemos inyectar por cada Hz sin perder información. Mientra más eficiente sea el canal, se pueden transmitir más bits por segundo.

Límite de confiabilidad: Es la cantidad máxima de bits por segundos que podemos utilizar para transportar una señal de manera confiable a través de un canal ruidoso

3.3.3. Delay

Por último, debemos analizar el tiempo que tarda en llegar un paquete completo desde una punta a otra de la conexión. Este tiempo se ve afectado por varias cosas:

- **Retardo de procesamiento:** Tiempo requerido en analizar el encabezado y decidir a dónde enviar el paquete. En un enrutador, dependerá del número de entradas en la tabla de rutas, la implementación (estructuras de datos), el hardware, etc. Puede incluir la verificación de errores.
- **Retardo de encolamiento:** Tiempo en que el paquete espera en un buffer hasta ser transmitido. El número de paquetes esperando en cola dependerá de la intensidad y la naturaleza del tráfico. Los algoritmos de colas en los enrutadores intentan adaptar estos retardos a ciertas preferencias, o imponer un uso equitativo.
- **Retardo de transmisión:** El tiempo requerido para "empujar" todos los bits de un paquete a través del medio de transmisión. Si R es la capidad del canal, L la longitud del

paquete y D_T el delay o retardo de transmición:

$$D_T = \frac{L}{R}$$

- **Retardo de propagación:** Una vez que el bit es 'empujado' en el medio, el tiempo transcurrido en su propagación hasta el final del trayecto físico. La velocidad de propagación del enlace depende más que nada de la distancia del medio físico. Si d es la distancia a recorrer y s la velocidad de propagación:

$$D_p = \frac{d}{s}$$

3.4. Medios de transmición

3.4.1. Medios guiados

Par trenzado de cobre

Uno de los medios de transmisión más viejos, y todavía el más común. Éste consiste en dos alambres de cobre aislados, en general, de 1 mm de grosor. Los alambres se trenzan en forma helicoidal, igual que una molécula de DNA. Esto se hace porque dos alambres paralelos constituyen una antena simple. Cuando se trenzan los alambres, las ondas de diferentes vueltas se cancelan, por lo que la radiación del cable es menos efectiva.

La aplicación más común del cable de par trenzado es en el sistema telefónico. Casi todos los teléfonos están conectados a la compañía telefónica mediante un cable de par trenzado. La distancia que se puede recorrer con estos cables es de varios kilómetros sin necesidad de amplificar las señales, pero para distancias mayores se requieren repetidores.

Se pueden utilizar para transmisión tanto analógica como digital.

El ancho de banda depende del grosor del cable y de la distancia que recorre; en muchos casos pueden obtenerse transmisiones de varios megabits/seg, en distancias de pocos kilómetros.

Coaxial

Consiste en un alambre de cobre rígido como núcleo, rodeado por un material aislante. El aislante está forrado con un conductor cilíndrico, que con frecuencia es una malla de tejido fuertemente trenzado. El conductor externo se cubre con una envoltura protectora de plástico.

La construcción y el blindaje del cable coaxial le confieren una buena combinación de ancho de banda alto y excelente inmunidad al ruido. El ancho de banda posible depende de la calidad y longitud del cable, y de la relación señal a ruido de la señal de datos. Los cables modernos tienen un ancho de banda de cerca de 1 GHz. Los cables coaxiales solían ser ampliamente usados en el sistema telefónico para las líneas de larga distancia, pero en la actualidad han sido reemplazados por la fibra óptica en rutas de distancias considerables. Sin embargo, el cable coaxial aún se utiliza ampliamente en la televisión por cable y en las redes de área metropolitana.

Fibra óptica:

Un sistema de transmisión óptico tiene tres componentes: la fuente de luz, el medio de transmisión y el detector. Convencionalmente, un pulso de luz indica un bit 1 y la ausencia de luz indica un bit 0. El medio de transmisión es una fibra de vidrio ultradelgada. El detector genera un pulso eléctrico cuando la luz incide en él. Al agregar una fuente de luz en un extremo de una fibra óptica y un detector en el otro, se tiene un sistema de transmisión de datos unidireccional que acepta una señal eléctrica, la convierte y transmite mediante pulsos de luz y, luego, reconvierte la salida a una señal eléctrica en el extremo receptor.

Cuando un rayo de luz pasa por un medio a otro —por ejemplo, de sílice fundida al aire—, el rayo se refracta (se dobla) en la frontera de la sílice y el aire. Un rayo de luz que incide en la frontera con un ángulo α_1 y que emerge con un ángulo β_1. El grado de refracción depende de las propiedades de los dos medios (en particular sus índices de refracción). Para ángulos con incidencias mayores de ciertos valores críticos, la luz se refracta nuevamente a la sílice; ninguna parte de él escapa al aire. Por lo tanto, un rayo de luz que incide en un ángulo mayor o igual que el crítico queda atrapado dentro de la fibra.

Puesto que cualquier rayo de luz que incida en la frontera con un ángulo mayor que el crítico se reflejará internamente, muchos rayos estarán rebotando con ángulos diferentes. Se dice que cada rayo tiene un modo diferente, por lo que una fibra que tiene esta propiedad se denomina **fibra multimodo**. Por otro lado, si el diámetro de la fibra se reduce a unas cuantas longitudes de onda de luz, la fibra actúa como una guía de ondas y la luz se puede propagar sólo en línea recta, sin rebotar, lo cual da como resultado una **fibra monomodo**.

Las fibras monomodo son más caras, pero se pueden utilizar en distancias más grandes. Las fibras monomodo disponibles en la actualidad pueden transmitir datos a 50 Gbps a una distancia de 100 km sin amplificación.

3.4.2. Medios inalámbricos

El espectro electromagnético

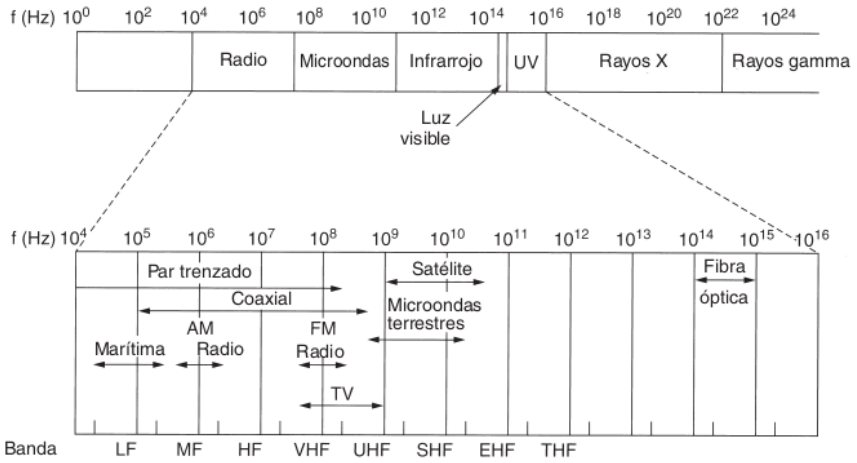

Figura 6: Espector magnetico

Las porciones de radio, microondas, infrarrojo y luz visible del espectro pueden servir para transmitir información modulando la amplitud, frecuencia o fase de las ondas. La luz ultravioleta, los rayos X y los rayos gamma serían todavía mejores, debido a sus frecuencias más altas, pero son difíciles de producir y modular, no se propagan bien entre edificios y son peligrosos para los seres vivos.

La cantidad de información que puede transportar una onda electromagnética se relaciona con su ancho de banda. Con la tecnología actual, es posible codificar unos cuantos bits por hertz a frecuencias bajas, pero a frecuencias altas el número puede llegar hasta 8 bits, de modo que un cable coaxial con un ancho de banda de 750 MHz puede transportar varios gigabits/seg.

A las bandas más altas se les nombró como bandas VHF (frecuencia muy alta), UHF (frecuencia ultraalta), EHF (frecuencia extremadamente alta) y THF (frecuencia tremendamente alta).

Radio

Las ondas de radio son fáciles de generar, pueden viajar distancias largas y penetrar edificios sin problemas, y por ello su uso está muy generalizado en la comunicación, tanto en interiores como en exteriores. Las ondas de radio también son omnidireccionales, lo que significa que viajan en todas direcciones a partir de la fuente, por lo que no es necesario que el transmisor y el receptor se encuentren alineados físicamente.

Las propiedades de las ondas de radio dependen de la frecuencia. A bajas frecuencias, esas ondas cruzan bien casi cualquier obstáculo, pero la potencia se reduce de manera drástica a medida que se aleja de la fuente.

En las bandas VLF, LF y MF las ondas de radio siguen la curvatura de la Tierra. Estas ondas se pueden detectar hasta a 1000 km en las frecuencias más bajas, y a menos en frecuencias más altas.

En las bandas HF y VHF, las ondas a nivel del suelo tienden a ser absorbidas por la tierra. Sin embargo, las ondas que alcanzan la ionosfera, una capa de partículas cargadas que rodea a la Tierra a una altura de 100 a 500 km, se refractan y se envían de regreso a nuestro planeta. En ciertas condiciones atmosféricas, las señales pueden rebotar varias veces. Los operadores de radio aficionados usan estas bandas para conversar a larga distancia. El ejército se comunica también en las bandas HF y VHF.

Láser

La señalización óptica coherente con láseres es inherentemente unidireccional, de modo que cada edificio necesita su propio láser y su propio fotodetector. Este esquema ofrece un ancho de banda muy alto y un costo muy bajo. También es relativamente fácil de instalar.

Una desventaja es que los rayos láser no pueden penetrar la lluvia ni la niebla densa, pero normalmente funcionan bien en días soleados. Sin embargo, el calor del sol causa corrientes de convección que se elevabn desde el techo de los edificio. Este aire turbulento desviaba el rayo y lo hacía danzar alrededor del detector.

Satélites

Los satélites de comunicaciones tienen algunas propiedades interesantes que los hacen atractivos para muchas aplicaciones. En su forma más simple, un satélite de comunicaciones se puede considerar como un enorme repetidor de microondas en el cielo. Contiene numerosos transponedores, cada uno de los cuales se encarga de una parte del espectro, amplifica la señal entrante y a continuación la retransmite en otra frecuencia para evitar interferencia con la señal entrante. Los haces pueden ser amplios y cubrir una fracción sustancial de la superficie de la Tierra, o estrechos, y abarcar sólo algunos cientos de kilómetros de diámetro. Este modo de operación se conoce como de tubo doblado.

3.5. Red télefonica

La **Red Telefónica Pública Conmutada** (PSTN), fue diseñadas hace muchos años, con el propósito de transmitir la voz humana en una forma más o menos reconocible.

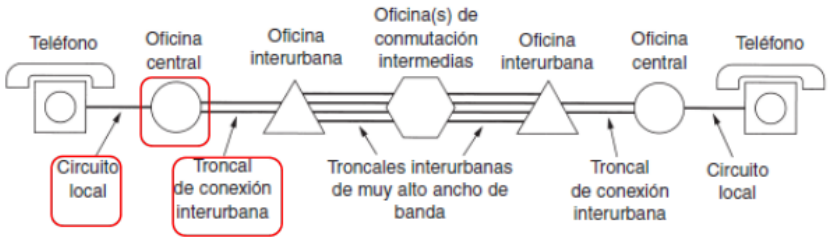

Figura 7: Sistema telefónico

El sistema telefónico consiste en tres componentes principales:

1. Circuitos locales: Cables de par trenzado que van hacia las casas y las empresas. Dan acceso a todo mundo al sistema completo, debido a lo cual son cruciales. Por desgracia, también son la parte más débil del sistema. Cada oficina central tiene varias líneas salientes a uno o más centros de conmutación cercanos, llamados **oficinas interurbanas**.
2. Troncales (fibra óptica digital que conecta a las oficinas de conmutación).
3. Oficinas de conmutación (donde las llamadas pasan de una troncal a otra).

Para las troncales de largo alcance, la principal consideración es cómo reunir múltiples llamadas y enviarlas juntas por la misma fibra. Este tema se llama **multiplexión**.

3.5.1. Tipos de multiplexión

Las compañías telefónicas han desarrollado esquemas complejos para multiplexar muchas conversaciones en una sola troncal física. Estos esquemas de multiplexión se pueden dividir en dos categorías principales: FDM (Multiplexión por División de Frecuencia) y TDM (Multiplexión por División de Tiempo). En FDM el espectro de frecuencia se divide en bandas de frecuencia, y cada usuario posee exclusivamente alguna banda. En TDM los usuarios esperan su turno (en round-robin), y cada uno obtiene en forma periódica toda la banda durante un breve lapso de tiempo.

Multiplexión por División de Frecuencia: Los filtros limitan el ancho de banda utilizable a cerca de 3000 Hz por canal de calidad de voz. Cuando se multiplexan muchos canales juntos, se asignan 4000 Hz a cada canal para mantenerlos bien separados. Primero se eleva la frecuencia de los canales de voz, cada uno en una cantidad diferente, después de lo cual se pueden combinar, porque en ese momento no hay dos canales que ocupen la misma porción del espectro. Hay cierta superposición entre canales adyacentes porque los filtros no tienen bordes bien definidos. Esta superposición significa que un pico fuerte en el borde de un canal se detectará en el adyacente

como ruido no térmico. Para los canales de fibra óptica se utiliza una variante de la multiplexión por división de frecuencia llamada **Multiplexión por División de Longitud de Onda** (WDM).

Multiplexión por División de Tiempo: Aunque FDM aún se utiliza sobre cables de cobre o canales de microondas, requiere circuitos analógicos y no es fácil hacerla con una computadora. En contraste, TDM puede manejarse por completo mediante dispositivos digitales y a ello se debe su popularidad en los últimos años. Desgraciadamente, sólo se puede utilizar para datos digitales. Puesto que los circuitos locales producen señales analógicas, se necesita una conversión de analógico a digital en la oficina central, en donde todos los circuitos locales individuales se juntan para combinarse en troncales.

3.5.2. Conversión analógico digital

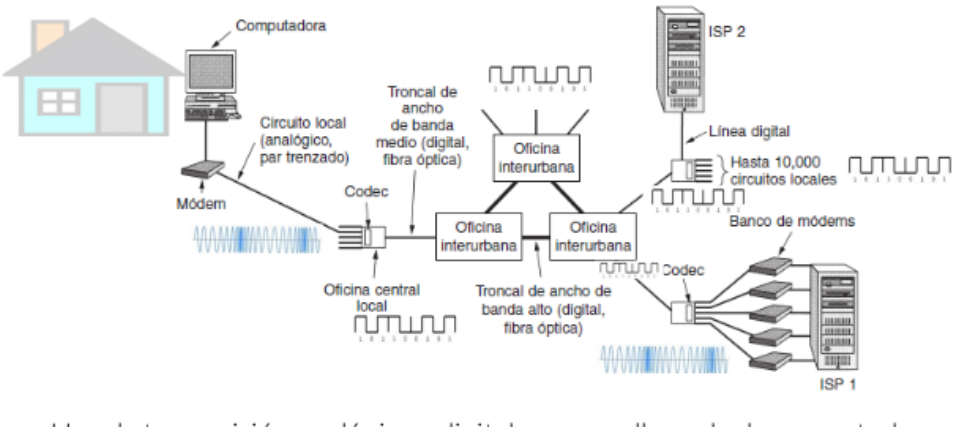

Figura 8: Conversión analogico digital

3.6. Modulación

La transmisión analógica se realiza en una señal continua de frecuencia constante denomidada **portadora**. La frecuencia de la portadora se elige de tal forma que sea compatible con las características del medio que se vaya a utilizar. Los datos se pueden transmitir modulando la señal portadora para asegurarnos de que llegue al próximo extremo sin perder información. Todas las técnicas de modulación implican la modificación de uno o más de los tres parámetros fundamentales de la frecuencia portadora: amplitud, frecuencia y fase.

La señal de entrada $m(t)$ se denomina señal **moduladora** a la señal resultante de la modulación de la señal portadora.

Hay distintos tipos de modulaciones:

- **Desplazamiento de amplitud (ASK):** Los dos valores binarios se representan mediante dos amplitudes diferentes de la portadora. Es usual que una de las amplitudes sea cero.

$$S(t) = \begin{cases} A\cos(2\pi f_c t) & 1 \text{ binario} \\ 0 & 0 \text{ binario} \end{cases}$$

  ASK es sensible a cambios repentinos de la ganancia, además es una técnica de modulación bastante ineficaz. Se usa para la transmición de datos digitales en fibras ópticas.

- **Desplazamiento de frecuencia (FSK):** Los dos valores binarios se representan mediante dos frecuencias diferentes próximas a la frecuencia portadora. La señal resultante es

$$S(t) = \begin{cases} A\cos(2\pi f_1 t) & 1 \text{ binario} \\ A\cos(2\pi f_2 t) & 0 \text{ binario} \end{cases}$$

  donde f_1 y f_2 corresponden a desplazamientos de la frecuencia portadora f_c, de igual magnitud pero en sentidos opuestos.

  FSK es menos sensible a errores que ASK.

- **Desplazamiento de fase (PSK):** La fase de la señal portadora se desplaza para representar con ello a los datos digitales: En este sistema, un cero binario se representa mediante la transmición de una señal con la misma fase que la fase de la señal anteriormente enviada. Mientras que un uno se representa mediante la transmición de una señal cuya fase esté en opocisión de fase respecto de la señal precedente. Esta técnica se conoce como PSK diferencial, ya que el desplazamiento en fase es relativo a la fase correspondiente al último símbolo transmitido.

$$S(t) = \begin{cases} A\cos(2\pi f_c t + \pi) & 1 \text{ binario} \\ A\cos(2\pi f_c t) & 0 \text{ binario} \end{cases}$$

Velocidad de modulación: El número de cambios de señal por unidad de tiempo. Se expresa en **baudios** (símbolos por segundo).

Velocidad de transmición: Equivale a la velocidad de modulación multiplicado por el número de bits N representados por cada símbolo. Se expresa en bits por segundo: $V_t = V_m \cdot m$

Se puede conseguir una utilización más eficaz del ancho de banda si cada elemento de señalización representa más de un bit. En el método PSKQ se consideran desplazamientos de fase correspondientes a $\pi/2$ (90°) por lo que cada señal representa dos bits en lugar de uno:

$$S(t) = \begin{cases} A\cos(2\pi f_c t + \frac{\pi}{4}) & 11 \\ A\cos(2\pi f_c t + \frac{3\pi}{4}) & 10 \\ A\cos(2\pi f_c t + \frac{5\pi}{4}) & 00 \\ A\cos(2\pi f_c t + \frac{7\pi}{4}) & 01 \end{cases}$$

3.6.1. Codificación

Una vez modulada, la señal analógica viaja hasta un **codec** en la oficina central que se encarga de digitalizarla. El codec toma 8000 muestras por segundo porque el teorema de Nyquist dice que esto es suficiente para capturar toda la información del ancho de banda de 4 kHz del canal telefónico. A una velocidad de muestreo menor, la información se perdería; a una mayor, no se ganaría información extra. Esta técnica se llama **Modulación por Codificación de Impulsos** (PCM).

A veces, se usa un técnica conocida como **Modulación Delta**, en el que la entrada analógica se aproxima mediante una función escalera que en cada intervalo de muestreo (T_S) sube o baja un nivel de cuantización δ. La característica principal de la función escalera es que su comportamiento es binario: En cada instante de muestreo, la función sube o baja una cantidad constante. Por tanto, la salida del modulador delta se puede representar mediante un único bit para cada muestra. Resumiendo: se obiente una cadena de bits que aproxima a la derivada de la señal analógica de entrada en cualquier lugar de la amplitud.

No retorno a cero (NRZ)

La forma más frecuente y fácil de transmitir señales digitales es mediante la utilización de un nivel diferente de tensión para cada uno de los dos dígitos binarios. Los códigos que siguen esta estrategia comparten la propiedad de que el nivel de tensión se mantiene constante durante la duración del bit. Es habitual usar un nivel negativo para representar un valor binario y un tensión positiva para representar el otro.

Una variante de este código se nomina **No retono a Cero con Inversion de unos** (NZRI) en la que un 1 se codifica mediante la transición al principio del intervalo de señalización, mientras que un 0 se representa por la ausencia de transición. Esta codificación es un ejemplo de codificación diferencial, en lugar de determinar el valor absoluto, la señal se codifica comparando la polaridad de los elementos de señal adyacentes.

Codificación Manchester (Bifase)

El valor de un bit se códifica en una transición a la mitad del intervalo de duración del bit. Esta transición en la mitad del bit sirve como un procedimiento de sincronización a la vez que sirve para transmitir los datos: Una transción de bajo a alto representa un 1 y una transición de alto a bajo representa un cero.

En Manchester diferencial, la transición a mitad del intervalo se utiliza tan solo para proporcionar sincronización. La codificación de un cero se representa por la presencia de una trancisión al principio del intervalo.

Toda técnica bifase fuerza al menos una transición por cada bit pudiendo tener hasta dos en ese mismo período. Por tanto, la velocidad de modulación máxima es el doble que en los NRZ y el ancho de banda necesario es mayor.

3.7. Redes de conmutación

En la actualidad se utilizan dos técnicas de conmutación diferentes: conmutación de circuitos y conmutación de paquetes. A continuación presentaremos una breve introducción a cada una de ellas.

Conmutación de circuitos

Cuando se realiza una llamada telefónica, el equipo de conmutación del sistema telefónico busca una trayectoria física que vaya desde su teléfono al del receptor. Esta técnica se llama **conmutación de circuitos**.

Una propiedad importante de la conmutación de circuitos es la necesidad de establecer una trayectoria de un extremo a otro antes de que se pueda enviar cualquier dato. El tiempo que transcurre entre que se termina de marcar y que el timbre comienza a sonar puede ser fácilmente de 10 seg, y más en las llamadas de larga distancia o internacionales. Durante este intervalo de tiempo, el sistema telefónico busca una trayectoria de cobre. La señal de petición de llamada se debe propagar hasta el destino y se debe confirmar su recepción. En muchas aplicaciones de computadora, los tiempos de establecimiento largos son indeseables.

Por otro lado, al existir una trayectoria de cobre entre las partes en comunicación, una vez que se termina de establecer, el único retardo de los datos es el tiempo de propagación de la señal electromagnética y no hay peligro de congestión; es decir, una vez que la llamada entra, no hay posibilidad de obtener una señal de ocupado, aunque podría obtener una antes de establecer la conexión debido a la falta de capacidad de conmutación o de troncal.

Conmutación de paquetes

La alternativa a la conmutación de circuitos es la **conmutación de paquetes**. Con esta tecnología, los paquetes individuales se envían conforme se necesite, y no se les asigna por adelantado ninguna trayectoria dedicada.

En este caso, al no ser necesaria una conexión previa, el primer paquete se puede enviar apenas esté listo.

Con la conmutación de paquetes no hay trayectoria, por lo que diferentes paquetes pueden seguir trayectorias distintas, dependiendo de las condiciones de la red en el momento en el que se enviaron. Pueden llegar en desorden.

La conmutación de paquetes es más tolerante a las fallas que la conmutación de circuitos. De hecho, ésa es la razón por la cual se inventó. Si falla la conmutación, todos los circuitos que la están utilizando se cancelan y no se puede enviar nada más a través de ellos. Con la conmutación de paquetes, los paquetes pueden enrutarse evitando a los conmutadores averiados.

La conmutación de paquetes utiliza transmisión de almacenamiento y reenvío. Un paquete se almacena en la memoria del enrutador y luego se reenvía al siguiente enrutador. Con la conmutación de paquetes los bits simplemente fluyen de manera continua a través del cable. La técnica de almacenamiento y reenvío agrega retardo.

Parte II
Nivel de enlace

4. Introducción

La capa de enlace de datos tiene que desempeñar varias funciones específicas, entre las que se incluyen:

- Proporcionar una interfaz de servicio bien definida con la capa de red.
- Manejar los errores de transmisión.
- Regular el flujo de datos para que receptores lentos no sean saturados por emisores rápidos.

Para cumplir con estas metas, la capa de enlace de datos toma de la capa de red los paquetes y los encapsula en **tramas** para transmitirlos. Cada trama contiene un **encabezado**, un campo de carga útil (**payload**) para almacenar el paquete y un **terminador** o final. El manejo de las tramas es la tarea primordial de la capa de enlace de datos.

4.1. Servicios proporcionados

La capa de enlace de datos puede diseñarse para ofrecer varios servicios. Los servicios reales ofrecidos pueden variar de sistema a sistema. Tres posibilidades razonables que normalmente se proporcionan son:

1. **Servicio no orientado a la conexión sin confirmación de recepción:** Consiste en hacer que la máquina de origen envíe tramas independientes a la máquina de destino sin pedir que ésta confirme la recepción. No se establece conexión de antemano ni se libera después. Si se pierde una trama debido a ruido en la línea, en la capa de enlace de datos no se realiza ningún intento por detectar la pérdida ni por recuperarse de ella.
2. **Servicio no orientado a la conexión con confirmación de recepción:** Cuando se ofrece este servicio tampoco se utilizan conexiones lógicas, pero se confirma de manera individual la recepción de cada trama enviada. De esta manera, el emisor sabe si la trama ha llegado bien o no. Si no ha llegado en un tiempo especificado, puede enviarse nuevamente.
3. **Servicio orientado a la conexión con confirmación de recepción:** Con este servicio, las máquinas de origen y de destino establecen una conexión antes de transferir datos. Cada trama enviada a través de la conexión está numerada, y la capa de enlace de datos garantiza que cada trama enviada llegará a su destino. Es más, garantiza que cada trama será recibida exactamente una vez y que todas las tramas se recibirán en el orden adecuado.

4.2. Separación de frames:

Puesto que es demasiado riesgoso depender de la temporización para marcar el inicio y el final de cada trama, se han diseñado otros métodos:

1. **Largo fijo:** Todos los paquetes tienen la misma longitud.

2. **Conteo de caracteres**: Se vale de un campo en el encabezado para especificar el número de caracteres en la trama. Cuando la capa de enlace de datos del destino ve la cuenta de caracteres, sabe cuántos caracteres siguen y, por lo tanto, dónde está el fin del paquete. El problema con este algoritmo es que la cuenta puede alterarse por un error de transmisión. Si esto pasa, el destino perderá la sincronía y será incapaz de localizar el inicio de la siguiente trama. Incluso si el destino sabe que la trama está mal porque la suma de verificación es incorrecta, no tiene forma de saber dónde comienza la siguiente. Regresar una trama a la fuente solicitando una retransmisión tampoco ayuda, ya que el destino no sabe cuántos caracteres tiene que saltar para llegar al inicio de la retransmisión.

3. **Flags con bit-stuffing**: El segundo método de entramado evita el problema de tener que sincronizar nuevamente después de un error, haciendo que cada trama inicie y termine con bytes especiales llamado **flags**. De esta manera, si el receptor pierde la sincronía simplemente puede buscar el flag para encontrar el final e inicio de la trama actual. Dos flags consecutivos señalan el final de una trama y el inicio de la siguiente.

   Cuando se utiliza este método para transmitir datos binarios, como programas objeto o números de punto flotante, surge un problema serio. Se puede dar el caso con mucha facilidad de que el patrón de bits del flag aparezca en los datos (payload), lo que interferiría en el entramado. Una forma de resolver este problema es hacer que la capa de enlace de datos del emisor inserte un byte de **escape especial** (ESC) justo antes de cada flag "accidental" en los datos. La capa de enlace de datos del lado receptor quita el byte de escape antes de entregar los datos a la capa de red. Esta técnica se llama **relleno de caracteres** o bit-stuffing. Por lo tanto, un flag de entramado se puede distinguir de uno en los datos por la ausencia o presencia de un byte de escape que le anteceda.

4.3. Detección y corrección de errores

Los diseñadores de redes han desarrollado dos estrategias principales para manejar los errores. Una es incluir suficiente **información redundante** en cada bloque de datos transmitido para que el receptor pueda deducir lo que debió ser el carácter transmitido. La otra estrategia es incluir sólo suficiente redundancia para permitir que el receptor sepa que ha ocurrido un error (pero no qué error) y entonces solicite una retransmisión. La primera estrategia utiliza **códigos de corrección de errores**; la segunda usa **códigos de detección de errores**.

4.3.1. Detección de errores

Bit de paridad: La forma más sencilla. Consiste en añadir un bit de paridad al final del bloque de datos. El valor de ese bit se determina de tal forma que el código resultante tenga un número impar de unos. El receptor examina el código recibido y, si el número total es impar, supondrá que no ha habido errores.

La utilización de bits de paridad no es infalible, ya que los impulsos de ruido son a veces lo suficientemente largos como para destruir más de un bit.

Comprobación de redundancia cíclica (CRC): Dado un bloque o mensaje de k-bits, el trasmisor genera una secuencia de n-bits denominada **secuencia de comprobación de la trama** de tal manera que la trama resultante con $n + k$ bits, es divisible por algún número determinado. El receptor entonces dividirá la trama recibida por ese número y, si no hay resto en la división, se supone que no ha habido errores.

4.3.2. Códigos de corrección de errores

Por lo general, una trama consiste en m bits de datos (es decir, de mensaje) y r bits redundantes o de verificación. Sea la longitud total n (es decir, $n = m + r$). A una unidad de n bits que contiene datos y bits de verificación se le conoce como palabra codificada de n bits.

Dadas dos palabras codificadas cualesquiera es posible determinar cuántos bits correspondientes difieren aplicando un OR exclusivo a las dos palabras codificadas y contar la cantidad de bits 1 en el resultado. La cantidad de posiciones de bits en la que difieren dos palabras codificadas se llama **distancia de Hamming**. Si dos palabras codificadas están separadas una distancia de Hamming d, se requerirán d errores de un bit para convertir una en la otra.

Para detectar e errores se necesita un código con distancia $e + 1$, pues con tal código no hay manera de que e errores de un bit puedan cambiar una palabra codificada válida a otra. Cuando el receptor ve una palabra codificada no válida, sabe que ha ocurrido un error de transmisión. De manera similar, para corregir e errores se necesita un código de distancia $2e + 1$, pues así las palabras codificadas legales están tan separadas que, aun con e cambios, la palabra codificada original sigue estando más cercana que cualquier otra palabra codificada, por lo que puede determinarse de manera única.

4.4. Protocolos de Transmición confiable

4.4.1. Stop & Wait

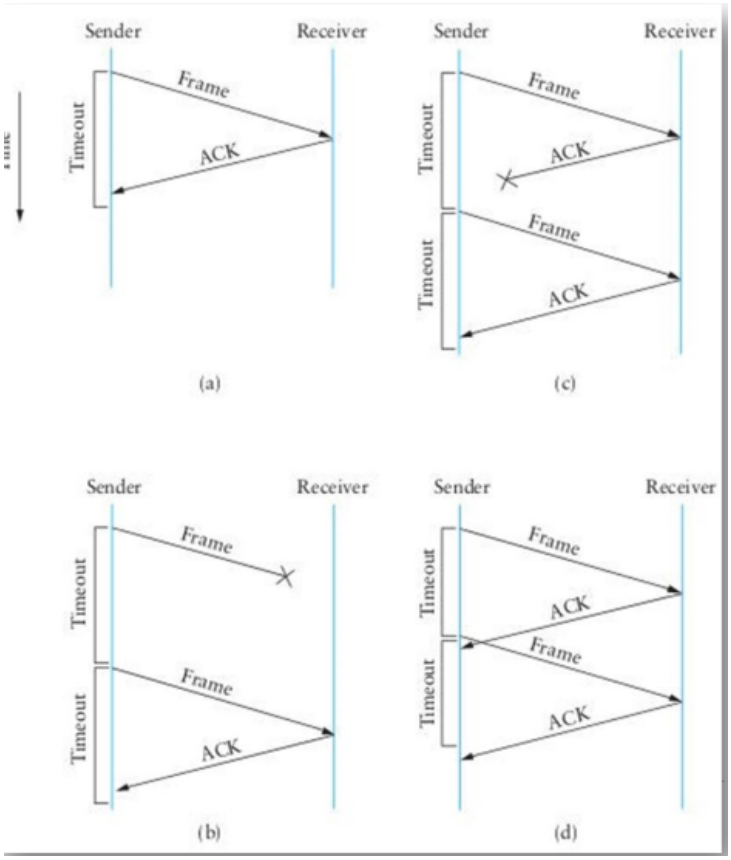

Figura 9: Protocolo Stop And Wait

La máquina de origen emite un único frame y espera la recepción de una confirmación (ACKnowledgment) durante un período determinado de tiempo t. Mientras tanto no podrá enviar ningún otro frame.

Pueden surgir dos problemas durante la comunicación:

El receptor detecta algún error, entonces descartará el frame y no manda el ACK. El temporizador expira antes de la recepción del ACK y re-envía el último frame.

Por otro lado, si el frame llega correctamente y el receptor envía una ACK pero el ACK se pierde en el camino, tambien expira el temporizado y el origen vuelve a enviar el mismo frame.

El destino recibe por segunda vez el frame como si fuese un frame distinto. Para que el receptor pueda reconocerlo como duplicado, los frames se etiquentan alternadamente con 0 1 y las confirmaciones positivas serán de la forma ACK0 y ACK1. Un ACK0 confirma la recepción de un frame numerado con 0 e indica que el receptor está esperando para aceptar un frame

numerado con 0.

Ahora, cuando el receptor reciba dos frames con la misma numeración, podrá interpretar que está recibiendo un frame duplicado porque su ACK se perdió en el camino. En este caso, el receptor descartará el frame y volverá a mandar el ACK correspondiente.

Eficiencia de un protocolo: Vamos a definir:

- T_{tx} el tiempo de transmición de un frame (lo que tarda en ir desde el origen hasta el destino).
- $RTT(F)$ el tiempo de retorno del ACK, osea el tiempo que tarda en llegar el frame al receptor sumado al tiempo que tarda en llegar el ACK al emisor original. En genaral va a pasar que $RTT(F) = Delay \times 2$

El rendimiento de un protocolo η se define como:

$$\eta_{proto} = \frac{T_{tx}(F)}{RTT(F)}$$

4.5. Ventana deslizante

Aumentar la eficiencia η implica disminuir al minimo la cantidad de tiempo que el origen se bloquea durante la espera de un ACK.

Una estrategia posible para esto es enviar varios frames seguidos sin esperar ACKs para cada uno. Aparece el concepto de **ventana de frames**: en una ventana se envía una cierta cantidad de frames. Esto resulta en una definición diferente para la eficiencia:

$$\eta_{proto} = \frac{T_{tx}(V)}{RTT(F)}$$

donde $T_{tx}(V)$ es el tiempo de transmición de una ventana.

Sea V_{tx} la velocidad de transmición, definimos la capacidad de volumen de un canal $C_{vol} = V_{tx} \times Delay$ como la cantidad de bits que entran en un canal de manera simultánea sin saturarlo. Con esto, podemos calcular el **Sliding Window Size (SWS)** de la siguiente manera:

$$SWS = \frac{V_{tx} \times RTT(F)}{|Frames|} \text{ frames}$$

Al comenzar la comunicación, la máquina origen manda SWS frames a la máquina destino. Y espera a recibir el ACK del primer frame enviado. Cuando esto sucede, el origen manda el siguiente frame de la secuencia.

Por cada ACK recibido, el origen va mandando de a uno los frames restantes siempre y cuando el último frame enviado pertezca a una ventana que contenga al frame del último ACK recibido.

- **ACKs Acumulativos:** Si un ACK se pierde, el receptor descarta todos los paquetes hasta que reciba el paquete que está esperando. Esto va a provocar que se produzca un timeout en el emisor del primer paquete perdido y de todos los siguientes, por lo que volverá a enviar todos los frames de nuevo desde el último ACK recibido.

- **ACKs Selectivos:** El receptor recibe y se guarda todos los frames que van llegando. Supongamos que recibe el frame i con errores. Cuando llega el frame $i+1$ responde al receptor con un NAKi. Esto le indica al emisor que si bien recibió el último paquete enviado, hubo un error en el frame anterior y tiene que reenviarlo.

  El receptor sigue aceptando frames y respondiendo con ACK$i-1$ hasta que consigue correctamente el frame perdido. En este momento, responde con un ACKj donde j corresponde con el último frame que recibió sin error.

Los frames están enumerados de manera cíclica desde 1 hasta $SWS + RWC$ donde RWC es el tamaño de la ventana de recepción y se define de la siguiente manera:

$$RWC = \begin{cases} SWS & \text{si hay ACKs Selectivos} \\ 1 & \text{si hay ACKs Acumulativos} \end{cases}$$

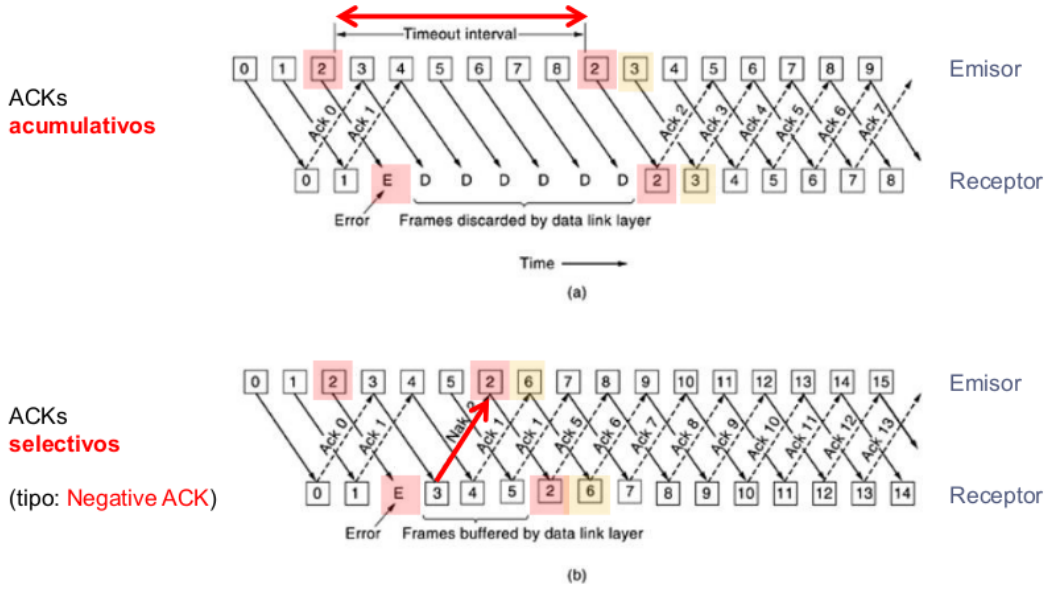

Figura 10: Protocolo Sliding Window

Parte III
Medios Compartidos

La idea es controlar el acceso a los medios compartidos de manera tal que haya la menor cantidad de intervención humana posible en el proceso. Es decir, se busca no tener la figura de un administrador que tenga que solucionar manualmente las cosas.

Los procesos de multiplexión y división de banda vistos en la primer sección comparten una característica: debe estar decidido a priori qué usuario está usando qué parte del tiempo, frecuencia, etc. en cada momento. Si bien sirven para las redes troncales en donde la cantidad de nodos conectados cambia con poca frecuencia, no sirve en redes LAN o Wi-Fi donde los usuarios se conectan y desconectan constantemente a gusto. Esto es porque requieren un administrador dedicado a una red con características estáticas y rígidas y no escala de manera automática.

Los protoclos utilizados en estas redes dinámicas usan lo que se conoce **contención estadística**, en la que se intenta máximizar la cantidad de transmiciones exitosas asegurando la igualdad de opotrtunidades para cada usuario y, en caso de que haya algún conflicto, resolverlos de manera automática. A este tipo de protocolos lo llamamos **MAC Protocols (Medium Access Control Protocols)**. En estos casos, el control es descentralizado, y surge la necesidad de un esquema de direccionamiento y de controlar el acceso.

Ejemplos: Aloha, Ethernet, Wi-Fi, Token Ring.

5. Ethernet (IIEEE 802.3)

Las computadoras de un edificio se conectan entre sí por medio de cables. Cada versión de Ethernet tiene un máximo de distancia física entre segmentos. Para permitir conexiones de mayor distancia se pueden utilizar **repetidores**, que son dispositivos que amplifican y retransmiten señales en ambas direcciones. El formato utilizado para enviar mensajes en la red es el siguiente:

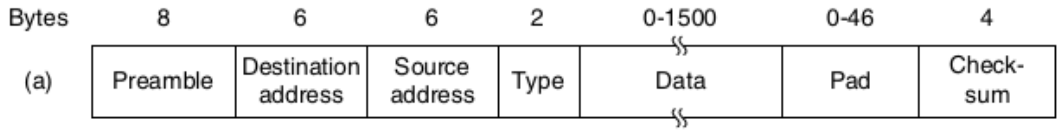

Figura 11: Frame del protocolo Ethernet

- Los primeros 8 bytes son un preambulo que permiten a los receptors sincronizarse con la señal.
- Luego, vienen dos direcciones, cada uno de 6 bytes.

  Cuando el primer bit de la dirección de destino es un 0, significa que el paquete está dirigido a una máquina específica. Si es un 1, entonces es un dirección grupal: Estas direcciones permiten que varias máquinas escuchen una única dirección y que todas las máquinas

pertenecientes al grupo reciban los paquetes dirigidas a esa dirección. Este tipo de envío se llama **multicasting**. Además, la dirección especial que consiste en todos bits de valor 1 está reservada para hacer **Broadcasting** (enviar un mensaje a todos los dispositivos de la red).

- Los próximos 2 bytes identifican el tipo de protocolo que debe usarse para procesar el contenido del paquete.
- Después viene la información per se, que puede ocupar hasta 1500 bytes (un valor decidido de manera arbitraria).
- Si la data enviada es menor a 46 bytes, se usa padding hasta completar los 46 bytes necesarios para cumplir con los requirimientos de longitud mínima del mensaje. Esta longitud mínima permite que la máquina de origen detecte colisiones durante la transmición, en el caso de haberlas (más adelante explicado).
- Los últimos 4 bytes del paquete son un checksum (CRC de 32 bits) que permite detectar si hubo algún error en el frame, si lo hubo, el frame se descarta.

5.1. Colisiones

Una de las razones para tener una longitud mínima de un frame es para evitar que una máquina termine de transmitir el frame antes de que el primer bit haya alcanzado el otro extremo del cable, donde podría colisionar con otro frame.

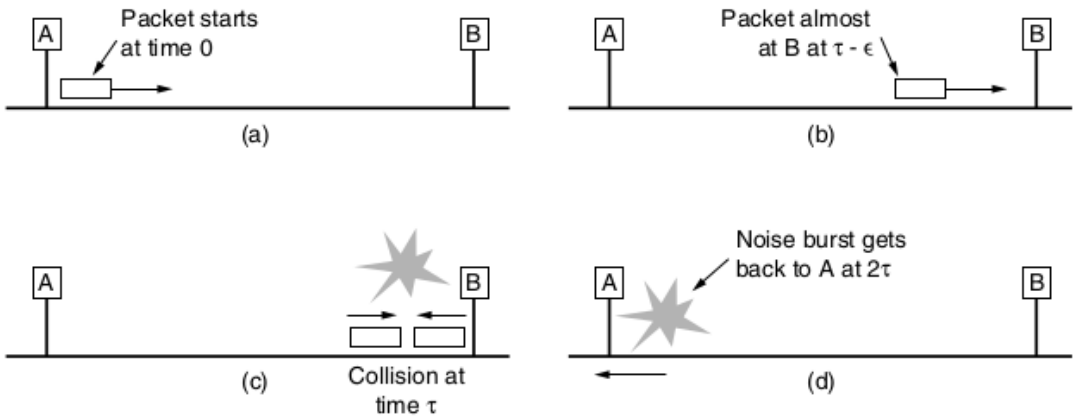

Figura 12: Detección de Colisiones

Supongamos que en un timpo 0, el host A comienza a transmitir el paquete. Sea τ el tiempo de propagación necesario para que el frame llegue al host B. Supongamos que justo antes de que el frame llegue a destino, en el tiempo $\tau - \epsilon$, B comienza a transmitir. Cuando B detecta que está

recibiendo más energía que la que está emitiendo, se da cuenta que ocurre una colisión, aborta su propia transmisión y genera una rafaga de ruid de 48 bits para alertar a las demás estaciones.

En otras palabras, genera interferencia para asegurarse que el emisor (A) se de cuenta de que ocurrió la colisión. En el momento 2τ, A ve el ruido y aborta su propia transmición. Luego espera un intervalo de tiempo random antes de intentar de nuevo.

Si A trata de enviar un frame muy corto, puede llegar a pasar que termine de trasmitir antes de que perciba el ruido generado por B (en el momento 2τ). Entonces A concluiria que el frame se envió correctamente. Por esta razón, se utiliza el pading en el frame de Ethernet para completar la longitud mínima de 64 bytes.

Este valor se deduce de las especificacioens de IIEEE 802.3: Para una red de 10Mbps con una longitud de 2500 metros y a lo sumo 4 repetidores, se determinó que el round-trip time es de 50 μsegundos, asi que 500 bits es el frame más corto posible para detectar colisiones. Este valor se redondeó a 512 bits (64 bytes).

5.2. CSMA/CD con Exponential BackOff

CSMA/CD (Carrier Sense Multiple Access with Collision Detection), es decir, acceso múltiple con sensado de portadora y detección de colisiones, es un algoritmo de control de acceso a un medio compartido.

Utiliza el sensado de portadora para determinar si hay nodos transmitiendo. Cuando un host tiene datos para enviar, sensa el medio compartido:

- Si el medio está libre, el host transmite.
- Si el medio está ocupado, no puede enviar porque habría una colisión. Entonces debe esperar a que el medio se libere:
  - Si el algoritmo es **1-persistente**, el host comienza a transmitir apenas se libere el medio.
  - Si es **p-persistente**, el host espera a que se libere el medio y transmite con probabilidad p. El uso de un componente azaroso (en la p-persistencia) tiene sentido porque si hay varios hosts esperando a que se libere el medio, y todos intentan transmitir ni bien éste se libera, va a ocurrir una colisión. Imponer una probabilidad para transmitir reduce las probabilidades de colisiones.

Este algoritmo es de categoría half-duplex: La lógica de recepción está establecida en el sensado para detectar colisiones. Es decir, no se puede enviar y recibir a la vez (eso sería full-duplex).

Supongamos ahora que ocurre una colisión, entonces el tiempo se divide en slots de tamaño 2τ. Después de la primera colisión, cada estación espera 0 o 1 slots de tiempo al azar. Si dos estaciones colisionan y eligen el mismo número, cada una elige entre 0, 1, 2 ó 3 tiempos al azar y espera el número de slots de tiempos elegidos antes de volver a intentar transmitir.

En general, después de la i-ésima colisión, cada host debe elegir un número entre 0 y $2^i - 1$ que será la cantidad de slots de tiempo que debe dejar pasar antes de volver a intentar la transmición.

Este algoritmo se llama **Exponential Backoff Binario**, sirve para adaptar de manera dinámica el número de estaciones que están tratando de emitir simultaneamente. Si el intervalo de elección para todas las elecciones fuese 1023, las chances de colisión son despreciables.

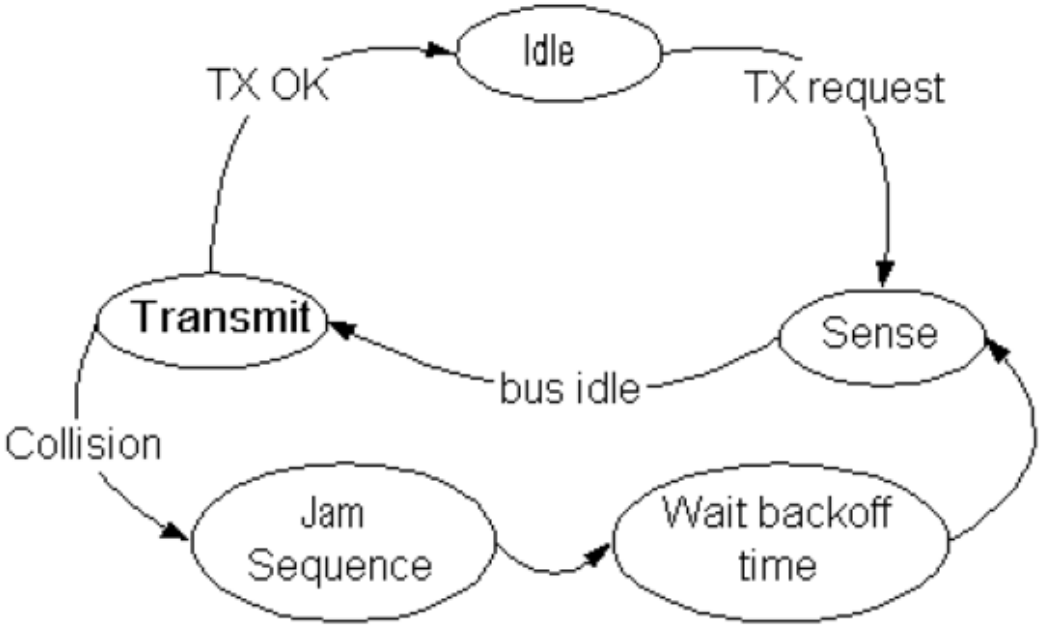

Figura 13: Topologías de Red

5.3. Logical Link Control

Es la subcapa de la capa de enlace que se encarga de procesar el paquete recibido en la capa MAC y enviarlo al proceso correspondiente en la capa de Red.

6. Redes inalámbricas: Wi-Fi: IEEE 802.11b/g/n

En las redes inalámbricas, los frames se mandan a través de ondas en una determinada frecuencia, que depende de la tecnología usada para transmitirlas. En este tipo de red, la intensidad de la señal disminuye con la distancia y tiene fuentes de ruidos más impredecibles que en medios guiados. Esto se traduce en una tasa de errores elevadas.

6.1. Spread Spectrum

Esto sumado a que cualquier dispositivo cercano puede conectarse a ellas, impulsaron el diseño de técnicas para aprovechar al máximo el ancho de banda de tal forma de proteger a la red de intrusos.

Spread Spectrum es una técnica qu modula la señal usando una moduladora semi random compuesta de unos (1) y menos unos (-1) conocida por los dispositivos de la red qur aplican a la señal que quieren enviar. Cuando la señal llega el receptor, este realiza el proceso inverso para conseguir la señal original.

Otra forma de realizar esto, es enviar la información transmitida en un rango de frecuencias que es cambiado varías veces durante el proceso de transmición. En este método, la información original se divide en partes más pequeñas usando un patron conocido solamente por el transmisor y el receptor.

6.2. Carrier Sense Multiple Access / Collision Avoidance (CSMA/CA)

Dado que las redes inalámbricas son un medio de broadcasting, los equipos de la red debe estar atentos a multiples transmiciones realizadas simultaneamente. A diferencia de los medios guiados, implementar una comunicación full-duplex sobre radio frecuencias es costoso por lo que se intenta evitar las colisiones en vez de resolverlas.

Para manejar esto, se utiliza el sistema CSMA descripto en la sección 5.2 pero adaptado a los problemas introducidos por este tipo de red

6.2.1. Colission Avoidance

Antes de transmitir, una estación debe determinar el estado del medio. Si el canal no está ocupado, realiza una espera adicional llamada **espaciado entre tramas** para asegurarse de que no colisione. Si durante esta espera, el medio no permanece libre, entonces suspende la transmición hasta que se cumpla dicha condición.

Una vez que transmite la trama, espera recibir un ACK. Si esto no sucede, asume que la misma se perdió en una colisión y la retransmitirá.

Una vez que un host transmite una trama exitosamente, espera un **período de contención** (cuantificado por un back-off) para minimizar la probabilidad de colisiones. Si durante este período, el medio se libera, el host puede transmitir una nueva trama. Si no, vuelve a esperar un nuevo período de contención.

Este método recién descripto es llamado **Distributed Coordination Function (DCF)** y es el método de acceso al medio más básico de 802.11.

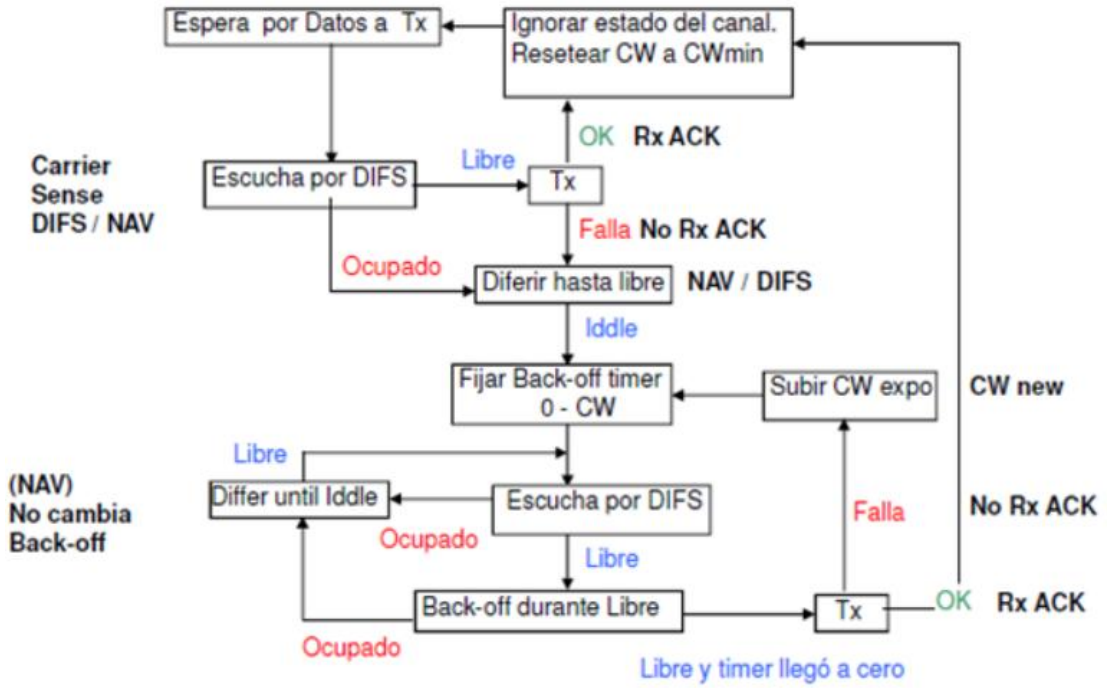

Figura 14: Máquina de estados de un host en una red Wifi

6.2.2. Problema de la estación oculta

Supongamos que la computadora A comienza a transmitir a B.

Si C detecta el medio no escuchará a A porque está fuera de su alcance, y por lo tanto deducirá erroneamente que puede transmitir. Si lo hace, interferirá en B eliminando la trama de A.

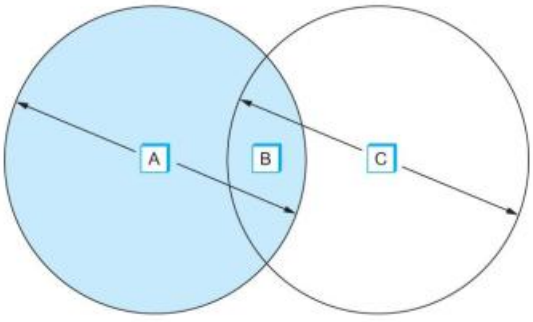

Figura 15: Problema de la estación oculta

6.2.3. Problema de la estación expuesta

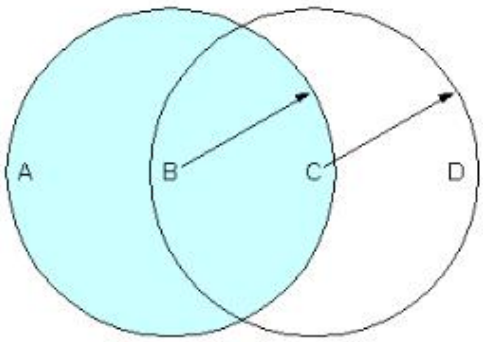

Figura 16: Problema de la estación expuesta

Supongamos ahora que A está transmitiendo una tramba a B. Supongamos ahora que C quiere transmitir a D. Cuando detecta al medio, escuhará una transmición y concluirá que no puede realizar su envio. Sin embargo, esa transmición causaría una mala recepción solo en la zona entre B y C, en la que no está localizado ninguno de los receptores pretendidos.

Parte IV
Nivel de Red

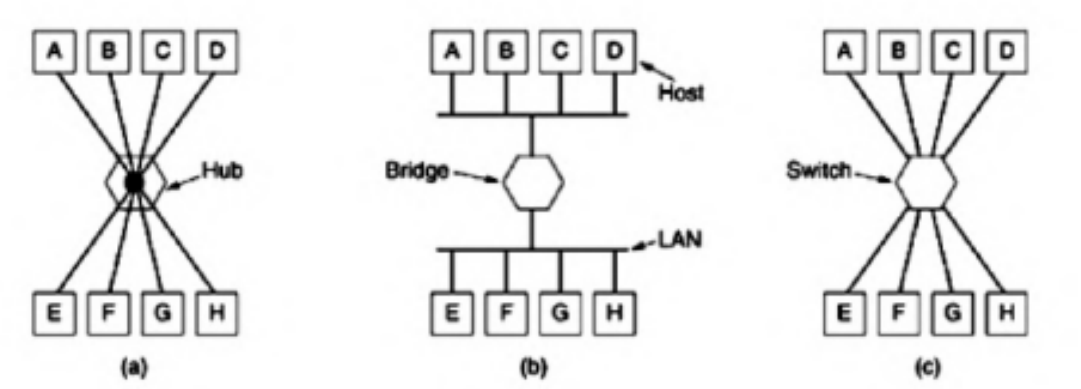

Figura 17: Topologías de Red

En la figura 17 se pueden ver tres tipos de LAN:

a) En la primera vemos un conjunto de hosts conectados por un **hub**, esto es: Todos los hosts están conectados entre sí como si los uniese un único cable. El **hub**, como los **repetidores** (amplificadores de señal), funcionan a nivel físico permitiendo agregar equipos a una red.

b) En el segundo, un puente o **bridge**, separa en dos grupos los hosts de la red creando dos zonas independientes para la detección de colisiones. En este caso los hosts de un grupo podrán emitir sin tener que preocuparse por interferir con los hosts del otro grupo. Un paquete enviado la dirección de **broadcast**, alcanza a todos los hosts de la LAN.

   Puede interconectar dos tipos de tecnología distintas (por ejemplo, ethernet y wifi). Por esta razón en la capa MAC, se agrega al header del paquete un campo que permite identificar el tipo de red del que viene. Cuando el bridge capta el paquete y se da cuenta que el host está en una red de distinto tipo, cambia el header para que matcheen y lo reenvía a través de la tecnología correspondiente

c) Por último tenemos un conjunto de host conectados por un **switch**: Cada host se conecta con una conexión full-duplex al dispositivo, que se encarga de recibir todos los mensajes de un host y redireccionarlo al host correspondiente. En este tipo de redes se elimina la necesidad detectar colisiones y se pasa a necesitar un algoritmo que permita al switch realizar el dispatch de los mensajes de manera correcta.

   Tanto el switch como el bridge funciona a nivel capa de enlace. Se encargan de redireccionar los frames enviados a través de una LAN para que lleguen al dispositivo correspondiente.

Agregamos un último dispositivo: El **router**, para conectar distintas redes a nivel red. Se encargán de buscar el camino que debe seguir para llegar a destino. Cuando una host decide enviar un paquete, lo encapsula en un frame que contiene el tipo de protocolo usado. Cuando el paquete llega el router, este se encarga de tomar el frame que le llego, desencapsular el paquete y guardarlo en un frame adecuado para que pueda ser interpretado por los dispositivos de la nueva red en la que ingresa el frame.

7. Switches

En términos simples, un switch es un mecanismo que nos permite interconectar enlaces para formar una red más grande. Un switch es un dispositivo de múltiples entradas y salidas que transfiere paquetes desde una entrada a una o más salidas. El trabajo principal de un switch es recibir paquetes entrantes en uno de sus enlaces y transmitirlos en otro enlace. Esta función se denomina a veces conmutación (**switching**) o reenvío (**forwarding**).

Por lo tanto, un switch agrega la topología de estrella a la topología de enlace punto a punto. Una topología de estrella tiene varias propiedades atractivas:

- Aunque un switch tiene un número fijo de entradas y salidas, lo que limita la cantidad de hosts que se pueden conectar a un solo switch, se pueden construir redes grandes interconectando varios switches.
- Podemos conectar switches entre sí y a hosts usando enlaces punto a punto, lo que típicamente significa que podemos construir redes de gran alcance geográfico.
- Agregar un nuevo host a la red conectándolo a un switch no necesariamente reduce el rendimiento de la red para otros hosts ya conectados.

Cada host en una red conmutada (Switched network) tiene su propio enlace al switch, por lo que puede ser completamente posible que muchos hosts transmitan a la velocidad de enlace completa (ancho de banda), siempre que el switch esté diseñado con suficiente capacidad agregada.

En general, las redes conmutadas son consideradas más escalables.

Cuando un router recibe un paquete, busca un identificador en el header del paquete que usa para decidir a que puerto redirigirlo. Los detalles de como usa este identificador varían, pero hay dos enfoques comunes. El primero es el enfoque **datagrama** o **sin conexión**. El segundo es el enfoque de **circuito virtual** o **orientado a conexión**. Un tercer enfoque, el **enrutamiento de origen** (**source routing**), es menos común que los otros dos pero tiene algunas aplicaciones útiles.

Una cosa que es común a todas las redes es que necesitamos tener una forma de identificar los nodos con los que podremos comunicarnos. El identificador que se les asigna se llama **direccion**.

Otra cosa que debemos definir es como el router va a tomar la decisión de a que puerto enviar el paquete. Para esto, se define una **tabla de enrutamiento** que asocia una dirección de

destino con un puerto de salida. La tabla de enrutamiento se construye a partir de un algoritmo de enrutamiento.

7.1. Redes de datagramas

La idea detrás de los datagramas es incluir en cada paquete suficiente información para permitir que cualquier switch pueda decidir cómo llevarlo a su destino. Es decir, cada paquete contiene la dirección de destino completa. Para decidir cómo reenviar un paquete, un switch consulta una tabla de forwarding (a veces llamada tabla de enrutamiento).

Las redes de datagramas tienen las siguientes características:

- Un host puede enviar un paquete a cualquier destino en cualquier momento, ya que cualquier paquete que llegue a un switch puede ser reenviado inmediatamente (suponiendo una tabla de forwarding correctamente armada). Por esta razón, las redes de datagramas a menudo se llaman sin conexión; esto contrasta con las redes orientadas a conexión en las que se debe establecer algún estado de conexión antes de que se envíe el primer paquete de datos.
- Cuando un host envía un paquete, no tiene forma de saber si la red es capaz de entregarlo ni si el host de destino está activo.
- Cada paquete se reenvía independientemente de los paquetes anteriores que podrían haberse enviado al mismo destino. Por lo tanto, dos paquetes sucesivos del host A al host B pueden seguir rutas completamente diferentes (quizás debido a un cambio en la tabla de forwarding en algún switch de la red).
- Una falla de switch o enlace puede no tener ningún efecto serio en la comunicación si es posible encontrar una ruta alternativa alrededor de la falla y actualizar la tabla de forwarding en consecuencia.

7.2. Circuitos virtuales

Una segunda técnica para el conmutación de paquetes, que difiere significativamente del modelo de datagrama, usa el concepto de un **circuito virtual (VC)**.

Este enfoque, que también se conoce como **modelo orientado a conexión**, requiere configurar una conexión virtual desde el host de origen hasta el host de destino antes de que se envíe cualquier dato. Podemos pensar en esto como un proceso de dos etapas. La primera etapa es la "configuración de la conexión". El segundo es la transferencia de datos.

Durante el setup de la conexión, se establece un estado de conexión en cada uno de los switches entre los hosts de origen y destino. Una sola conexión consiste en una entrada en la "tabla VC"de cada switch que describe la conexión y que interfaces dentro de ese switch están involucradas en la misma. Entonces cada entrada de la tabla VC está formada por cuatro campos:

- Un **identificador de circuito virtual (VCI)** que identifica de forma única la conexión en este switch y que se llevará dentro de la cabecera de los paquetes que pertenecen a esta conexión
- Una **interfaz de entrada** en la que el switch va a recibir los paquetes para este VC.
- Una **interfaz de salida** que se usará para reenviar los paquetes asociados al VC.
- El **VCI de salida** que se utilizará en el nuevo header de los paquetes que se forwardean.

VCI de entrada	Interfaz de entrada	Interfaz de salida	VCI de salida
x	i	o	y

Si un paquete llega a la interfaz de entrada i y ese paquete contiene el valor VCI x en su encabezado, entonces ese paquete debe enviarse a la interfaz de salida o con el valor y como VCI de salida (se remplaza el VCI de entrada en el header del paquete recibido por el nuevo antes de forwardearlo).

La combinación del VCI de los paquetes a medida que se reciben en el switch y la interfaz de entrada identifican de forma única la conexión virtual. Puede haber muchas conexiones virtuales establecidas en un mismo switch al mismo tiempo. Además, los valores de VCI de entrada y salida generalmente no son los mismos. Por lo que el VCI no es un identificador de importancia global para la conexión; más bien, tiene importancia solo en un enlace dado.

Cada vez que se crea una nueva conexión, debemos asignar un nuevo VCI para esa conexión en cada enlace que la conexión atravesará. También debemos asegurarnos de que el VCI elegido en un enlace determinado no esté actualmente en uso en ese enlace por alguna conexión existente.

7.2.1. Creación de la conexión

Hay dos enfoques generales para establecer el estado de conexión: Uno es que un administrador de red configure el estado, en cuyo caso el circuito virtual es "permanente"(**Permanent Virtual Cirtcuit**). Este circuito también puede ser eliminado por el administrador. Alternativamente, un host puede enviar mensajes a la red para hacer que se establezca el estado. Esto se conoce como **señalización (signaling)**, y los circuitos virtuales resultantes se denominan conmutados (switched). La característica más importante de un **circuito virtual conmutado (SVC)** es que un host puede configurarlo y eliminarlo dinámicamente sin la participación de un administrador de red.

Suponiendo que todos los nodos de la red saben la topología y la ubicación de cada nodo (esto se hace corriendo algoritmos de routing que vamos a ver más adelante), el algoritmo para crear una conexión es el siguiente: Si el host A quiere conectarse con el host B entonces:

1. A envía un paquete con un pedido de conexión a la red que contiene la dirección completa de B.
2. El paquete llega a un switch S_1 que crea una entrada en su tabla de VC para la conexión registrando la interfaz desde la que le llegó el paquete y asignando el VCI de entrada. Luego busca en su tabla de forwarding cual es el próximo nodo que debe recibir el paquete.

3. Cada switch al que le llega el paquete realiza la misma operación hasta que el mismo llega a B.

4. B decide si aceptar o no la conexion. Si lo hace, también asigna un valor VCI de entrada en su tabla de VC.

5. Para completar la conexión, B envía un mensaje de aceptación que debe recorrer el camino establecido hasta A indicando a cada uno de los nodos cual es el VCI de salida que deben usar en cada paso.

6. Cuando el último paquete llega a A, cada switch tiene una entrada completa en su tabla de VC para la conexión.

Cuando A ya no quiere enviar datos a B, destruye la conexión enviando un mensaje de destrucción a S_1. El switch elimina la entrada relevante de su tabla y reenvía el mensaje a los otros switches en el camino, que eliminan de manera similar las entradas apropiadas de sus tablas.

Hay varias cosas a tener en cuenta sobre el conmutación de circuitos virtuales:

- A tiene que esperar a que la solicitud de conexión llegue al otro lado de la red y regrese antes de que pueda enviar su primer paquete de datos, hay al menos un tiempo de ida y vuelta (RTT) de retraso antes de que esto pueda suceder.

- Si bien la solicitud de conexión contiene la dirección completa de B (que puede ser bastante grande, ya que es un identificador global en la red), cada paquete de datos contiene solo un identificador pequeño, que es único solo en un enlace. Por lo tanto, la sobrecarga por paquete causada por el encabezado se reduce en relación con el modelo de datagrama.

- Si un switch o un enlace en una conexión falla, la conexión se rompe y se deberá establecer una nueva. Además, el circuito virtual roto debe ser desmantelado para liberar espacio de almacenamiento de tablas de los switches.

7.2.2. Comparación con el modelo de datagrama

Una de las ventajas de los circuitos virtuales es que para cuando el host recibe la señal de que puede enviar datos, sabe bastante sobre la red, por ejemplo, que realmente hay una ruta hacia el receptor y que el receptor está dispuesto y es capaz de recibir datos. También es posible asignar recursos al circuito virtual en el momento en que se establece.

En comparación, una red de datagramas no tiene una fase de establecimiento de conexión, y cada switch procesa cada paquete de forma independiente, lo que hace menos obvio cómo una red de datagramas asignaría recursos de manera significativa. En cambio, cada paquete que llega compite con todos los demás por el espacio del búfer. Si no hay búferes libres, el paquete entrante debe descartarse.

7.3. Source Routing

Un tercer enfoque al switching que no utiliza ni circuitos virtuales ni datagramas convencionales se conoce como **source routing**. En esta técnica, toda la información sobre la topología de la red que se requiere para conmutar un paquete es proporcionada por el host de origen.

Es decir, el host de origen envía un paquete con un header que contiene toda la información necesaria para forwardear el paquete a su destino. El switch simplemente lee el header y lo reenvía al siguiente nodo. Una forma de hacer esto sería poner una lista ordenada de puertos de switch por el que se debe forwardear el paquete y rotar la lista para que el siguiente switch en el camino siempre esté al frente de la lista.

Hay varias cosas a tener en cuenta sobre este enfoque:

- Asume que el host de origen sabe lo suficiente sobre la topología de la red para formar un encabezado que tenga todas las direcciones correctas para cada switch en el camino.
- No podemos predecir cuán grande debe ser el encabezado, ya que debe poder contener una palabra de información para cada switch en el camino. Esto implica que probablemente sean de longitud variable sin límite superior, a menos que podamos predecir con absoluta certeza el número máximo de switches a través de los cuales un paquete necesitará pasar.
- Hay algunas variaciones en este enfoque. Por ejemplo, en lugar de rotar el encabezado, cada switch podría simplemente eliminar el primer elemento a medida que lo usa. La rotación tiene una ventaja sobre el stripping, sin embargo: el host B obtiene una copia del encabezado completo, lo que puede ayudarlo a descubrir cómo volver al host A. Otra alternativa es que el encabezado lleve un puntero al "próximo puerto"que debe ser usado, de modo que cada switch simplemente actualice el puntero en lugar de rotar el encabezado; esto puede ser más eficiente de implementar.

El source routing se puede usar tanto en redes de datagramas como en redes de circuitos virtuales. Por ejemplo, el Protocolo de Internet (IP), que es un protocolo de datagramas, incluye una opción de source routing que permite que se enruten los paquetes seleccionados, mientras que la mayoría se conmutan como datagramas convencionales.

7.4. Bridges and LAN Switches

Comenzamos considerando una clase de switch que se usa para reenviar paquetes entre LAN (redes de área local) como Ethernets. Dichos switches a veces se conocen con el nombre de **LAN switches**; históricamente, también se los ha denominado **bridges**, y se usan ampliamente en redes de campus y empresas.

Este tipo de nodo implementan completamente la detección de colisiones y los protocolos de acceso al medio en cada una de sus interfaces. Por lo tanto, las restricciones del protocolo sobre la distancia y número de hosts no se aplicarían al par de redes combinadas de esta manera. Este dispositivo opera en modo promiscuo, aceptando todos los tramas transmitidos en cualquiera de las redes y reenviándolos al otro.

Una colección de redes conectadas por uno o más bridges se dice que forman una **LAN extendida**.

7.4.1. Learning Bridges

Para esto podemos hacer que cada bridge inspeccione la dirección de origen de todos los frames que recibe. Cuando un host A envía un frame a otro cualquiera, el bridge recibe este frame y registra el hecho de que se acaba de recibir un frame de A en la interfaz I. De esta manera, cuando reciba un mensaje que debe ser envíado a A lo manda directamente a la interfaz I.

Cuando un bridge se inicia, esta tabla está vacía. Las entradas se van agregan y borrando a medida que pasa el tiempo. Las entradas tienen asociado un timeout después del cual debe ser descartada para evitar que se mantenga una que se quedó invalidada si un host, y como consecuencia, su dirección LAN, se mueve de una red a otra. Por lo tanto, esta tabla no es necesariamente completa. Si el bridge recibe un frame que está dirigido a un host que no está actualmente en la tabla reenvía el frame en todos los demás puertos.

7.4.2. Spanning Tree Protocol

La estrategia anterior funciona bien hasta que la LAN extendida tiene un ciclo, en cuyo caso falla miserablemente: los frames potencialmente se repiten a través de la LAN extendida eternamente.

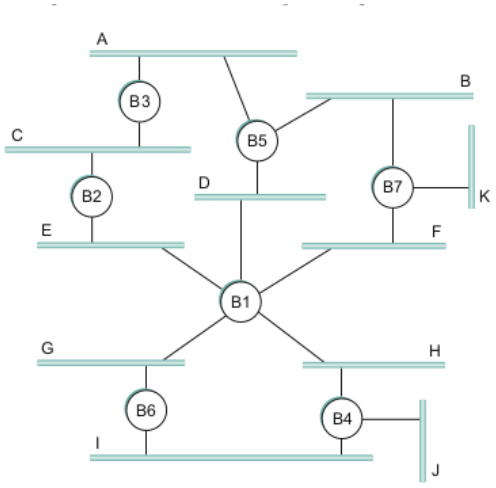

Figura 18: Red con un ciclo

Este problema se resuelve haciendo que los bridges ejecuten un algoritmo de **spaning tree** distribuido. Se piensa en la LAN extendida como si estuviera representada por un grafo que posiblemente tenga ciclos. Un spanning tree es un subgrafo de este grafo que cubre todos los

vértices de la red pero no contiene ciclos. Es decir, un spanning tree mantiene todos los vértices del grafo original pero descarta algunas de las aristas.

El siguiente protocolo es utilizado por un conjunto de bridges para acordar el spanning tree para una LAN extendida en particular y funciona de la siguiente manera: Cada bridge tiene un identificador único; para nuestros propósitos, usamos las etiquetas B_1, B_2, B_3, y así sucesivamente.

Los bridges tienen que intercambiar mensajes de configuración entre sí y luego decidir si son el root o un bridge designado en base a estos mensajes. Específicamente, estos mensajes contienen tres piezas de información:

1. El ID del bridge que está enviando el mensaje
2. El ID del bridge que es considerado el root (hasta el momento)
3. La distancia, medida en saltos (hops), desde el bridge que envía hasta el root

Inicialmente, cada bridge piensa que es la raíz, y por lo tanto envía un mensaje en cada uno de sus puertos identificándose como la raíz y dando una distancia de 0.

Al recibir un mensaje sobre un puerto en particular, el bridge verifica si ese nuevo mensaje es mejor que el ya almacenado para ese puerto. Un mensaje se considera mejor que la información actualmente grabada si se cumple alguna de las siguientes condiciones:

- Identifica una raíz con un ID más pequeño.
- Identifica una raíz con un ID igual pero con una distancia más corta.
- El ID de la raíz y la distancia son iguales, pero el bridge que envía tiene un ID más pequeño.

Si el nuevo mensaje es mejor que la información actualmente grabada, el bridge descarta la información antigua y guarda la nueva. Sin embargo, primero agrega 1 a la distancia ya que el bridge está a un salto más lejos de la raíz que el bridge que envió el mensaje.

Cuando un bridge recibe un mensaje que indica que no es la raíz, es decir, un mensaje de un bridge con un ID más pequeño, el mismo deja de generar mensajes por su cuenta y en su lugar solo reenvía los de otros bridges, después de agregar 1 al campo de distancia. Del mismo modo, cuando recibe un mensaje que indica que no es el designado para ese puerto, deja de generar mensajes sobre ese puerto. Por lo tanto, cuando el sistema se estabiliza, solo el bridge raíz sigue generando mensajes, y los otros están reenviando estos mensajes solo sobre los puertos para los que son el bridge designado. En este punto, se ha construido un spanning tree, y todos los bridges están de acuerdo en qué puertos se utilizan en el mismo. Solo esos puertos pueden usarse para reenviar paquetes de datos en la LAN extendida.

En cierto sentido, es eliminando puertos de la topología que la LAN extendida se reduce a un árbol acíclico. Incluso es posible que un bridge no participe en el reenvío de frames, lo que parece un poco extraño a primera vista. Sin embargo, el algoritmo es dinámico, lo que significa que los bridges siempre están preparados para reconfigurarse en un nuevo spanning tree si alguno

falla, y por lo tanto, esos puertos y bridges no utilizados proporcionan la capacidad redundante necesaria para recuperarse de fallas.

Una cosa importante a tener en cuenta es que, aunque el algoritmo es capaz de reconfigurar el árbol de expansión cada vez que falla un bridge, no es capaz de reenviar frames por rutas alternativas para evitar un bridge congestionado.

Dado que el objetivo de un bridge es extender una LAN de manera transparente a través de múltiples redes, y dado que la mayoría de las LAN admiten tanto broadcast como multicast, los bridges también deben admitir estas dos características. Broadcast es simple: cada bridge reenvía un frame con una dirección de destino de broadcast en cada puerto activo que no sea el puerto por el que se recibió el frame.

Multicast se puede implementar de la misma manera, con cada host decidiendo por sí mismo si aceptar o no el mensaje.

El algoritmo del árbol de expansión se puede extender para podar las redes sobre las cuales no es necesario reenviar los frames multicast. Cada host que es miembro del grupo M debe periódicamente enviar un frame con la dirección para el grupo M en el campo de origen del encabezado del frame. Este frame tendría como dirección de destino la dirección multicast para los bridges. Sin embargo, esto es ineficiente, por lo que en general se usa la primera opción.

7.4.3. Limitaciones de los bridges

En cuanto a la escala, no es realista conectar más de unas pocas LAN mediante bridges. Una razón para esto es que el algoritmo del spanning tree escala linealmente. Una segunda razón es que los bridges reenvían todos los frames de broadcast.

Un enfoque para aumentar la escalabilidad de las LAN extendidas es la VLAN. Las VLAN permiten que una sola LAN extendida se divida en varias LAN separadas. A cada LAN virtual se le asigna un identificador (a veces llamado color), y los paquetes solo pueden viajar de un segmento a otro si ambos segmentos tienen el mismo identificador. Esto tiene el efecto de limitar el número de segmentos en una LAN extendida que recibirá cualquier paquete de broadcast.

En cuanto a la heterogeneidad, los bridges son bastante limitados en los tipos de redes que pueden interconectar. En particular, los bridges utilizan el header del frame de la red y, por lo tanto, solo pueden admitir redes que tengan exactamente el mismo formato para las direcciones. Por lo tanto, los bridges se pueden usar para conectar Ethernets a Ethernets, anillos de tokens a anillos de tokens y una red 802.11 a otra. También es posible poner un bridge entre, por ejemplo, un Ethernet y una red 802.11, ya que ambas redes admiten el mismo formato de dirección de 48 bits. Sin embargo, los bridges no se generalizan fácilmente a otros tipos de redes con diferentes formatos de direccionamiento.

8. IP

Utilizamos el término **internet**, o a veces simplemente **internet** con una i minúscula, para referirnos a una colección arbitraria de redes interconectadas para proporcionar algún tipo de servicio de entrega de paquetes de host a host.

Una internet conecta diversas redes através de routers. Es fácil confundirse con la distinción entre bridges, switches y routers. Hay una buena razón para tal confusión, ya que en algún nivel todos reenvían mensajes de un enlace a otro. Una distinción que la gente hace se basa en la estratificación: los bridges son nodos de nivel de enlace (reenvían frames de un enlace a otro para implementar una LAN extendida), los switches son nodos de nivel de red (reenvían paquetes de un enlace a otro para implementar una red conmutada por paquetes) y los routers son nodos de nivel de internet (reenvían datagramas de una red a otra para implementar un internet).

El **Protocolo de Internet (Internet Protocol - IP)** es la herramienta clave utilizada hoy en día para construir internets escalables y heterogéneas. La filosofía utilizada en la definición del modelo de servicio IP fue hacerlo poco exigente para que casi cualquier tecnología de red que pueda aparecer en una internet pueda proporcionar el servicio necesario.

El modelo de servicio IP puede pensarse en dos partes: un esquema de direccionamiento, que proporciona una forma de identificar todos los hosts en la internet, y un modelo de datagrama (sin conexión) de entrega de datos. Este modelo de servicio a veces se llama mejor esfuerzo (**best effort**) porque, aunque IP hace todo lo posible para entregar datagramas, no ofrece garantías.

Cada datagrama lleva suficiente información para permitir que la red reenvíe el paquete a su destino correcto; no es necesario ningún mecanismo de configuración previa para decirle a la red qué hacer cuando llega el paquete. Simplemente lo envía y la red hace todo lo posible para llevarlo al destino deseado. La parte de "mejor esfuerzo"significa que si algo sale mal y el paquete se pierde, se corrompe, se entrega incorrectamente o de cualquier manera no llega a su destino previsto, la red no hace nada: hizo su mejor esfuerzo y eso es todo lo que tiene que hacer. No hace ningún intento de recuperarse del fallo. Esto a veces se llama un servicio no confiable.

La entrega de mejor esfuerzo no significa solo que los paquetes pueden perderse. A veces pueden entregarse fuera de orden, y a veces el mismo paquete puede entregarse más de una vez. Los protocolos de nivel superior o las aplicaciones que se ejecutan por encima de IP deben ser conscientes de todos estos posibles tipos de falla.

8.1. Formato del paquete

Una parte clave del modelo de servicio IP es el tipo de paquetes que se pueden transportar. El datagrama IP, como la mayoría de los paquetes, consiste en una cabecera seguida de un número de bytes de datos.

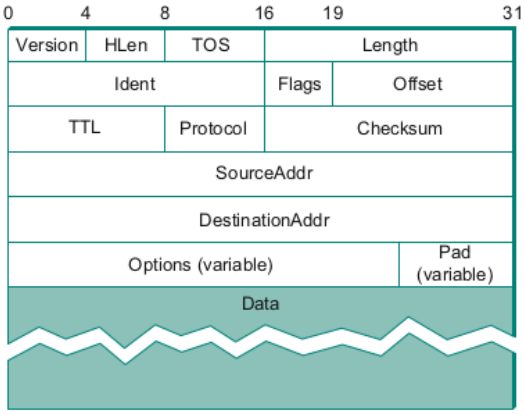

Figura 19: Paquete IP

El header del paquete consta de los siguientes campos:

- El campo **Version** especifica la versión de IP. La versión actual de IP es 4, y a veces se llama IPv4. Colocar este campo justo al comienzo del datagrama facilita que todo lo demás en el formato del paquete se redefina en versiones posteriores.
- El campo **HLEN** especifica la longitud del encabezado en palabras de 32 bits. Cuando no hay opciones, lo que sucede en la mayoía de los casos, el encabezado tiene 5 palabras (20 bytes) de largo.
- El campo **TOS (tipo de servicio)** de 8 bits ha tenido una serie de definiciones diferentes a lo largo de los años, pero su función básica es permitir que los paquetes se traten de manera diferente según las necesidades de la aplicación.
- El campo **Length** (16 bits) contienen la longitud del datagrama, incluido el encabezado. A diferencia del campo HLen, el campo Longitud cuenta bytes en lugar de palabras. Por lo tanto, el tamaño máximo de un datagrama IP es de 65.535 bytes. Sin embargo, la red física a través de la cual se ejecuta IP puede no admitir paquetes tan largos. Por esta razón, IP admite un proceso de fragmentación y reensamblaje.
- Los campos **Ident**, **Flags** y **Offset** contienen información sobre la fragmentación, que se describe en la siguiente sección.
- El campo **TTL** es el tiempo de vida del paquete. La intención del campo es capturar paquetes que han estado dando vueltas en bucles de enrutamiento y descartarlos, en lugar de dejar que consuman recursos indefinidamente.

  Dado que no todos los enrutadores no tienen acceso a un reloj común, la mayoría de los enrutadores simplemente decrementaban el TTL en 1 a medida que envian el paquete. Por lo tanto, se convirtió más en un recuento de saltos que en un temporizador.

Establecer este campo demasiado alto haría que los paquetes circulen bastante antes de ser descartados; Establecerlo demasiado bajo podría conllevar a que no lleguen a su destino. El valor 64 es el valor predeterminado actual.

- El campo **protocolo** es simplemente una clave de demultiplexión que identifica el protocolo de nivel superior al que se debe pasar este paquete IP. Hay valores definidos para TCP (Protocolo de control de transmisión - 6), UDP (Protocolo de datagramas de usuario - 17) y muchos otros protocolos.
- El campo **Checksum** es un campo de detección de errores de 16 bits que cubre tanto el encabezado como los datos. Sirve para detectar errores, no así para corregirlos. Por lo tanto, si el host detecta un error en el paquete, lo descarta.
- El campo **SourceAddr** es la dirección completa del host que envía el paquete. Es necesaria para permitir que los destinatarios decidan si desean aceptar el paquete y para permitirles responder.
- El campo **DestinationAddr** es la dirección completa del host que debe recibir el paquete. Es la clave para la entrega de datagramas.
- Finalmente, puede haber una serie de opciones al final del encabezado. La presencia o ausencia de opciones se puede determinar examinando el campo HLen. Si bien las opciones se usan con bastante poca frecuencia, una implementación IP completa debe poder procesarlas todas.

8.2. Fragmentación y reensambe

Cada tipo de red tiene una **unidad de transmisión máxima (Maximum Transmission Unit - MTU)**, que es el datagrama IP más grande que puede transportar en un frame.

Cuando un host envía un datagrama IP, puede elegir cualquier tamaño que desee. Una opción razonable es el MTU de la red a la que está directamente conectado. En este caso, la fragmentación solo será necesaria si la ruta hacia el destino incluye una red con un MTU más pequeño. Si el protocolo de transporte que se encuentra en la parte superior de IP le da un paquete más grande que el MTU local, entonces el host de origen debe fragmentarlo.

Para permitir que estos fragmentos se vuelvan a ensamblar en el host de recepción, todos llevan el mismo identificador en el campo `Ident`. Este identificador es elegido por el host de envío y se elige de tal forma que sea único entre todos los datagramas que puedan llegar al destino desde esta fuente durante un período de tiempo razonable. Dado que todos los fragmentos del datagrama original contienen este identificador, el host de reensambe podrá reconocer aquellos fragmentos que van juntos. Si no llegan todos los fragmentos al host de recepción, el host abandona el proceso de reensamble y descarta los fragmentos que llegaron. IP no intenta recuperarse de fragmentos faltantes.

Supongamos que hay que fragmentar un mensaje recibido, entonces se divide el datgrama en paquetes I más pequeños y en el primer paquete se establece el bit `M` en el campo `Flags`, lo que

significa que hay más fragmentos por seguir, y establece el `Offset` en 0, ya que este fragmento contiene la primera parte del datagrama original.

Cuando el host recibe todos los fragmentso, usa el offset de cada paquete para reordenarlos y luego los concatena para formar el datagrama completo. El reensamblaje se realiza en el host receptor y no en cada enrutador.

8.3. Direcciones globales

Las direcciones Ethernet son globalmente únicas pero eso solo no es suficiente para un esquema de direccionamiento en una gran internet. Las direcciones Ethernet también son planas, lo que significa que no tienen estructura y proporcionan muy pocas pistas para los protocolos de enrutamiento.

En contraste, las direcciones IP son jerárquicas: Están compuestas de varias partes que corresponden a algún tipo de jerarquía en la internet. Específicamente, las direcciones IP consisten en dos partes denominadas parte de red y parte de host. La parte de red de una dirección IP identifica la red a la que está conectado el host; todos los hosts conectados a la misma red tienen la misma parte de red en su dirección IP. La parte del host identifica de forma única cada host en esa red en particular.

Los router que se conectan a dos o más redes necesitan tener una dirección en cada red, una para cada interfaz. Por lo tanto, teniendo en cuenta que un enrutador podría implementarse como un host con dos interfaces de red, es más preciso pensar en las direcciones IP como pertenecientes a interfaces que a hosts.

La clase de una dirección IP se identifica en los bits más significativos. Si el primer bit es 0, es una dirección de clase A. Si el primer bit es 1 y el segundo es 0, es una dirección de clase B. Si los dos primeros bits son 1 y el tercero es 0, es una dirección de clase C.

Class	Leading bits	Size of *network number* bit field	Size of *rest* bit field	Number of networks	Addresses per network	Start address	End address
A	0	8	24	128 (2^7)	16,777,216 (2^{24})	0.0.0.0	127.255.255.255
B	10	16	16	16,384 (2^{14})	65,536 (2^{16})	128.0.0.0	191.255.255.255
C	110	24	8	2,097,152 (2^{21})	256 (2^8)	192.0.0.0	223.255.255.255

Figura 20: Clases de direcciones IP

Por convención, las direcciones IP se escriben como cuatro enteros decimales separados por puntos. Cada entero representa el valor decimal contenido en 1 byte de la dirección, comenzando por el bit más significativo.

8.3.1. Forwarding de datagramas en IP

Un datagrama se envía desde un host de origen a un host de destino, posiblemente pasando por varios routers en el camino. Cualquier nodo, ya sea un host o un enrutador, primero intenta establecer si está conectado a la misma red física que el destino. Para hacer esto, compara la parte de red de la dirección de destino con la parte de red de la dirección de cada una de sus interfaces de red.

Si hay alguna una coincidencia significa que el destino se encuentra en la misma red física que la interfaz y el paquete se puede entregar directamente a través de esa red.

Si el nodo no está conectado a la misma red física que el nodo de destino entonces debe elegir a quien debe pasar el datagrama para acercarlo a su destino. En general, cada nodo tendrá una **tabla de forwarding** que enumera los posibles routers a los que pueda reenviar el paquete por lo que debe elegir el mejor, o al menos uno que tenga una posibilidad razonable de realizar esta tareas. El router que termina eligiendo se conoce como **next hop**. Cada fila de la tabla es un par $< NetworkNum, NextHop >$.

En caso de no encontrar una opción que considere viable, el paquete se envía a un router default almacenado en la tabla.

8.4. Subnetting

Subnetting proporciona un primer paso para reducir el número total de números de red que se asignan a una tabla de forwarding. La idea es tomar un solo número de red IP y asignar las direcciones IP con ese número de red a varias redes físicas, que ahora se denominan subredes. Varias cosas deben hacerse para que esto funcione. Primero, las subredes deben estar cerca unas de otras. Esto se debe a que en un punto distante de Internet, todas parecerán una sola red, teniendo solo un número de red entre ellas. Esto significa que un enrutador solo podrá seleccionar una ruta para llegar a cualquiera de las subredes, por lo que es mejor que todas estén en la misma dirección general.

Se agrega a la identificación de un host una **máscara de subred** que permitirá identificar la red a la que pertenece un host. De esta manera, podrá haber varios host con el mismo número de red pero pertenecen a distintas redes físicas.

Cuando el host quiere enviar un paquete a una cierta dirección IP, lo primero que hace es realizar un AND bit a bit entre su propia máscara de subred y la dirección IP de destino. Si el resultado es igual al número de subred del host de envío, entonces sabe que el host de destino está en la misma subred y el paquete se puede entregar directamente sobre la subred.

Si los resultados no son iguales, el paquete debe enviarse a un enrutador para que se reenvíe a otra subred.

La tabla de forwarding de un enrutador también cambia ligeramente cuando introducimos el subnetting. Recordemos que anteriormente teníamos una tabla de forwarding que consistía en entradas de la forma $< NetworkNum, NextHop >$. Para soportar el subnetting, la tabla ahora debe tener entradas de la forma $< SubnetNumber, SubnetMask, NextHop >$. Para encontrar la entrada correcta en la tabla, el enrutador realiza un AND bit a bit entre la dirección de destino

del paquete y la SubnetMask para cada entrada; si el resultado coincide con el SubnetNumber de la entrada, entonces esta es la entrada correcta para usar, y reenvía el paquete al enrutador indicado.

8.4.1. Direcciones sin clases

El subnetting tiene un contraparte, a veces llamado supernetting, pero más a menudo llamado **Classless Interdomain Routing** o **CIDR**, pronunciado "cider". CIDR lleva la idea de subnetting a su conclusión lógica al eliminar las clases de direcciones por completo.

El subnetting solo nos permite dividir una dirección de clase entre múltiples subredes, mientras que CIDR nos permite fusionar varias direcciones de clase en una sola "supernet" y lo hace de una manera que evita que el sistema de enrutamiento se sobrecargue.

CIDR trata de equilibrar el deseo de minimizar el número de rutas que un enrutador necesita conocer contra la necesidad de asignar direcciones de manera eficiente. Para hacer esto, nos ayuda a agregar rutas. Es decir, nos permite usar una sola entrada en una tabla de forwarding para decirnos cómo llegar a muchas redes diferentes. Como se señaló anteriormente, lo hace rompiendo los límites rígidos entre las clases de direcciones.

CIDR requiere un nuevo tipo de notación para representar los números de red, o prefijos como se los conoce, que pueden tener cualquier longitud. La convención es colocar un /X después del prefijo, donde X es la longitud del prefijo en bits.

Si asignamos prefijos a los clientes de tal manera que muchas redes de clientes diferentes conectadas a la red del proveedor compartan un prefijo de dirección común y más corto, entonces podemos obtener una mayor agregación de rutas.

IP Forwarding y CIDR

A veces puede sucederque varios prefijos en la tabla de forwarding "se superpongan", en el sentido de que algunas direcciones pueden coincidir con más de un prefijo.

La regla en este caso se basa en el principio de çoincidencia más larga"; es decir, el paquete coincide con el prefijo más largo.

El algoritmo para encontrar eficientemente la coincidencia más larga entre una dirección IP y los prefijos de longitud variable en una tabla de forwarding utiliza un enfoque conocido como árbol PATRICIA.

8.5. Addressing Translation (ARP)

Uno de los principales problemas que falta resolver es que los datagramas IP contienen direcciones IP, pero el hardware de interfaz física en el host o enrutador al que desea enviar el datagrama solo comprende el esquema de direccionamiento de esa red en particular. Por lo tanto, necesitamos traducir la dirección IP a una dirección de nivel de enlace que tenga sentido en esta red (por ejemplo, una dirección Ethernet de 48 bits). Luego podemos encapsular el datagrama IP dentro de un frame que contenga esa dirección de nivel de enlace y enviarlo al destino final o a un enrutador que prometa reenviar el datagrama hacia el destino final.

Esto se puede lograr utilizando el Protocolo de Resolución de Direcciones (ARP). El objetivo de ARP es permitir que cada host en una red construya una tabla de asignaciones entre direcciones IP y direcciones de nivel de enlace. Dado que estas asignaciones pueden cambiar con el tiempo, las entradas se eliminan periódicamente. Esto sucede en el orden de cada 15 minutos. El conjunto de asignaciones almacenadas actualmente en un host se conoce como caché ARP o tabla ARP.

ARP aprovecha el hecho de que muchas tecnologías de red de nivel de enlace, como Ethernet, admiten broadcasting. Si un host desea enviar un datagrama IP a otro host (o a un enrutador) que sabe que está en la misma red, primero verifica si hay un mapeo en la caché. Si no se encuentra ningún mapeo, debe invocar el Protocolo de resolución de direcciones en la red. Esto se hace transmitiendo una consulta ARP en la red que contiene la dirección IP de destino. Cada host recibe la consulta y verifica si coincide con su dirección IP. Si coincide, el host envía un mensaje de respuesta que contiene su dirección de nivel de enlace de regreso al originador de la consulta. El originador agrega la información contenida en esta respuesta a su tabla ARP.

Si el host ya tiene una entrada para ese host en su tabla, aumenta el time to live de esta entrada para que dure más tiempo.

El formato del paquete ARP para mapeos de direcciones IP a Ethernet contiene:

- Un campo **HardwareType**, que especifica el tipo de red física.
- Un campo **ProtocolType**, que especifica el protocolo de capa superior
- Campos **HLen** ("longitud de dirección de hardware") y **PLen** ("longitud de protocolo"), que especifican la longitud de la dirección de capa de enlace y la dirección de protocolo de capa superior, respectivamente
- Un campo **Operation**, que especifica si se trata de una solicitud o una respuesta
- Las direcciones de hardware (Ethernet) y protocolo (IP) de origen y destino

8.6. Host Configuration (DHPC)

Las direcciones IP, además de ser única únicas en una interred determinada, también deben reflejar la estructura de la misma. Como se señaló anteriormente, contienen una parte de red y una parte de host, y la parte de red debe ser la misma para todos los hosts en la misma red. Por lo tanto, no es posible configurar la dirección IP una vez en un host cuando se fabrica, ya que eso implicaría que el fabricante sabía qué hosts iban a terminar en qué redes, y significaría que un host, una vez conectado a una red, nunca podría moverse a otra. Por esta razón, las direcciones IP deben ser reconfigurables.

El método principal de configuración utiliza un protocolo conocido como **Protocolo de configuración dinámica de host (DHCP)**. Este protocolo se basa en la existencia de un servidor DHCP que es responsable de proporcionar información de configuración a los hosts. Existe al menos un servidor DHCP por dominio administrativo.

El servidor DHCP mantiene un grupo de direcciones disponibles que entrega a los hosts a pedido. Esto reduce considerablemente la cantidad de configuración que debe realizar un administrador, ya que ahora solo es necesario asignar un rango de direcciones IP (todas con el mismo número de red) a cada red.

Para contactar a un servidor DHCP, un host recién iniciado o conectado envía un mensaje **DHCPDISCOVER** a una dirección IP especial (255.255.255.255) que es una dirección de broadcast. Esto significa que será recibido por todos los hosts y enrutadores en esa red (Los enrutadores no reenvían dichos paquetes a otras redes, evitando la difusión a toda Internet). En el caso más simple, uno de estos nodos es el servidor DHCP para la red. El servidor respondería entonces al host que generó el mensaje de descubrimiento.

Sin embargo, no es realmente deseable requerir un servidor DHCP en cada red, porque esto implica una gran cantidad de servidores que deben configurarse correctamente y de manera consistente. Por lo tanto, DHCP utiliza el concepto de un agente de retransmisión. Hay al menos un agente de retransmisión en cada red, y se configura con solo una pieza de información: la dirección IP del servidor DHCP. Cuando un agente de retransmisión recibe un mensaje DHCP-DISCOVER, lo envía en unicast al servidor DHCP y espera la respuesta, que luego enviará al cliente solicitante.

El mensaje se envía realmente utilizando un protocolo llamado **Protocolo de Datagrama de Usuario (UDP)** que se ejecuta sobre IP.

En el caso en que DHCP asigne dinámicamente direcciones IP a los hosts, no pueden mantener las direcciones indefinidamente, ya que esto eventualmente causaría que el servidor agote su pool de direcciones. Al mismo tiempo, no se puede depender de que un host devuelva su dirección, ya que podría haberse bloqueado, desconectado de la red o apagado. Por lo tanto, DHCP permite que las direcciones se asignen por un período de tiempo determinado. Una vez que la asignación expire, el servidor es libre de devolver esa dirección a su pool. Un host con una dirección asignada necesita renovar el arriendo periódicamente si todavía está conectado a la red y funciona correctamente.

8.7. Error Reporting (ICMP)

IP siempre se configura con un protocolo complementario conocido como **Protocolo de mensajes de control de Internet (ICMP)**. Este protocolo define una colección de mensajes de error que se envían al host de origen cada vez que un enrutador o host no puede procesar un datagrama IP correctamente.

ICMP también define mensajes de control que un enrutador puede enviar de vuelta a un host de origen. Uno de los mensajes de control más útiles, llamado ICMP-Redirect, le dice al host de origen que hay una mejor ruta hacia el destino.

8.8. Virtual Networks and Tunneling

Las empresas con varios sitios tienden a construir redes privadas alquilando lineas de compañias electrónicas para interconectarlos. En estas redes, la comunicación se restringe a los sitios

de la corporación, lo cual es deseable por razones de seguridad. Para hacer una red privada virtual, las lineas de transmisión alquiladas - que no son compartidas con otras corporaciones - serían reemplazadas por algún tipo de red compartida. Un circuito virtual (VC) es un reemplazo razonable para una linea alquilada porque aún provee una conexión lógica punto a punto entre los sitios de la corporación.

También es posible proporcionar una función similar utilizando una red IP para proporcionar la conectividad. Sin embargo, no podemos simplemente conectar los sitios de las diversas corporaciones a una sola interred porque eso proporcionaría conectividad entre la corporación X y la corporación Y, lo que deseamos evitar. Para resolver este problema, necesitamos introducir un nuevo concepto, el **túnel IP**.

Podemos pensar en un túnel IP como un enlace virtual punto a punto entre un par de nodos que están separados por un número arbitrario de redes. El enlace virtual se crea dentro de un enrutador R_1 en la entrada del túnel al proporcionarle la dirección IP del enrutador R_2 en el otro extremo del túnel. Cada vez que R_1 desea enviar un paquete por este enlace virtual, encapsula el paquete dentro de un datagrama IP. La dirección de destino en el encabezado IP es la dirección de R_2, mientras que la dirección de origen es la de R_1.

Una vez que el paquete sale de R_1, parece al resto del mundo como un paquete IP normal destinado a R_2, y se reenvía en consecuencia. Todos los enrutadores en la interred lo reenvían utilizando medios normales, hasta que llega a R_2. Cuando R_2 recibe el paquete, encuentra que lleva su propia dirección, por lo que elimina el encabezado IP y mira la carga útil del paquete. Lo que encuentra es un paquete IP cuya dirección de destino está en una red interna. R_2 ahora procesa este paquete como cualquier otro paquete IP que recibe. Dado que R_2 está directamente conectado a esta red, reenvía el paquete a esa red.

Este mecanismo tiene sus desventajas. Una es que aumenta la longitud de los paquetes; esto podría representar una pérdida significativa de ancho de banda para paquetes cortos. Los paquetes más largos pueden estar sujetos a fragmentación, que tiene su propio conjunto de inconvenientes. También puede haber implicaciones de rendimiento para los enrutadores en cada extremo del túnel, ya que necesitan hacer más trabajo que el reenvío normal al agregar y eliminar el encabezado del túnel.

Finalmente, existe un costo de administración para la entidad administrativa que es responsable de configurar los túneles y asegurarse de que sean manejados correctamente por los protocolos de enrutamiento.

9. Ruteo interno y externo

El problema de ruteo es como los switches y routers adquieren la información en sus tablas de forwarding.

Si bien los términos **tabla de forwarding** y **tabla de ruteo** son a veces usados indistintamente, haremos una distinción entre ellos aquí. La **tabla de forwarding** es usada cuando un paquete está siendo reenviado y por lo tanto debe contener suficiente información para realizar la función de forwarding. Esto significa que una fila en la tabla de forwarding contiene el mapeo de un prefijo de red a una interfaz de salida y alguna información MAC, como la dirección Ethernet del siguiente salto. La **tabla de ruteo**, por otro lado, es la tabla construida por los algoritmos de ruteo como un precursor para construir la tabla de forwarding. Generalmente contiene mapeos de prefijos de red a siguientes saltos. También puede contener información sobre cómo se aprendió esta información para que el enrutador pueda decidir cuándo debe descartarla.

Los protocolos descritos en esta sección se conocen colectivamente como **protocolos de enrutamiento intradominio**, o Interior Gateway Protocols (IGP). Para comprender estos términos, debemos definir un dominio de enrutamiento: es una internet en la que todos los enrutadores están bajo el mismo control administrativo.

9.1. La red como un grafo

El ruteo es, en esencia, un problema de teoría de grafos. Los nodos del grafo pueden ser hosts, switches, routers o redes. Para nuestra discusión inicial, nos centraremos en el caso en que los nodos son enrutadores. Las aristas del grafo corresponden a los enlaces de red. Cada arista tiene un costo asociado, que da alguna indicación de la facilidad con la que se puede enviar tráfico a través de ese enlace.

Para resolver el problema de ruteo debemos encontrar el camino de menor costo entre dos nodos, donde el costo de un camino es igual a la suma de los costos de todas las aristas que componen el camino.

Los protocolos de ruteo proporcionan una forma distribuida y dinámica de resolver este problema en presencia de fallas de enlace y nodo y cambios en los costos de los enlaces.

Asumiendo que ya sabemos los costos de cada enlace vamos a ver dos algoritmos distribuidos que nos permiten crear la tabla de ruteo: **Distance Vector** y **Link State**.

9.2. Distance Vector

Cada nodo construye un vector que contiene las distancias (costos) a todos los otros nodos y distribuye ese vector a sus vecinos inmediatos. Vamos a suponer que cada nodo conoce el costo del enlace a cada uno de sus vecinos directamente conectados. Estos costos pueden ser provistos cuando el router es configurado por un administrador de red. Un enlace que está caído es asignado un costo infinito.

Inicialmente, cada nodo establece un costo de x a sus vecinos directamente conectados y ∞ a todos los demás nodos.

En el siguiente paso, cada nodo envía un mensaje a sus vecinos directamente conectados indicando su lista personal de distancias.

Cuando un nodo recibe un vector de distancias, compara las distancias recibidas con las que tiene almacenadas en su tabla de ruteo. Si la distancia recibida es menor que la que tiene almacenada, la actualiza y envía su propio vector de distancias a sus vecinos.

Table 3.11 Initial Routing Table at Node A

Destination	Cost	NextHop
B	1	B
C	1	C
D	∞	—
E	1	E
F	1	F
G	∞	—

Figura 21: Ejemplo tabla de ruteo

En ausencia de cambios en la topología, solo se necesitan unos pocos intercambios de información entre vecinos antes de que cada nodo tenga una tabla de enrutamiento completa. El proceso de obtener información de enrutamiento consistente para todos los nodos se llama convergencia.

Los algoritmos distribuidos como este permiten que todos los nodos en la red se adapten automáticamente a los cambios en la topología de la misma. No se requiere intervención manual por parte de un administrador de red.

9.2.1. Comportamiento ante fallas

Hay dos circunstancias en las cuales un nodo dado decide enviar una actualización de enrutamiento a sus vecinos:

- Una **actualización periódica** se envía cada cierto tiempo, incluso si no ha cambiado nada. Esto sirve para que los otros nodos sepan que este nodo todavía está en funcionamiento. También se asegura de que sigan recibiendo información que puedan necesitar si sus rutas actuales se vuelven inviables. La frecuencia de estas actualizaciones periódicas varía de un protocolo a otro, pero generalmente está en el orden de varios segundos a varios minutos.
- Cuando un enlace o nodo falla (**triggered update**). Los nodos que notan esta situación primero envían nuevas listas de distancias a sus vecinos, y normalmente el sistema se estabiliza bastante rápido a un nuevo estado.

Dependiendo el protocolo, la detección de una falla se puede dar de dos formas:

- Un nodo prueba continuamente el enlace a otro nodo enviando un paquete de control y viendo si recibe un acuse de recibo.
- Un nodo determina que el enlace (o el nodo en el otro extremo del enlace) está inactivo si no recibe la actualización de enrutamiento periódica esperada durante los últimos ciclos de actualización.

9.2.2. Conteo hasta el infinito

Sucede cuando se cae un nodo o enlace y los nodos restantes y hay nodos en la red que no alcanzan a enterarse de esto antes de enviar su vector de distancia. Los nodos que reciben esta información deciden usarlos para crear el nuevo camino que los lleve al nodo caído.

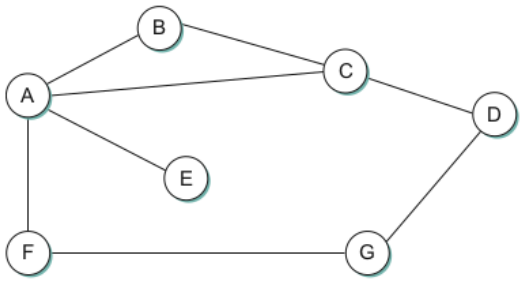

Figura 22: Grafo de una red

Supongamos que se cae el enlace de A a E en la red mostrada arriba, supongamos se produce la ronda de periodic updates. Todos los nodos, envían su vector de distancias a sus vecinos. Puede suceder lo siguiente:

1. A anuncia una distancia de ∞ a E.
2. B recibe de A: $\langle E, \infty \rangle$ y de C la tupla $\langle E, 2 \rangle$.
3. B decide que a partir de ahora puede llegar a E usando C con una distancia de 3 y reenvía su vector de distancias antes de recibir el vector de distancias modificado de C.
4. A recibe el mensaje de B y decide que puede llegar a E usando B con una distancia de 4 y reenvía su vector de distancias.
5. Cuando C reciba este paquete, sabe que puede usar A para recibir paquetes de E con una distancia de 5.

Este ciclo se detiene solo cuando las distancias alcanzan algún número lo suficientemente grande como para considerarse infinito. Mientras tanto, ninguno de los nodos sabe realmente que E es inalcanzable, y las tablas de enrutamiento de la red no se estabilizan.

Hay varias soluciones parciales a este problema:

- La primera es poner un número de distancia máximo pequeño como aproximación al infinito. Esto al menos limita la cantidad de tiempo que se tarda en contar al infinito. Por supuesto, también podría presentar un problema si nuestra red creciera hasta el punto en que algunos nodos estuvieran separados por más de la distancia máxima que habiamos definido.
- Otra técnica se llama **Split Horizon**: La idea es que cuando un nodo envía una actualización de enrutamiento a sus vecinos, no envía esas rutas que aprendió de cada vecino de vuelta a ese vecino. Por ejemplo, si B tiene la ruta $\langle E, 2, A\rangle$ en su tabla, entonces sabe que debe haber aprendido esta ruta de A, por lo que cada vez que B envía una actualización de enrutamiento a A, no incluye la ruta $\langle E, 2\rangle$ en esa actualización.
- Una variación más fuerte, llamada **Split Horizon with posion reverse**, vuelve a evniar la ruta A pero con una distancia de ∞. Esto le dice a A que no puede llegar a E a través de B.

El problema con estas técnicas que solo funcionan para loops que involucran dos nodos, se necesitan tomar decisiones más drásticas para loops más grandes. Se podría haber hecho que B y C esperen una cantidad determinada de tiempo antes de reenviar las nuevas rutas. Desafortunadamente, este enfoque retrasa la convergencia del protocolo; la velocidad de convergencia es una de las principales ventajas de su competidor: **Link-State Routing**.

Ruting Information Protocol (RIP)

Uno de los protocolos de enrutamiento más utilizados en las redes IP es el Protocolo de Información de Enrutamiento (RIP).

En una internet, el objetivo de los enrutadores es aprender a reenviar paquetes a varias redes. Por lo tanto, en lugar de anunciar el costo de llegar a otros enrutadores, los enrutadores anuncian el costo de llegar a las redes.

RIP es una implementación bastante sencilla del algoritmo de Vector Distance Routing. Los enrutadores que ejecutan RIP envían sus anuncios cada 30 segundos.

Un enrutador también envía un mensaje de actualización cada vez que una actualización de otro enrutador hace que cambie su tabla de enrutamiento. Un punto de interés es que admite múltiples familias de direcciones, no solo IP. RIP versión 2 (RIPv2) también introdujo las máscaras de subred mientras que RIP versión 1 funcionaba con las direcciones de clase antigua de IP.

9.3. Link State Routing

Link State Routing es la segunda clase principal de protocolo de enrutamiento intradominio. Se supone que cada nodo es capaz de descubrir el estado del enlace a sus vecinos y el costo de cada enlace.

Cada nodo sabe cómo llegar a sus vecinos directamente conectados, si nos aseguramos de que la totalidad de este conocimiento se difunda a cada nodo entonces cada nodo tendrá suficiente

conocimiento de la red para construir un mapa completo. Esta es una condición suficiente (aunque no necesaria) para encontrar el camino más corto a cualquier punto de la red. Los protocolos de link state routing se basan en dos mecanismos: la difusión confiable de la información de estado de enlace y el cálculo de rutas a partir de la suma de todo el conocimiento acumulado por este algoritmo.

9.3.1. Reliable flooding

La inundación confiable (reliable flooding) es el proceso de asegurarse de que todos los nodos que participan en el protocolo de enrutamiento obtengan una copia de la información de estado de enlace de todos los demás. Como sugiere el término inundación, la idea es que un nodo envíe su información de estado de enlace en todas sus conexiones directas; cada nodo que recibe esta información la reenvía en todas sus conexiones. Este proceso continúa hasta que la información ha llegado a todos los nodos de la red.

El paquete paquete de actualización que envían los nodos, también llamado paquete de estado de enlace (Link State Packet, LSP), contiene la siguiente información:

- El ID del nodo que creó el LSP.
- Una lista de vecinos directamente conectados de ese nodo, con el costo del enlace a cada uno.
- Un número de secuencia
- Un tiempo de vida (Time To Live - TTL) para este paquete

Los primeros dos elementos son necesarios para permitir el cálculo de rutas; los dos últimos se utilizan para que el proceso de inundación del paquete a todos los nodos sea confiable. La confiabilidad incluye asegurarse de tener la copia más reciente de la información, ya que puede haber múltiples LSP contradictorios de un mismo nodo circulando en la red.

Veamos como el proceso de flooding:

- Supongamos que un nodo A crea un LSP y lo envía a todos sus vecinos.
- Un nodo B que recibe una copia de un LSP que se originó en A.
- B compara el número de secuencia del LSP que acaba de recibir con el número de secuencia del LSP que ya tiene almacenado para A. Si el nuevo LSP tiene un número de secuencia más grande, se asume que es el más reciente, y ese LSP se almacena, reemplazando al antiguo. Un número de secuencia más pequeño (o igual) implicaría un LSP más antiguo (o no más nuevo) que el almacenado, por lo que se descartaría y no sería necesario tomar más medidas.
- Si el LSP recibido fue el más nuevo, B envía una copia de ese LSP a todos sus vecinos, excepto al vecino del que acaba de recibir el LSP.

El hecho de que el LSP no se reenvíe de vuelta al nodo desde el que se recibió ayuda a poner fin a la inundación del mismo. Dado que B pasa el LSP a todos sus vecinos, que luego hacen lo mismo, la copia más reciente del LSP eventualmente llega a todos los nodos.

Al igual que en RIP, cada nodo genera LSPs bajo dos circunstancias:

- La expiración de un LSP, debido a que su TTL ha llegado a cero.
- Un cambio en la topología de la red. Sin embargo, la única razón basada en la topología para que un nodo genere un LSP es si uno de sus enlaces conectados directamente o vecinos inmediatos ha fallado.

La falla de un enlace puede detectarse en algunos casos por el protocolo de capa de enlace. La desaparición de un vecino o la pérdida de conectividad con ese vecino se puede detectar mediante paquetes periódicos "hello". Cada nodo envía estos a sus vecinos inmediatos a intervalos definidos. Si pasa suficiente tiempo sin recibir un "hello"de un vecino, el enlace a ese vecino se declarará inactivo y se generará un nuevo LSP para reflejar este hecho.

9.3.2. Consideraciones de diseño

Uno de los objetivos de diseño más importantes de la inundación en este protocolo es que la información más reciente debe inundarse a todos los nodos lo más rápido posible, mientras que la información antigua debe eliminarse de la red. Además, es deseable minimizar la cantidad total de tráfico de enrutamiento que se envía por la red; después de todo, esto es solo una sobrecarga desde la perspectiva de aquellos que realmente usan la red para sus aplicaciones.

Una forma facil de reducir el overhead es generar LSPs solo cuando es necesario. Esto puede hacerse usando TTLs muy largos, a menudo del orden de horas. Dado que el protocolo de inundación es realmente confiable cuando cambia la topología, es seguro asumir que no es necesario emitir mensajes de que algo ha cambiado.

El algoritmo de enrutamiento de estado de enlace tiene muchas propiedades agradables: se ha demostrado que se estabiliza rápidamente, no genera mucho tráfico y responde rápidamente a los cambios de topología o fallas de los nodos. Por otro lado, la cantidad de información almacenada en cada nodo (un LSP para cada nodo en la red) puede ser bastante grande.

9.3.3. Número de secuencia

Cada vez que un nodo genera un nuevo LSP, lo incrementa en 1. A diferencia de la mayoría de los números de secuencia utilizados en otros protocolos, no se espera que estos se reseteen, por lo que el campo debe ser bastante grande (digamos, 64 bits).

Si un nodo se apaga y luego vuelve a encenderse, comienza con un número de secuencia de 0. Si el nodo estuvo inactivo durante mucho tiempo, todos los LSP antiguos para ese nodo habrán expirado; de lo contrario, eventualmente recibirá una copia de su propio LSP con un número de secuencia más alto, que luego puede incrementar y usar como su propio número de secuencia. Esto asegurará que su nuevo LSP reemplace cualquiera de sus viejos LSP que hayan quedado desde antes de que el nodo se apagara.

9.3.4. Time To Live

Se usa para asegurarse de que la información de estado de enlace antigua se elimine eventualmente de la red. Un nodo siempre decrementa el TTL de un LSP recién recibido antes de inundarlo a sus vecinos. También .[en]vejece.[el] LSP mientras se almacena en el nodo. Cuando el TTL llega a 0, el nodo vuelve a inundar el LSP con un TTL de 0, que es interpretado por todos los nodos de la red como una señal para eliminar ese LSP.

9.3.5. Calculo de rutas

Una vez que un nodo determinado tiene una copia del LSP de todos los nodos de la red, puede calcular un mapa completo para la topología de la red, y a partir de este mapa puede decidir la mejor ruta para cada destino.

En la práctica, cada switch calcula su tabla de enrutamiento directamente a partir de los LSP que ha recopilado utilizando una versión del algoritmo de Dijkstra llamado algoritmo de **forward searching**. Específicamente, cada switch mantiene dos listas, conocidas como `Tentative` y `Confirmed`. Cada una de estas listas contiene un conjunto de entradas de la forma $\langle$`Destino`, `Costo`, `NextHop`$\rangle$. El algoritmo funciona de la siguiente manera:

1. Se inicializa la lista `Confirmed` con una entrada para el propio nodo; esta entrada tiene un costo de 0.

2. Para el nodo recién agregado a la lista `Confirmed` en el paso anterior, se selecciona su LSP y lo marcamos como `Next`.

3. Para cada vecino (`Neighbor`) de `Next`, se calcula el costo (**Costo**) para llegar a este Vecino como la suma del costo desde mí mismo hasta `Next` y desde `Next` hasta `Neighbor`.

   a) Si `Neighbor` no está actualmente en la lista `Confirmed` ni en la lista `Tentative`, agregue $\langle$`Vecino`, `Costo`, `NextHop`$\rangle$ a la lista `Tentative`, donde `NextHop` es el nodo al que necesito ir para llegar a `Next`.

   b) Si `Neighbor` está actualmente en la lista `Tentative`, y el `Costo` es menor que el costo actualmente listado para `Neighbor`, reemplace la entrada actual con $\langle$`Neighbor`, `Costo`, `NextHop`$\rangle$, donde `NextHop` es el nodo al que necesito ir para llegar a `Next`.

4. Si `Tentative` está vacía, ya están todas las rutas calculadas. De lo contrario, se elije la entrada con el costo más bajo, se la mueve `Confirmed` y se vuelve a hacer todo el proceso desde el paso 2.

9.3.6. Open Shortest Path First Protocol (OSPF)

OSPF agrega una cantidad considerable de características al algoritmo básico de link-state descrito anteriormente:

- **Autenticación de mensajes de enrutamiento**: una característica de los algoritmos de enrutamiento distribuido es que dispersan información de un nodo a muchos, y toda la red puede verse afectada por la información incorrecta de un nodo. Por esta razón, es una buena idea asegurarse de que todos los nodos que participan en el protocolo puedan confiar en ellos. La autenticación de mensajes de enrutamiento ayuda a lograr esto.
- **Jerarquía**: Es una de las herramientas fundamentales utilizadas para hacer que los sistemas sean más escalables. OSPF introduce otra capa de jerarquía en el enrutamiento al permitir que un dominio se divida en áreas. Esto significa que un enrutador dentro de un dominio no necesariamente necesita saber cómo llegar a cada red dentro de ese dominio, puede ser capaz de salir adelante sabiendo cómo llegar a la zona correcta. Por lo tanto, hay una reducción en la cantidad de información que debe transmitirse y almacenarse en cada nodo.
- **Balanceo de carga**: OSPF permite que varias rutas al mismo lugar se asignen el mismo costo y hará que el tráfico se distribuya uniformemente entre ellas, lo que permite un mejor uso de la capacidad de red disponible.
- Todo el paquete, excepto los datos de autenticación, está protegido por una suma de comprobación de 16 bits utilizando el mismo algoritmo que la cabecera IP.
-

De los cinco tipos de mensajes OSPF, el tipo 1 es el mensaje "hola", que un enrutador envía a sus pares para notificarles que todavía está vivo y conectado como se describe anteriormente. Los tipos restantes se utilizan para solicitar, enviar y confirmar la recepción de mensajes de estado de enlace. El bloque de construcción básico de los mensajes de estado de enlace en OSPF es el anuncio de estado de enlace (Link Stata advertisement - LSA). Un mensaje puede llegar a contener varios LSA.

Específicamente, un enrutador que ejecuta OSPF puede generar LSPs que anuncian una o más de las redes que están directamente conectadas a ese enrutador. Además, un enrutador que está conectado a otro enrutador por algún enlace debe anunciar el costo de llegar a ese enrutador a través del enlace. Estos dos tipos de anuncios son necesarios para permitir que todos los enrutadores en un dominio determinen el costo de llegar a todas las redes en ese dominio y el siguiente salto apropiado para cada red.

9.4. Métricas

Una de las métricas más basicas, que es bastante razonable y muy simple, es asignar un costo de 1 a todos los enlaces: la ruta de menor costo será la que tenga menos saltos. Sin embargo, dicho enfoque tiene varias desventajas:

- No distingue entre enlaces en función de la latencia,
- No distingue entre rutas en función de la capacidad,

- No distingue entre enlaces en función de su carga actual, lo que hace imposible evitarlos si están sobrecargados.

9.4.1. ARPANET

La métrica de enrutamiento original de ARPANET medía la cantidad de paquetes que se encolaban esperando ser transmitidos en cada enlace. Sin embargo, dado que la longitud de la cola es una medida artificial de la carga, cuando un enlace tiene un costo demasiado alto, los paquetes se terminan enviando hacia la cola más corta en lugar de hacia el destino.

9.4.2. ARPANET v2

Una segunda versión del algoritmo de ruteo ARPANET, a veces llamado el nuevo mecanismo de ruteo, tuvo en cuenta tanto el ancho de banda del enlace como la latencia y utilizó el retraso, en lugar de la longitud de la cola, como medida de carga. Esto se hizo de la siguiente manera. Primero, cada paquete entrante se marca con la hora de llegada al enrutador ($ArrivalTime$) y se registra su hora de salida del enrutador ($DepartTime$). En segundo lugar, cuando se recibió el ACK de nivel de enlace desde el otro lado, el nodo calculó el retraso para ese paquete como

$$Delay = (DepartTime - ArrivalTime) + TransmissionTime + Latency$$

donde $TransmissionTime$ y $Latency$ se definieron estáticamente para el enlace y representan el ancho de banda y la latencia del mismo, respectivamente.

Si el ACK no llegó, sino que en su lugar el paquete agotó el tiempo de espera, entonces $DepartTime$ se restablece al momento en que se retransmitió el paquete. En este caso, $DepartTime - ArrivalTime$ captura la confiabilidad del enlace: cuanto más frecuente sea la retransmisión de paquetes, menos confiable será el enlace y más querremos evitarlo. Finalmente, el peso asignado a cada enlace se deriva del retraso promedio experimentado por los paquetes enviados recientemente a través de ese enlace.

Sin embargo, bajo una carga pesada, un enlace congestionado comienza a anunciar un costo muy alto. Esto causa que todo el tráfico se moviera fuera de ese enlace, dejándolo inactivo, por lo que luego anunciaría un costo bajo, atrayendo de vuelta todo el tráfico, y así sucesivamente. El efecto de esta inestabilidad es que, bajo una carga pesada, muchos enlaces pasarían mucho tiempo inactivos, que es lo último que desea bajo una carga pesada.

9.4.3. Revised ARPANET Routing Metric

Un tercer enfoque, llamado "métrica de enrutamiento ARPANET revisada", abordó estos problemas. Los principales cambios fueron comprimir el rango dinámico de la métrica considerablemente, tener en cuenta el tipo de enlace y suavizar la variación de la métrica con el tiempo.

La suavización se logró usando los siguientes mecanismos:

- La medición del retraso se transformó en una utilización del enlace, y este número se promedió con la última utilización informada para suprimir los cambios repentinos.

- Se impuso un límite estricto sobre cuánto podría cambiar la métrica de un ciclo de medición al siguiente. Al suavizar los cambios en el costo, la probabilidad de que todos los nodos abandonen una ruta a la vez se reduce en gran medida.

Parte V
Nivel de Transporte

10. Protocolos End-to-End

Los procesos del nivel de aplicación que usan los servicios de la capa de transporte esperan que proporcione ciertos requisitos:

- Garantizar la entrega de mensajes
- Entregar mensajes en el mismo orden en que se envían
- Entregar a lo sumo una copia de cada mensaje
- Soportar mensajes arbitrariamente grandes
- Soportar sincronización entre el emisor y el receptor
- Permitir al receptor aplicar control de flujo al emisor
- Soportar múltiples procesos de aplicación en cada host

10.1. UDP

El protocolo de transporte más simple posible es uno que extiende el servicio de entrega de host a host de la red subyacente en un servicio de comunicación de proceso a proceso. Es probable que haya muchos procesos en ejecución en cualquier host dado, por lo que el protocolo necesita agregar un nivel de demultiplexión, lo que permite que varios procesos de aplicación en cada host compartan la red. Aparte de este requisito, el protocolo de transporte no agrega ninguna otra funcionalidad al servicio de mejor esfuerzo proporcionado por la red subyacente. El **Protocolo de Datagramas de Usuario** (User Datagrama Procol - UDP) de Internet es un ejemplo de este tipo de protocolo de transporte.

Los procesos se identifican indirectamente entre sí utilizando un localizador abstracto, generalmente llamado puerto (**port**).

La cabecera de un protocolo de extremo a extremo que implementa esta función de demultiplexación típicamente contiene un identificador (puerto) tanto para el remitente (origen) como para el receptor (destino) del mensaje. Es decir, un proceso es realmente identificado por un puerto en algún host en particular, un par $\langle host, puerto \rangle$.

10.1.1. Comunicación entre procesos

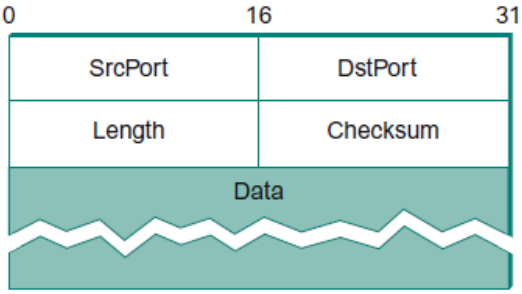

Figura 23: Formate de Header para UDP

Típicamente, un proceso cliente inicia un intercambio de mensajes con un proceso servidor. Una vez que el mensaje inicial esrecibido, el servidor conoce el puerto del cliente (del campo **SrcPrt** contenido en la cabecera del mensaje) y puede responderle.

Los servidores, esperan recibir mensajes en puertos bien conocidos (**well-known ports**). Algunos ejemplos:

- Los servidores de nombres de dominio (DNS) recibe mensajes en el puerto 53.
- El servicio de correo escucha mensajes en el puerto 25.
- Unix acepta mensajes en el puerto 517

Este mapeo se publica periódicamente en un RFC y está disponible en la mayoría de los sistemas Unix en el archivo /etc/services. A veces, un puerto bien conocido es solo el punto de partida para la comunicación: el cliente y el servidor usan el puerto bien conocido para acordar algún otro puerto que usarán para la comunicación posterior, dejando el puerto bien conocido libre para otros clientes.

10.1.2. Manejo de mensajes

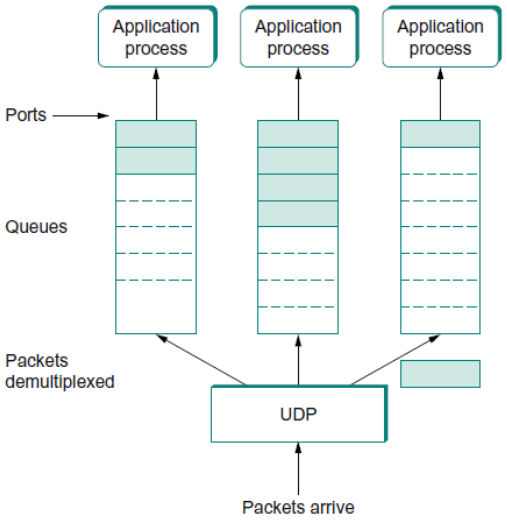

Figura 24: Cola de mensajes de UDP

Cuando llega un mensaje, el protocolo lo agrega al final de la cola. Si la cola está llena, el mensaje se descarta. No hay mecanismo de control de flujo en UDP para decirle al remitente que se ralentice. Cuando un proceso de aplicación desea recibir un mensaje, se elimina uno de la parte delantera de la cola. Si la cola está vacía, el proceso se bloquea hasta que haya un mensaje disponible.

Finalmente, aunque UDP no implementa el control de flujo o la entrega confiable / ordenada, también garantiza la corrección del mensaje mediante el uso de un cheksum. El mismo toma como entrada el encabezado UDP, el contenido del cuerpo del mensaje y algo llamado **pseudoencabezado**. El pseudoencabezado consiste en tres campos del encabezado IP: número de protocolo, dirección IP de origen y dirección IP de destino, más el campo de longitud UDP. La motivación detrás de tener el pseudoencabezado es verificar que este mensaje se haya entregado entre los dos puntos finales correctos.

10.2. Transmission Control Protocol (TCP)

TCP garantiza la entrega confiable y en orden de un flujo de bytes. Es un protocolo full-duplex, lo que significa que cada conexión TCP admite dos flujos de bytes simulteanos en dirección opuesta. También incluye un mecanismo de control de flujo para cada uno de ellos, lo que permite al receptor limitar la cantidad de datos que el remitente puede transmitir en un momento dado. La idea de este mecanismo es limitar la velocidad a la que TCP envía datos, no por el bien de evitar que el remitente sobrecargue al receptor, sino para evitar que el remitente sobrecargue la red. Finalmente, al igual que UDP, TCP admite un mecanismo de demultiplexación que permite que

varios programas de aplicación en cualquier host dado mantengan una conversación simultánea con sus pares.

10.2.1. Sliding Window en TCP

TCP usa el algoritmo de ventana deslizante para asegurar la entrega confiable de los mensajes. Aunque este es el mismo algoritmo básico que vimos en la Sección 4.5, debido a que TCP se ejecuta a través de Internet en lugar de una conexión punto a punto, hay muchas diferencias importantes:

- TCP admite conexiones lógicas entre procesos que se ejecutan entre dos computadoras cualquieras conectadas al Internet por lo que necesita una fase explícita de establecimiento de conexión durante la cual ambos nodos acurdan intercambiar información entre sí. Esto genera un estado compartido entre ellos que se usa para comenzar el algoritmo de ventana deslizante y que debe ser liberado cuando la conexión se cierra por lo que TCP también tiene una fase explícita de desmontaje de la conexión.
- Mientras que un solo enlace físico que siempre conecta las mismas dos computadoras tiene un tiempo de Round-Trip Time (RTT), las conexiones TCP probablemente tendrán tiempos de ida y vuelta diferentes dependiendo de la distancia entre ellas, la carga de la red, etc. Las variaciones en el RTT incluso son posibles durante una sola conexión TCP que dura solo unos minutos. Osea que el algoritmo de ventana deslizante debe tener un mecanismo retransmisiones con tiempos de espera adaptativos.
- Los paquetes pueden reordenarse a medida que cruzan el Internet. Los paquetes que están ligeramente desordenados no causan un problema ya que el algoritmo de ventana deslizante puede reordenar los paquetes correctamente usando el número de secuencia. El verdadero problema es cuán desordenados pueden estar los paquetes o, dicho de otra manera, cuán tarde puede llegar un paquete a su destino. En el peor de los casos, un paquete puede retrasarse en Internet hasta que expire el campo de tiempo de vida (TTL) de IP, momento en el cual el paquete se descarta (y, por lo tanto, no hay peligro de que llegue tarde). Sabiendo que IP descarta los paquetes después de que expire su TTL, TCP asume que cada paquete tiene un tiempo de vida máximo. El tiempo de vida exacto, conocido como tiempo de vida máximo del segmento (Maximum Segment Life - MSL), es una opción de ingeniería. La implicación es significativa: TCP tiene que estar preparado para que los paquetes muy antiguos aparezcan repentinamente en el receptor, lo que podría confundir el algoritmo de ventana deslizante.
- Casi cualquier tipo de computadora puede estar conectada a Internet, lo que hace que la cantidad de recursos dedicados a cualquier conexión TCP sea muy variable, especialmente considerando que cualquier host puede potencialmente admitir cientos de conexiones TCP al mismo tiempo. Esto significa que TCP debe incluir un mecanismo que cada lado usa para .ªprender"qué recursos (por ejemplo, cuánto espacio de búfer) el otro lado puede aplicar a la conexión. Este es el problema de control de flujo.

10.2.2. Formato de los segmentos

TCP es un protocolo orientado a bytes, lo que significa que el remitente escribe bytes en una conexión TCP y el receptor lee bytes de la conexión TCP. Aunque "flujo de bytes"describe el servicio que el protocolo ofrece a los procesos de aplicación, el mismo no transmite bytes individuales a través de Internet. En cambio, en el host de origen, almacena en un búfer suficientes bytes del proceso de envío para llenar un paquete de tamaño razonable y luego envía este paquete a su par en el host de destino. En el host de destino, TCP vacía el contenido del paquete en un búfer de recepción, y el proceso de recepción lee de este búfer a su gusto.

Los paquetes intercambiados entre pares TCP se llaman segmentos, ya que cada uno lleva un segmento del flujo de bytes. Cada segmento contiene el siguiente encabezado:

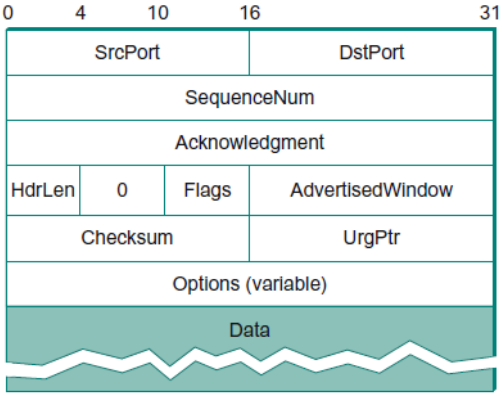

Figura 25: Encabezado de un paquete TCP

- `SrcPort` y `DstPort` identifican los puertos de origen y destino, respectivamente, al igual que en UDP. Estos dos campos, más las direcciones IP de origen y destino, se combinan para identificar de manera única cada conexión TCP. Es decir, la clave de demultiplexación de TCP está dada por la tupla de 4 elementos `SrcPort`, `SrcIPAddr`, `DstPort`, `DstIPAddr`.
- `Acknowledgment`, `SequenceNum` y `AdvertisedWindow` están involucrados en el algoritmo de ventana deslizante de TCP.
- `Flags` es un campo de 6 bits que se utiliza para transmitir información de control. Incluye los siguientes bits:
  - `ACK` se establece cada vez que el campo `Acknowledgment` es válido, lo que implica que el receptor debe prestar atención a él.
  - `URG` significa que este segmento contiene datos urgentes. Cuando se establece este indicador, el segemento contiene datos urgentes desde el byte 0 hasta el byte `UrgPtr`
  - `PUSH` significa que el remitente invocó la operación de empuje, esto indica que el receptor debe poder leer la información contenida en el segmento apenas llegue.

  - `RESET` significa que el receptor se ha confundido, por ejemplo, porque recibió un segmento que no esperaba recibir, por lo que desea abortar la conexión.

- `Checksum` se usa exactamente de la misma manera que para UDP: se calcula sobre el encabezado TCP, los datos TCP y el pseudoencabezado, que está compuesto por los campos de dirección de origen, dirección de destino y longitud del encabezado IP. El checksum es obligatorio para TCP tanto en IPv4 como en IPv6.
- `HdrLen` indica la longitud del encabezado TCP en palabras de 32 bits. Este campo también se conoce como el campo `Offset`, ya que mide el desplazamiento desde el inicio del paquete hasta el inicio de los datos.

10.2.3. Establecimiento de una conexión TCP

Una conexión TCP comienza con un cliente (caller) haciendo una apertura activa a un servidor (callee). Suponiendo que el servidor hubiera hecho una apertura pasiva anteriormente, los dos lados se involucran en un intercambio de mensajes para establecer la conexión. Solo después de que finalice esta fase, los dos lados pueden comenzar a enviar datos. De manera similar, tan pronto como un participante haya terminado de enviar datos, cierra una dirección de la conexión, lo que hace que TCP inicie una ronda de mensajes de terminación de conexión.

3-Way Handshake

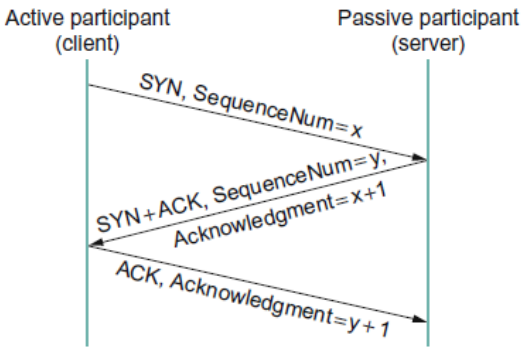

Figura 26: Intercambio de mensajes para 3-Way Handshake

El algoritmo utilizado por TCP para establecer y terminar una conexión se llama handshake de tres vías (Three Way Handshake).

Dos computadoras que quieran inicializar una conexión, deben acordar los números de secuencia de inicio que las dos partes planean usar para sus respectivos flujos de bytes. Primero, el cliente (el participante activo) envía un segmento al servidor (el participante pasivo) indicando el número de secuencia inicial que planea usar (`FLAG = SYN, SequenceNum = x`). El servidor responde con un solo segmento que reconoce el número de secuencia del cliente (`FLAG = ACK, Ack =`

`x + 1`) y establece su propio número de secuencia de inicio (`Flags = SYN + ACK, SequenceNum = y`). Finalmente, el cliente responde con un tercer segmento que reconoce el número de secuencia del servidor (`Flags = ACK, Ack = y + 1`). La razón por la cual cada lado reconoce un número de secuencia que es uno más grande que el enviado es que el campo de reconocimiento realmente identifica el "siguiente número de secuencia esperado", reconociendo implícitamente todos los números de secuencia anteriores. Aunque no se muestra en esta línea de tiempo, se programa un temporizador para cada uno de los dos primeros segmentos, y si no se recibe la respuesta esperada, el segmento se retransmite.

10.2.4. Diagrama de transición de estados de TCP

Todas las conexiones comienzan en el estado CLOSED. A medida que la conexión avanza, la conexión se mueve de un estado a otro de acuerdo con los arcos. Cada arco está etiquetado con una etiqueta de la forma evento / acción.

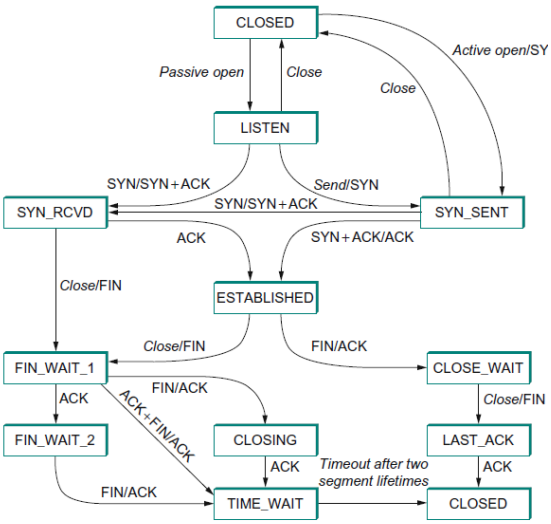

Figura 27: Transiciones de estado TCP

Inicio de conexión: Al abrir una conexión, el servidor primero invoca una operación de apertura pasiva, lo que hace que TCP se mueva al estado `LISTEN`. En algún momento posterior, el cliente realiza una apertura activa, lo que hace que su extremo de la conexión envíe un segmento `SYN` al servidor y se mueva al estado `SYN SENT`. Cuando el segmento `SYN` llega al servidor, se mueve al estado `SYN RCVD` y responde con un segmento `SYN + ACK`. La llegada de este segmento hace que el cliente se mueva al estado `ESTABLISHED` y envíe un `ACK` de vuelta al servidor. Cuando llega este `ACK`, el servidor finalmente se mueve al estado `ESTABLISHED`.

Si el `ACK` del cliente al servidor se pierde, la conexión aún funciona correctamente. Esto se debe a que el lado del cliente ya está en el estado `ESTABLISHED`, por lo que el proceso de aplicación local

puede comenzar a enviar datos al otro extremo. Cada uno de estos segmentos de datos tendrá el indicador `ACK` establecido y el valor correcto en el campo de reconocimiento, por lo que el servidor pasará al estado `ESTABLISHED` cuando llegue el primer segmento de datos. Este es en realidad un punto importante sobre TCP: cada segmento informa qué número de secuencia el remitente espera ver a continuación, incluso si esto repite el mismo número de secuencia contenido en uno o más segmentos anteriores.

Fin de la conexión: Las aplicaciones en ambos lados de la conexión deben cerrar de forma independiente su mitad de la conexión. Si solo un lado cierra la conexión, esto significa que no tiene más datos para enviar, pero aún está disponible para recibir datos del otro lado. Esto complica el diagrama de transición de estado porque debe tener en cuenta la posibilidad de que los dos lados invoquen el operador de cierre al mismo tiempo, así como la posibilidad de que primero un lado invoque el cierre y luego, en algún momento posterior, el otro lado invoque el cierre:

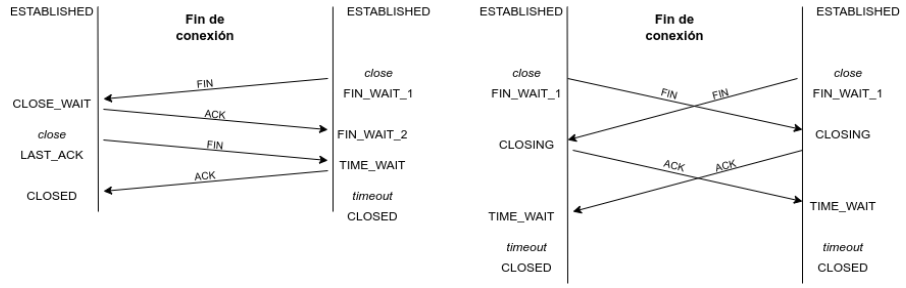

Figura 28: Diagrama de cierre de conexión

Una conexión en el estado `TIME_WAIT` no puede pasar al estado `CLOSED` hasta que haya esperado dos veces la cantidad máxima de tiempo que un datagrama IP podría vivir en Internet. La razón de esto es que, si bien el lado local de la conexión ha enviado un `ACK` en respuesta al segmento `FIN` del otro lado, no sabe si el `ACK` se entregó correctamente. Como consecuencia, el otro lado podría retransmitir su segmento `FIN`, y este segundo segmento podría retrasarse en la red.

Si se permitiera pasar directamente al estado CLOSED, entonces otro par de procesos de aplicación podrían llegar y abrir la misma conexión (es decir, usar el mismo par de números de puerto), y el segmento `FIN` retrasado de la encarnación anterior de la misma inmediatamente iniciaría la terminación de la encarnación posterior.

Sliding Window en TCP

La variante de TCP del algoritmo de ventana deslizante sirve para varios propósitos:

- Garantiza la entrega confiable de datos,
- Garantiza que los datos se entreguen en orden

- Impone un control de flujo entre el remitente y el receptor.

Para lograr lo último, en lugar de tener una ventana deslizante de tamaño fijo, el receptor anuncia un tamaño de ventana al remitente. Esto se hace utilizando el campo `AdvertisedWindow` en el encabezado TCP. Luego, el remitente está limitado a tener no más `AdvertisedWindow` bytes de datos no reconocidos en un momento dado. El receptor selecciona un valor adecuado para `AdvertisedWindow` en función de la cantidad de memoria asignada a la conexión con el fin de almacenar en búfer los datos. De esta manera, se evita que el remitente sobrecargue el búfer del receptor.

Envio confiable y ordenado

El remitente mantiene un búfer de envío que utiliza para almacenar datos que se han enviado pero que aún no se han reconocido, así como datos que ha escrito la aplicación de envío pero que aún no se han transmitido.

En el lado receptor, TCP mantiene un búfer de recepción. Este búfer contiene datos que llegan desordenados, así como datos que están en el orden correcto (es decir, no faltan bytes antes en el flujo) pero que el proceso de aplicación aún no ha tenido la oportunidad de leer.

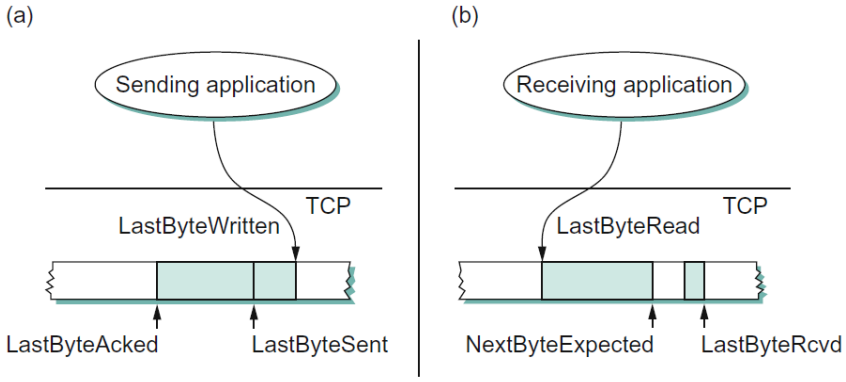

Figura 29: Buffers de TCP

Del lado del remitente, se mantienen tres punteros en el búfer de envío: `LastByteAcked`, `LastByteSent` y `LastByteWritten` y se cumplen las siguientes condiciones:

- `LastByteAcked` $\leq$ `LastByteSent`: El receptor no puede haber reconocido un byte que aún no se ha enviado
- `LastByteSent` $\leq$ `LastByteWritten`: TCP no puede enviar un byte que el proceso de aplicación aún no ha escrito.
- Ninguno de los bytes a la izquierda de `LastByteAcked` necesita ser guardado en el búfer porque ya han sido reconocidos, y ninguno de los bytes a la derecha de `LastByteWritten` necesita ser almacenado en el búfer porque aún no se han generado.

Un conjunto similar de punteros (números de secuencia) se mantienen en el lado receptor: `LastByteRead`, `NextByteExpected` y `LastByteRcvd`. Sin embargo, las desigualdades son un poco menos intuitivas, debido al problema de la entrega desordenada:

- `LastByteRead` < `NextByteExpected`: Un byte no puede ser leído por la aplicación hasta que sea todos los bytes anteriores y ese mismo byte hayan sido recibidos. `NextByteExpected` apunta al byte inmediatamente después del último byte que cumple con este criterio.
- `NextByteExpected` ≤ `LastByteRcvd`+1: Si los datos llegaron en orden, `NextByteExpected` apunta al byte después de `LastByteRcvd`, mientras que si los datos han llegado desordenados, entonces `NextByteExpected` apunta al comienzo del primer espacio en los datos.
- Los bytes a la izquierda de `LastByteRead` no necesitan estar en búfer porque ya han sido leídos por el proceso de aplicación local, y los bytes a la derecha de `LastByteRcvd` no necesitan estar en búfer porque aún no han llegado.

Control de flujo

Ambos búferes mencionados en la sección anterior son de algún tamaño finito, denotados como `MaxSendBuffer` y `MaxRcvBuffer`.

En TCP, el receptor se debe asegurar qué

$$\texttt{LastByteRcvd} - \texttt{LastByteRead} \leq \texttt{MaxRcvBuffer}$$

para evitar la sobrecarga de su búfer. Para esto, controla el tamaño de la ventana deslizante que anuncia al remitente seteando el campo `AdvertisedWindow` en el encabezado TCP con el valor

$$\texttt{MaxRcvBuffer} - ((\texttt{NextByteExpected} - 1) - \texttt{LastByteRead})$$

que es la cantidad de espacio libre que tiene en el búfer.

A medida que le llegan los datos, los reconoce siempre que también hayan llegado todos los bytes anteriores. Además, `LastByteRcvd` se mueve a la derecha (se incrementa), lo que significa que la ventana anunciada potencialmente se reduce. Si se reduce o no depende de qué tan rápido el proceso de la aplicación local está consumiendo datos. Si el proceso local está leyendo datos tan rápido como llegan, entonces la ventanta anunciada se mantiene. Sin embargo, si el proceso receptor se atrasa, tal vez porque realiza una operación muy costosa en cada byte de datos que lee, entonces la ventana anunciada se vuelve más pequeña con cada segmento que llega, hasta que eventualmente llega a 0.

Wraparound: El número de secuencia utilizado en una conexión determinada podría dar la vuelta, un byte con el número de secuencia x podría enviarse en un momento determinado, y luego en un momento posterior un segundo byte con el mismo número de secuencia x podría enviarse. Nuevamente, asumimos que los paquetes no pueden sobrevivir en Internet por más tiempo que el MSL recomendado. Si esto sucede o no depende de qué tan rápido se pueden transmitir los datos a través de Internet, es decir, qué tan rápido se puede consumir el espacio de número de secuencia de 32 bits.

Manteniendo el pipe lleno: El valor del `AdvertisedWindow` debe ser lo suficientemente grande como para permitir que el remitente mantenga el pipe lleno. El receptor es libre de no anunciar una ventana de tamaño máximo (tan grande como lo permite el campo `AdvertisedWindow`).

El máximo `AdvertisedWindow` posible se puede calcular como el producto de la demora y el ancho de banda $delay \times bandwidth$.

10.2.5. Transmisión de datos

TCP usa tres mecanismos para decidir cuando transmitir datos:

- TCP mantiene una variable, generalmente llamada **tamaño máximo de segmento** (MSS), y envía un segmento tan pronto como ha recopilado MSS bytes del proceso de envío. Generalmente se establece este valor de tal manera de evitar que la IP local se fragmente.Es decir, MSS se establece en la unidad máxima de transmisión (MTU) de la red conectada directamente, menos el tamaño de los encabezados TCP e IP.
- TCP admite una operación de empuje, y el proceso de envío invoca esta operación para vaciar efectivamente el búfer de bytes no enviados.
- El último disparador para transmitir un segmento es que se active un temporizador; el segmento resultante contiene tantos bytes como los que están actualmente almacenados en búfer para su transmisión.

Silly Window Syndrome: Supongamos que el remitente está acumulando bytes para enviar, pero la ventana está actualmente cerrada. Ahora suponga que llega un `ACK` que abre efectivamente una ventana de pocos bytes. La especificación original no aclaraba que hacer en esta situación y las primeras implementaciones de TCP decidieron transmitir un segmento medio lleno. Después de todo, no hay forma de saber cuánto tiempo pasará antes de que la ventana se abra aún más.

Resulta que esta estrategia de aprovechar agresivamente cualquier ventana disponible conduce a una situación ahora conocida como el `síndrome de la ventana tonta`.

Si el remitente llena agresivamente un contenedor vacío más pequeño que un MSS tan pronto como llega, el receptor ACK ese número de bytes, y por lo tanto el contenedor introducido en el sistema permanece en el sistema indefinidamente. Es decir, se llena e inmediatamente se vacía en cada extremo y nunca se fusiona con contenedores adyacentes para crear contenedores más grandes.

Actualmente, se puede retrasar este sindromehaciendo que el receptor espere a poder aceptar MSS bytes antes de anunciar una ventana abierta.

Dado que no podemos eliminar la posibilidad de que se introduzca un contenedor pequeño en el flujo, también necesitamos mecanismos para fusionarlos. El receptor puede hacer esto retrasando los ACK: enviando un ACK combinado en lugar de varios más pequeños, pero esta es solo una solución parcial porque el receptor no tiene forma de saber cuánto tiempo es seguro esperar a que llegue otro segmento o para que la aplicación lea más datos.

Algoritmo de Naggle: Mientras TCP tenga datos en vuelo, el remitente eventualmente recibirá un ACK. Este ACK se puede tratar como un temporizador que se activa, lo que desencadena la transmisión de más datos. El algoritmo de Nagle proporciona una regla simple y unificada para decidir cuándo transmitir:

La aplicacion produce b bytes:

Si $b \geq MSS$ y `AdvertisedWindow` $\geq MSS$

Se manda un segmento completo

Sino:

Si hay datos que todavia no fueron reconocidos (sin acks)

Acumular datos hasta que el próximo `ACL` llegue.

Sino:

Enviar todos los datos ahora

En otras palabras, siempre está bien enviar un segmento completo si la ventana lo permite. También está bien enviar inmediatamente una pequeña cantidad de datos si actualmente no hay segmentos en tránsito, pero si hay algo en vuelo, el remitente debe esperar un ACK antes de transmitir el siguiente segmento.

Debido a que algunas aplicaciones no pueden permitirse tal retraso para cada escritura que realiza en una conexión TCP, la interfaz de socket permite a la aplicación desactivar el algoritmo de Nagel estableciendo la opción TCP `NODELAY`. Usar esta opción significa que los datos se transmiten lo antes posible.

10.2.6. Retransmisión adaptativa

Debido a que TCP garantiza la entrega confiable de datos, retransmite cada segmento si no se recibe un `ACK` en un cierto período de tiempo. El protocolo establece este tiempo de espera como una función del RTT que espera entre los dos extremos de la conexión. Desafortunadamente, dada la gama de RTT posibles entre cualquier par de hosts en Internet, así como la variación en RTT entre los mismos dos hosts con el tiempo, elegir un valor de tiempo de espera apropiado no es tan fácil. Para abordar este problema, TCP utiliza un mecanismo de retransmisión adaptativo.

Algortimo original: Cada vez que TCP envía un segmento de datos, registra el tiempo. Cuando llega un ACK para ese segmento, TCP lee el tiempo nuevamente y luego toma la diferencia entre estos dos tiempos como un `SampleRTT`. TCP luego calcula un `EstimatedRTT` como un promedio ponderado entre la estimación anterior y esta nueva muestra. Es decir,

$$\texttt{EstimatedRTT} = \alpha \times \texttt{EstimatedRTT} + (1 - \alpha) \times \texttt{SampleRTT}$$

El parámetro α se selecciona para suavizar el `EstimatedRTT`. Un α pequeño realiza un seguimiento de los cambios en el RTT, pero tal vez esté demasiado influenciado por fluctuaciones temporales. Por otro lado, un α grande es más estable pero quizás no sea lo suficientemente

rápido como para adaptarse a cambios reales. La especificación original de TCP recomendó una configuración de α entre 0.8 y 0.9.

TCP luego usa `EstimatedRTT` para calcular el tiempo de espera de una manera bastante conservadora:

$$\texttt{TimeOut} = 2 \times \texttt{EstimatedRTT}$$

Algoritmo de Karn/Partridge: Un ACK no reconoce realmente una transmisión; en realidad, reconoce la recepción de datos. En otras palabras, cada vez que se retransmite un segmento y luego llega un ACK al remitente, es imposible determinar si este ACK debe asociarse con la primera o la segunda transmisión del segmento con el fin de medir el tiempo de ida y vuelta de la muestra. Es necesario saber qué transmisión asociar con el fin de calcular un `SampleRTT` preciso.

Este algoritmo, resuelve esto haciendo que solo se calcule un `SampleRTT` para segmentos que fueron transmitidos una sola vez. Cada vez que TCP retransmite un segmento, duplica el valor del siguiente timeout (usa **exponential backoff**). Esta última decisión se debe a que la congestión es la causa más probable de segmentos perdidos, lo que significa que el remitente de un mensaje no debe reaccionar agresivamente a un timeout. De hecho, cuanto más veces se agota el tiempo de espera de la conexión, más cautelosa debe ser la fuente.

Algoritmo de Jacobson/Karels: Es una mejora al algoritmo anterior. Si la variación entre las muestras es pequeña, entonces el `EstimatedRTT` puede ser considerado confiable y no hay razón para multiplicar esta estimación por 2 para calcular el timeout. Por otro lado, una gran varianza en las muestras sugiere que el valor de timeout no debe estar demasiado acoplado al `EstimatedRTT`. Para tener en cuenta estas situaciones, una vez obtenido un `SampleRTT`, se calcula el `TimeOut` de la siguiente manera:

$$\texttt{Difference} = \texttt{SampleRTT} - \texttt{EstimatedRTT}$$

$$\texttt{EstimatedRTT} = \texttt{EstimatedRTT} + (\delta \times \texttt{Difference})$$

$$\mathit{Deviation} = \mathit{Deviation} + (\delta \times (|\texttt{Difference}| - \texttt{Deviation}))$$

$$\texttt{TimeOut} = \mu \times \texttt{EstimatedRTT} + \phi \times \texttt{Deviation}$$

Donde δ es una valor valor entre 0 y 1. $\mu = 1$ y $\phi = 4$. De esta manera, si la varianza es pequeña, el `Timeout` es cercano al `EstimatedRTT`, y si la varianza es grande, entonces `Deviation` domina el calculo.

account. Intuitively, if the varia tion among samples is small, then the EstimatedRTT can be better trusted and there is no reason for multiplying this estimate by 2 to compute the timeout. On the other hand, a large variance in the samples suggests that the timeout value should not be too tightly coupled to the EstimatedRT Intuitivamente, si la variación entre las muestras es pequeña, entonces el `EstimatedRTT` puede ser mejor confiable y no hay razón para multiplicar esta estimación por 2 para calcular el tiempo de espera. Por otro lado, una gran varianza en las muestras sugiere que el valor de tiempo de espera no debe estar demasiado acoplado al `EstimatedRTT`.

11. Congestión

Resource allocation

Es el proceso por el cual los elementos de la red tratan de satisfacer las demandas competitivas que las aplicaciones tienen para los recursos de la red, principalmente el ancho de banda del enlace y el espacio de buffer en los routers o switches.

Control de congestión

Utilizamos el término control de congestión para describir los esfuerzos realizados por los nodos de la red para prevenir o responder a las condiciones de sobrecarga. Dado que la congestión es generalmente mala para todos, el primer orden de negocios es hacer que la congestión disminuya, o prevenirla en primer lugar. Esto podría lograrse simplemente persuadiendo a algunos hosts a dejar de enviar, mejorando así la situación para todos los demás. Sin embargo, es más común que los mecanismos de control de congestión tengan algún aspecto de equidad, es decir, tratan de compartir el dolor entre todos los usuarios, en lugar de causar un gran dolor a unos pocos.

En general, podemos encontrar dos tipos de mecanismos de control: **de lazo abierto** y **de lazo completo**:

- Los sistemas de control de lazo abierto se caracterizan por el hecho de que las mediciones tomadas se basan en mediciones previas y no en el estado actual del sistema. Por ejemplo, un sistema de control de lazo abierto podría medir la cantidad de tráfico que se envió en el pasado y luego usar esta información para determinar cuánto tráfico enviar en el futuro sin tener en cuenta el estado actual de los enlaces que hay que usar.
- Los sistemas de control de lazo cerrado aparecen ante la necesidad de corregir las desviaciones de la salida frente a la referencia de manera automática. Esto se logra mediante la realimentación de la salida del sistema hacia la entrada, de manera que la salida se compara con la referencia y se corrige el error. En este caso, el sistema de control de lazo cerrado mide la salida del sistema y la compara con la referencia para determinar si el sistema está funcionando correctamente. Si no es así, el sistema de control de lazo cerrado ajusta la entrada para corregir el error.

Flujo

Para esta sección vamos a asumir que la red es esencialmente sin conexión, con cualquier servicio orientado a la conexión implementado en el protocolo de transporte que se ejecuta en los hosts finales. Sin embargo, generalmente ocurre que una secuencia de datagramas entre un par de hosts particulares **fluye** a través de un conjunto particular de enrutadores, La suposición de que todos los datagramas son completamente independientes en una red sin conexión es demasiado fuerte. Los datagramas ciertamente se conmutan independientemente, pero generalmente ocurre que una secuencia de datagramas entre un par de hosts particulares fluye a través de un conjunto particular de enrutadores. Esta idea de un flujo, una secuencia de paquetes enviados entre un par

$\langle origen, destino \rangle$ y siguiendo la misma ruta a través de la red, es una abstracción importante en el contexto de la asignación de recursos.

Soft state

Dado que varios paquetes relacionados fluyen a través de cada enrutador, a veces tiene sentido mantener información de estado para cada flujo, información que se puede utilizar para tomar decisiones de asignación de recursos sobre los paquetes que pertenecen al mismo. Este estado a veces se llama **soft state**. Este estado representa un punto intermedio entre una red puramente sin conexión que no mantiene ningún estado en los enrutadores y una red puramente orientada a la conexión que mantiene un **hard state** en los enrutadores. En general, el funcionamiento correcto de la red no depende de que el **soft state** esté presente (cada paquete aún se enruta correctamente sin tener en cuenta este estado).

Un flujo puede definirse implícitamente o establecerse explícitamente. En el primer caso, cada enrutador observa los paquetes que viajan entre el mismo par de $\langle origen, destino \rangle$ inspeccionando las direcciones en el encabezado y trata estos paquetes como pertenecientes al mismo flujo. En el último caso, la fuente envía un mensaje de configuración de flujo a través de la red, declarando que un flujo de paquetes está a punto de comenzar.

Quality of Service (QoS)

Con **best effort serivce**, todos los paquetes reciben el mismo tratamiento, sin que los hosts finales tengan la oportunidad de pedir a la red que algunos paquetes o flujos reciban ciertas garantías o un servicio preferencial. Hay modelos de servicios que proporciona múltiples calidades de servicio (QoS).

11.1. Asiganción de recursos

11.1.1. Caracteristicas de mecanismos de asignación de recursos

Router Centric vs Host Centric

Los mecanismos de asignación de recursos se pueden clasificar en dos grupos amplios: aquellos que abordan el problema desde dentro de la red y aquellos que lo abordan desde los bordes:

En un diseño centrado en el enrutador, cada enrutador se hace responsable de decidir cuándo se reenvían los paquetes y seleccionar qué paquetes se deben descartar, así como de informar a los hosts que generan el tráfico de red cuántos paquetes se les permite enviar. En un diseño centrado en el host, los hosts finales observan las condiciones de la red y ajustan su comportamiento en consecuencia. Estos dos grupos no son mutuamente excluyentes, dado que es el caso de que tanto los enrutadores dentro de la red como los hosts en los bordes de la red participan en la asignación de recursos, el problema real es dónde recae la mayor parte de la carga.

Reserva vs Feedback

Los mecanimos de resource allocation según si usan reservas o feedback.

En un sistema basado en reservas, alguna entidad le pide a la red una cierta cantidad de capacidad para asignar a un flujo. Cada router asigna entonces suficientes recursos (buffers y / o porcentaje del ancho de banda al enlace) para satisfacer esta solicitud. Si hay un router que no puede satisfacer el pedio porque eso comprometería sus recursos, entonces rechaza la reserva. En un enfoque basado en feedback, los hosts finales comienzan a enviar datos sin reservar primero ninguna capacidad y luego ajustan su velocidad de envío de acuerdo con el feedback que reciben. Esta retroalimentación puede ser explícita (es decir, un enrutador congestionado envía un mensaje pidiendole que reduzca la velocidad) o implícita (es decir, ajusta su velocidad de envío de acuerdo con el comportamiento observable de la red, como pérdidas de paquetes).

Window vs Rate

Una tercera forma de caracterizar estos mecánismos es según si son basados en ventanas o en tasas (rate). TCP es uno de los protocolos basados en ventana en los que el receptor anuncia una ventana al remitente. Esta ventana corresponde a la cantidad de espacio de búfer que tiene el receptor y limita la cantidad de datos que el remitente puede transmitir; es decir, admite el control de flujo. Este mecanismo, tambien se puede usar dentro de la red para reservar espacio de búfer. También es posible controlar el comportamiento de un remitente utilizando un rate, es decir, cuántos bits por segundo el receptor o la red pueden absorber. El control basado en rates tiene sentido para muchas aplicaciones multimedia, que tienden a generar datos a una velocidad promedio y que necesitan al menos un rendimiento mínimo para ser útiles

11.1.2. Criterios de evaluación

Para calcular la efectividad de un esquema de asignación de recursos, se consideran las dos métricas principales de la red: throughput y delay. Queremos la mayor cantidad de throughput y la menor cantidad de delay posible. Desafortunadamente, lograr estos dos objetivos simultneamente no es feasible. Una forma segura de que un algoritmo de resource allocation aumente el throughput es permitir que tantos paquetes como sea posible entren en la red, para impulsar la utilización de todos los enlaces hasta el 100 %. El problema con esta estrategia es que aumentar el número de paquetes en la red también aumenta la longitud de las colas en cada enrutador. Las colas más largas, a su vez, significan que los paquetes se retrasan más tiempo en la red.

Para describir esta relación, algunos diseñadores de redes han propuesto han propuesto usar el ratio de throughput a delay como una métrica para evaluar la efectividad de un esquema de asignación de recursos. Este ratio es a veces referido como el poder de la red:

$$\text{Power} = \frac{\text{Throughput}}{\text{Delay}} \tag{1}$$

El objetivo es maximizar este ratio, que es una función de cuánta carga tiene la red. Buscamos un sistema estable, es decir que los paquetes continúen pasando por la red incluso cuando la red

está operando bajo una carga pesada. Si un mecanismo no es estable, la red puede experimentar un colapso de congestión.

Fair Resource Allocation

Cuando varios flujos comparten un enlace en particular, nos gustaría que cada flujo reciba una parte igual del ancho de banda. Esta definición presume que una parte justa del ancho de banda significa una parte igual del ancho de banda. Pero muchas veces, partes iguales pueden no equivaler a partes justas.

Suponiendo que justo implica igual y que todos los caminos tienen la misma longitud, se propuso una métrica que se puede usar para cuantificar la equidad de un mecanismo de control de congestión. El índice de equidad de Jain se define de la siguiente manera: Dado un conjunto de throughput de flujos $(x_1, x_2, ..., x_n)$ (medido en unidades consistentes como bits / segundo), la siguiente función asigna un índice de equidad a los flujos:

$$f(x_1, x_2, ..., x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2}$$

El índice de equidad siempre da como resultado un número entre 0 y 1, siendo 1 la mayor equidad.

11.2. Politicas de cola

Independientemente de lo simple o sofisticado que sea el mecanismo de asignación de recursos, cada enrutador debe implementar una política de encolamiento que indique cómo se almacenan en búfer los paquetes mientras esperan ser transmitidos.

11.2.1. First In First Out (FIFO)

El primer paquete que llega a un enrutador es el primer paquete en ser transmitido. Dado que la cantidad de espacio de búfer en cada enrutador es finita, si llega un paquete y la cola (espacio de búfer) está llena, el enrutador descarta ese paquete. Esto se hace sin tener en cuenta a qué flujo pertenece el paquete o qué tan importante es el paquete. A esto a veces se le llama **tail drop**, ya que los paquetes que llegan al final de la cola FIFO se descartan.

Una variación simple es usar encolamiento de prioridad. En esta politica se marca cada paquete con una prioridad; la marca podría llevarse, por ejemplo, en el encabezado IP. Los router implementan múltiples colas FIFO, una para cada clase de prioridad y siempre transmite paquetes desde la cola de mayor prioridad si esa cola no está vacía antes de pasar a la siguiente. Dentro de cada prioridad, los paquetes aún se administran de manera FIFO. Esta idea es una pequeña desviación del modelo de entrega de mejor esfuerzo, pero no va tan lejos como para garantizar una clase de prioridad en particular y puede generar starvation si no se implementa correctamente.

11.2.2. Fair Queueing

Se mantiene una cola separada por cada flujo que esté pasando por el router. El router envía paquetes desde esta cola usand round-robin. Cuando un flujo manda paquetes demasiado rápido, Cuando un flujo envía paquetes demasiado rápido, entonces su cola se llena. Cuando una cola alcanza su máxima longitud, los paquetes adicionales que pertenecen a ese flujo se descartan. De esta manera, una fuente determinada no puede aumentar arbitrariamente su parte de la capacidad de la red a expensas de otros flujos.

Sin embargo, los paquetes que se procesan en un enrutador no son necesariamente de la misma longitud. Para asignar verdaderamente el ancho de banda al enlace de salida de manera justa, es necesario tener en cuenta la longitud del paquete. Por esta razon, se usa un round robint bit a bit. Fair Queueing simula este comportamiento determinando primero cuándo terminaría de transmitirse un paquete determinado si se enviara utilizando este tipo de round robin y luego utilizando este tiempo de finalización para secuenciar los paquetes para su transmisión.

Si hay n flujos activos en un router, entonces el router calcula cuanto ticks de reloj va a tardar en enviar el siguiente paquete de cada flujo y le asigna un timestamp. Al momento de transmitir, el router elige el paquete con menor timestamp.

El enlace nunca queda inactivo si hay al menos un paquete en la cola. Cualquier esquema de encolamiento con esta característica se dice que conserva el trabajo. Si el enlace está completamente cargado y hay n flujos enviando datos, no puedo usar más de $1/n$ th del ancho de banda del enlace. Si intento enviar más que eso, mis paquetes se asignarán cada vez más grandes marcas de tiempo, lo que hará que se sienten en la cola más tiempo esperando la transmisión.

Es posible implementar una variación de FQ, llamada weighted fair queuing (WFQ), que permite asignar un peso a cada flujo (cola). Este peso especifica lógicamente cuántos bits transmitir cada vez que el enrutador atiende esa cola, lo que controla efectivamente el porcentaje del ancho de banda del enlace que ese flujo obtendrá.

11.3. Control de congestión en TCP

TCP asume que solo hay encolamiento FIFO en los routers de la red, pero también funciona con encolamiento justo.

Los hosts que desean enviar información pueden determinar cuanta capacidad de la red pueden usar usando TCP. Esto les sirve para saber cuantos paquetes pueden tener en tránsito. Una vez que transmitió esta cantidad de paquetes, usa la llegada de un ACK como una señal de que uno de sus paquetes salió de la red y que es seguro insertar uno nuevo sin agregar al nivel de congestión. Se dice que TCP es auto-regulado.

TCP mantiene una nueva variable de estado para cada conexión, llamada **CongestionWindow**, que es usada por la fuente para limitar cuanta información puede tener en tránsito en un momento dado. Se modifica el protocolo de tal manera que el número máximo de bytes de datos no reconocidos permitidos es ahora el mínimo de la ventana de congestión y la ventana anunciada.

$$\texttt{MaxWindow} = \text{mín}(\texttt{CongestionWindow}, \texttt{AdvertisedWindow})$$

El host emisor establece la ventana de congestión en función del nivel de congestión que percibe que existe en la red. Esto implica disminuir la ventana de congestión cuando el nivel de congestión aumenta y aumentarla cuando el nivel de congestión disminuye. En conjunto, para esto utiliza un politica de aumento aditivo / disminución multiplicativa (**Additive Increase / Multiplicative Decrease** - AIMD); Cada vez que recibe un timeout, disminuye la ventana de congestión a la mitad y cada vez que recibe tantos ACK como paquetes haya enviado, aumenta la ventana de congestión en 1 MSS (Maximum Segment Size).

Este patrón de aumentar y disminuir continuamente la ventana de congestión continúa durante toda la vida útil de la conexión. De hecho, si traza el valor actual de CongestionWindow como una función del tiempo, obtiene un patrón de dientes de sierra.

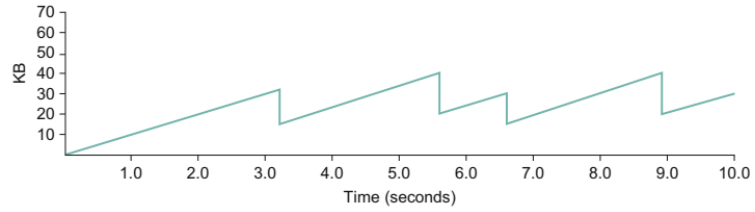

Figura 30: TCP Window Size

11.3.1. Slow Start

El mecanimos recién descripto se utiliza cuando el source host ya está haciendo uso de casi toda la capacidad disponible de la red. Cuando la conexión recién comienza, TCP setea la ventada de congestión en 1 MSS. Cada vez que recibe un ACK, aumenta la ventana de congestión en 1 MSS y transmite 2 paquetes. Cuando recibe todos los correspondientes a los paquetes en transmisión en ese momento, duplica la ventana de congestión. De esta forma, este valor aumenta exponencialmente hasta que se llega a perder el primer paquete. Aquí es donde se comienza a usa AIMD.

Slow start también se vuelve a usar en el caso de que la conexión haya quedado idle por un tiempo. Es decir, si la ventana de transmisión se redujo a cero.

Retransmisión y recuperación rápidas

Cada vez que un paquete de datos llega al lado receptor, el receptor responde con un ACK, incluso si este número de secuencia ya ha sido reconocido.

Cuando llega un paquete fuera de orden, TCP no puede reconocer los datos que contiene porque los datos anteriores aún no han llegado, entonces reenvía el mismo Ack que envió la última vez. Esta segunda transmisión se llama ACK duplicado. Cuando el lado emisor ve un ACK duplicado, sabe que el otro lado debe haber recibido un paquete fuera de orden, lo que

sugiere que un paquete anterior podría haberse perdido. Dado que también es posible que el paquete anterior solo se haya retrasado en lugar de perdido, el remitente espera hasta que ve algunos ACK duplicados y luego retransmite el paquete faltante. En la práctica, TCP espera hasta que ha visto tres ACK duplicados antes de retransmitir el paquete.

En general, esta técnica puede eliminar aproximadamente la mitad de los tiempos de espera en una conexión TCP típica, lo que resulta en una mejora de aproximadamente el 20 % en el rendimiento sobre lo que de otro modo se podría haber logrado.

Cuando el mecanismo de retransmisión rápida detecta la congestión, en lugar de reducir la ventana de congestión a un paquete y ejecutar el slow start, es posible utilizar los ACK que todavía están en transmición para sincronizar el envío de paquetes. Este mecanismo, que se llama recuperación rápida, elimina efectivamente la fase de inicio lento que ocurre entre cuando la retransmisión rápida detecta un paquete perdido y el aumento aditivo comienza.

11.3.2. Congestion Avoidance

TCP necesita crear pérdidas para encontrar el ancho de banda disponible de la conexión. Una alternativa atractiva, pero que aún no se ha adoptado ampliamente, es predecir cuándo está a punto de ocurrir la congestión y luego reducir la velocidad a la que los hosts envían datos justo antes de que comiencen a descartarse los paquetes.

DECbit

Cada enrutador monitorea la carga que está experimentando y notifica explícitamente a los nodos finales cuando está a punto de ocurrir una congestión. Esta notificación se implementa estableciendo un bit de congestión binario en los paquetes que fluyen a través del router. El host de destino copia este bit de congestión en el ACK que envía de vuelta a la fuente.

La fuente registra cuántos de sus paquetes resultaron en que algún enrutador estableciera el bit de congestión. En particular, la fuente mantiene una ventana de congestión, al igual que en TCP, y observa qué fracción de la última ventana de paquetes resultó en que se estableciera el bit. Si menos del 50 % de los paquetes de la ventana resultaron en que se estableciera el bit, la fuente aumenta su ventana de congestión en 1 MSS. Si más del 50 % de los paquetes de la ventana resultaron en que se estableciera el bit o hubo un timeout, la fuente reduce su ventana de congestión a la mitad.

Random Early Detection (RED)

Cada router está programado para monitorear su propia longitud de cola y, cuando detecta que la congestión es inminente, notificar (de manera implicita) a la fuente para que ajuste su ventana de congestión.

Esta notificación se implementa descartando paquetes antes de que la cola se llene. El router descarta paquetes de manera aleatoria, la probabilidad de descartar un paquete aumenta a medida que aumenta la longitud de la cola.

La fuente se da cuenta de esto cuando se produce el timeout para el paquete descartado o cuando recibe un ACK duplicado. En cuyo caso, reduce su ventana de congestión.

Para calcular la probabilidad con la que debe descartar un paquete, cada router calcula la longitud promedio ponderada de la cola:

$$\texttt{AvgLen} = (1 - \texttt{Weight}) \times \texttt{AvgLen} + \texttt{Weight} \times \texttt{SampleLen}$$

donde $0 \leq \texttt{Weight} \leq 1$ y `SampleLen` es la longitud de la cola en el momento en que se realiza la medición. En la mayoría de los casos ,este momento es cuando se recibe un paquete.

El promedio ponderado trata de detectar congestiones de larga duración. En el caso de que la cola del router se llene y vacie rapidamente sería erroneo notificar una congestión (esta situación es muy normal en el internet, dada la naturaleza de la transición que se realiza en ráfagas y no de manera uniforme a través del tiempo)

El router tiene configurado dos cotas para la longitud de la cola: `MinThreshold` y `MaxThreshold` Si $\texttt{AvgLen} < \texttt{MihThreshold}$ entonces no se descarta ningún paquete. Si $\texttt{AvgLen} > \texttt{MaxThreshold}$ entonces se descarta cada paquete. Si $\texttt{MinThreshold} < \texttt{AvgLen} < \texttt{MaxThreshold}$ entonces se descarta cada paquete con probabilidad P donde:

$$\texttt{TempP} = \texttt{MaxP} \times \frac{\texttt{AvgLen} - \texttt{MinThreshold}}{\texttt{MaxThreshold} - \texttt{MinThreshold}}$$

$$P = \frac{\texttt{TempP}}{1 - \texttt{count} \times \texttt{TempP}}$$

con `count` la cantidad de paquetes se encolaron y $\texttt{MaxP} = 0.02$.

La naturaleza aleatoria de RED confiere una propiedad interesante al algoritmo. La probabilidad de que se decida descartar los paquetes de un flujo en particular es aproximadamente proporcional a la parte del ancho de banda que ese flujo está recibiendo actualmente en ese router. Esto se debe a que un flujo que envía una cantidad relativamente grande de paquetes proporciona más candidatos para descartar al azar. Por lo tanto, hay cierto sentido de asignación justa de recursos incorporado en RED, aunque de ninguna manera es preciso.

Parte VI
Aplicaciones

En esta capa se encuentran las aplicaciones usadas por los usuarios de la red.

Los **procesos** (programas) que se ejecutan en un mismo host se pueden comunicar entre sí usando los sistemas de comunicación interproceso definidos por el sistema operativo sobre el que están corriendo.

Cuando un proceso necesita comunicarse con procesos ejecutándose en otros hosts, debe hacerlo usando algún protocolo de la capa de aplicación. Esto protocolos definen mensajes, reglas, acciones y servicios que deben aceptar las aplicaciones en cada punta de la conexión para poder comunicarse y hace uso de la capa de transporte para lograrlo.

12. Modelo Cliente-Servidor

En este modelo, un proceso cliente inicia la comunicación con un proceso servidor. El servidor espera a que lleguen conexiones de clientes y cuando llega una, la acepta y responde al cliente con los recursos solicitados.

Para identificar un proceso en un host se utiliza un **socket** que es un número formado por la dirección IP de un host y un número **port** que identifica el proceso dentro del host.

Cuando el paquete que desea enviar la aplicación llega a la capa de transporte, se utiliza TCP o UDP (dependiendo del protocolo de aplicación elegido) para enviar el mensaje al destino. La elección del protocolo de transporte depende de la aplicación y de las características de la red.

13. Domain Name System (DNS)

Es un servicio de resolución de nombres jerárquico distribuido. Se utiliza para traducir nombres de hosts a direcciones IP. Está implementado como un sistema de bases de datos distribuido generalizado para almacenar una variedad de información relacionada con la elección de un nombre.

Cuando un proceso necesita comunicarse con una llama a un procedimiento llamado **resolvedor** y le pasa como parámetro el nombre del host con el que desea comunicarse, por ejemplo: "www.google.com". El resolvedor envía un paquete UDP a un servidor DNS local que tiene configurado en el sistema. Este servidor devuelve la dirección IP correspondiente al servidor local y el último lo devuelve al host que originó el pedido.

Una vez que el host conoce la dirección IP del servidor, puede iniciar la comunicación con él usando UDP o TCP.

13.1. Espacio de dominios

La Internet está dividida en más de 250 **dominio de nivel superior** (Top-Level Domain), donde cada dominio cubre muchos hosts. Cada dominio está particionado en subdominios, y estos son particionados aún más, y así sucesivamente. Todos estos dominios pueden ser representados por un arbol, donde cada nodo interno es un dominio o subdominio y cada hoja es un host.

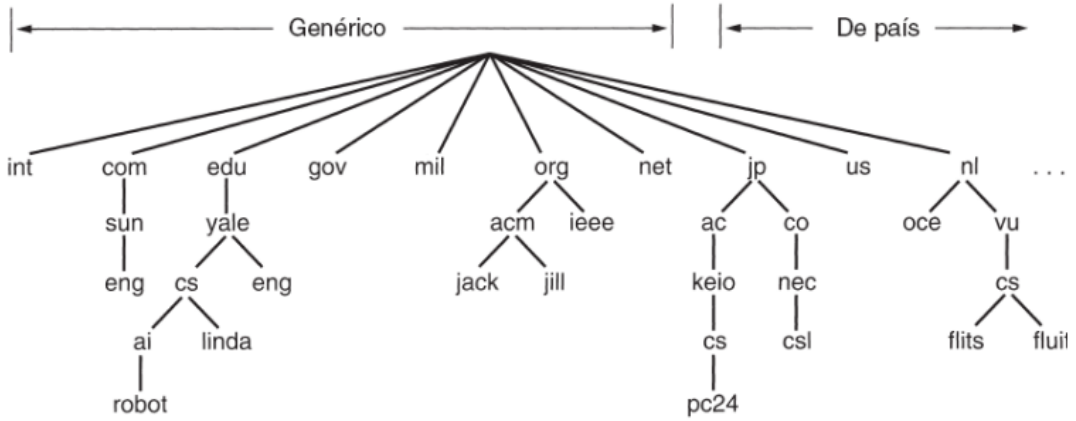

Figura 31: Arbol de dominios DNS

Los dominios de nivel superior vienen en dos sabores: genéricos y países. Los dominios genéricos incluyen dominios originales de los 80s y dominios introducidos a través de aplicaciones a ICANN. Otros dominios de nivel superior genéricos serán agregados en el futuro.

Obtener un dominio de segundo nivel, como nombre-de-empresa.com, es fácil. Los dominios de nivel superior son administrados por registradores designados por la **Internet Corporation for Assigned names and Numbers** (ICANN). Obtener un nombre simplemente requiere ir a un registrador correspondiente (para com en este caso) para verificar si el nombre deseado está disponible y no es una marca registrada de otra persona. Si no hay problemas, el solicitante paga al registrador una pequeña tarifa anual y obtiene el nombre.

13.2. Registros DNS

Cada dominio, ya sea un único host o un dominio de nivel superior, puede tener un conjunto de registros de recursos asociados con él. Estos registros son la base de datos DNS. Para un único host, el registro de recursos más común es solo su dirección IP, pero también existen muchos otros tipos de registros de recursos. Cuando un resolvedor le da un nombre de dominio a DNS, lo que obtiene son los registros de recursos asociados con ese nombre. Por lo tanto, la función principal de DNS es asignar nombres de dominio a registros de recursos.

Un registro de recursos es una tupla de cinco. Aunque están codificados en binario para la eficiencia, en la mayoría de las exposiciones los registros de recursos se presentan como texto ASCII, una línea por registro de recursos.

$$< \texttt{nombre\_dominio}, \texttt{tiempo\_de\_vida}, \texttt{clase}, \texttt{tipo}, \texttt{valor}, >$$

- El **nombre_dominio** es el nombre de dominio asociado al registro de recursos.
- El **tiempo_de_vida** indica que tan estable es el registro. Información que se actualiza periódicamente tiene un tiempo de vida corto, mientras que información que cambia raramente tiene un tiempo de vida largo.
- La **clase** es siempre IN, que significa Internet. Hay otras clases de recursos, pero no se usan en la práctica.
- El campo **tipo** indica el tipo de recursao y el campo **valor** contiene la información correspondiente y significa distintas cosas dependiendeo del tipo de recurso:
  - **SOA** (Start of Authority): Información sobre el servidor de la zona, el administrador, timeouts y como esta configurado
  - **A** (Address): Dirección IP de un host que acepta mensajes en el dominio.
  - **MX** (Mail Exchange): Nombre de algún host dentro del dominio que acepta emails.
  - **NS** (Name Server): Nombre de un servidor DNS dentro del dominio.
  - **CNAME** (Canonical Name): Nombre canónico de un alias, se utiliza cuando el dominio consultado en realidad es un alias de otro dominio
  - **PTR** (Pointer): Nombre de dominio asociado a una dirección IP específica.
  - **HINFO** (Host Information): Información sobre el host, como el tipo de CPU y el sistema operativo.
  - **TXT** (Text): Texto arbitrario asociado con el nombre de dominio.

13.3. Resolución de nombres

En teoría, un solo servidor de nombres podría contener toda la base de datos de DNS y responder a todas las consultas sobre ella. En la práctica, este servidor estaría tan sobrecargado como para ser útil. Además, si alguna vez se cae, toda la Internet se vería afectada.

Para evitar los problemas asociados con tener una sola fuente de información, el espacio de nombres de DNS se divide en zonas no superpuestas. Cada zona está asociada con uno o más servidores DNS que contienen la base de datos de la misma. Normalmente, una zona tendrá un DNS primario, que obtiene su información de un archivo en su disco, y uno o más DNS secundarios, que obtienen su información del DNS primario.

El proceso de buscar un nombre y encontrar una dirección se llama **resolución de nombres**. Cuando un resolvedor tiene una consulta sobre un nombre de dominio, pasa la consulta a un DNS. Si el dominio que se busca está bajo la jurisdicción del mismo, el DNS local responde con la dirección IP correspondiente. Si el dominio no está bajo la jurisdicción su jurisdicción, el DNS pasa la consulta a un un DNS de nivel superior.

El DNS de nivel superior responde con la dirección IP del DNS de la zona que contiene el dominio buscado. El servidor de DNS local luego pasa la consulta al DNS de la zona, que responde con la dirección IP del dominio buscado. Finalmente, el DNS local pasa la dirección IP al resolvedor, que puede entonces iniciar la comunicación con el host.

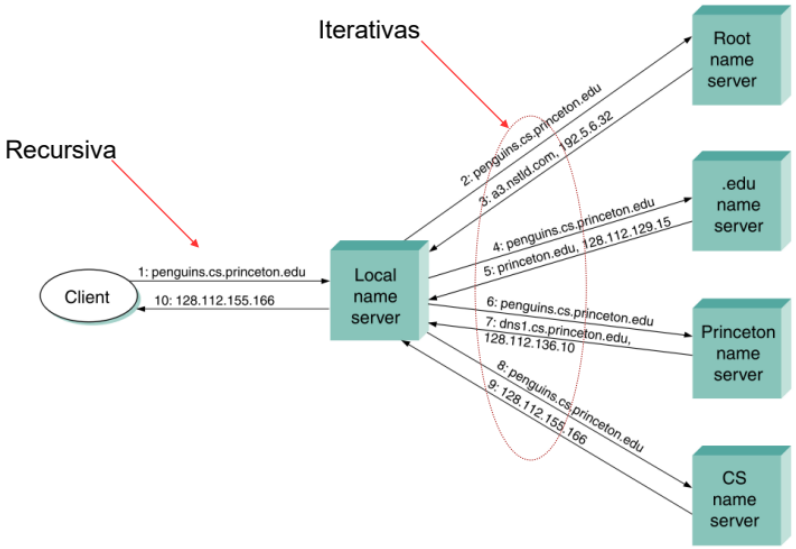

Figura 32: Consulta DNS

Si el dominio que se busca está bajo la jurisdicción del DNS, se dice que la respuesta obtenida es **autoritativa**: Un **registro autoritativo** es aquel que proviene de la autoridad que administra el registro y, por lo tanto, siempre es correcto. Las respuestas no autoritativas son respuestas que no provienen de la autoridad que administra el registro y, por lo tanto, pueden estar desactualizadas.

If the domain being sought falls under the jurisdiction of the name server, such as top.cs.vu.nl falling under cs.vu.nl, it returns the authoritative resource records. An authoritative record is one that comes from the authority that manages the record and is thus always correct. Authoritative records are in contrast to cached records, which may be out of date Si el dominio que se busca está bajo la jurisdicción del servidor de nombres, como top.cs.vu.nl que cae bajo cs.vu.nl, devuelve los registros de recursos autoritativos. Un registro autoritativo es aquel que proviene de la autoridad que administra el registro y, por lo tanto, siempre es correcto. Los registros autoritativos están en contraste con los registros en caché, que pueden estar desactualizados.

14. Envío de correo electrónico (SMTP)

14.1. Agente de usuarios:

El agente de usuario es un programa que proporciona una interfaz gráfica, o a veces una interfaz basada en texto y comandos que permite a los usuarios interactuar con el sistema de correo electrónico. Incluye un medio para componer, responder, mostrar y organizar mensajes. El acto de enviar nuevos mensajes al sistema de correo para su entrega se llama envío de correo.

Después de que se ha leído un mensaje, el usuario puede decidir qué hacer con él. Esto se llama disposición del mensaje. Las opciones incluyen eliminar el mensaje, enviar una respuesta, reenviar el mensaje a otro usuario y guardar el mensaje para referencia posterior. La mayoría de los agentes de usuario pueden administrar un buzón para correo entrante con varias carpetas para correo guardado. Las carpetas permiten al usuario guardar el mensaje según el remitente, el tema u otra categoría.

14.2. Servidor de correo

Los agentes de transferencia de mensajes son procesos del sistema. Se ejecutan en segundo plano en las máquinas del servidor de correo y están destinados a estar siempre disponibles. Su trabajo es mover automáticamente el correo electrónico a través del sistema desde el remitente hasta el destinatario con SMTP (Protocolo simple de transferencia de correo). Este es el paso de transferencia de mensajes.

14.3. Formato de email

El encabezado contiene toda la información necesaria para transportar el mensaje, como la dirección de destino, la prioridad y el nivel de seguridad, todos los cuales son distintos del mensaje en sí. Los agentes de transporte de mensajes utilizan esta información para el enrutamiento. Los campos más relevantes del encabezado son:

Campo	**Descripción**
To	Direcciónes de correo de los destinatarios primarios
CC	Direcciónes de correo de los destinatarios secundarios
BCC	Direcciónes de correo de los destinatarios ocultos
From	Persona o personas que crearon el mensaje
Sender	Dirección de correo del remitente
Received	Lista de servidores de correo que han recibido el mensaje
Return-Path	Usado por el servidor para identificar una ruta de regreso al remitente
Date	Fecha y hora en que se envió el mensaje
Reply-To	Dirección de correo a la que se debe responder
Message-Id	Identificador único para el mensaje
In-Reply-To	Identificador único para el mensaje al que se está respondiendo
References	Identificadores únicos para mensajes relacionados
Keywords	Palabras clave para ayudar a los usuarios a encontrar el mensaje
Subject	Tema del mensaje

14.3.1. MIME - Multipurpose Internet Mail Extensions

MIME define cinco nuevos encabezados de mensajes:

- **MIME-Version**: Incidica al agente de usuario que recibe el mensaje que está tratando con un mensaje MIME y qué versión de MIME usa. Cualquier mensaje que no contenga un encabezado MIME-Version: se asume que es un mensaje de texto en inglés (o al menos uno que usa solo caracteres ASCII) y se procesa como tal.
- **Content-Description**: Proporciona una descripción en inglés del contenido del mensaje.
- **Content-Id**: Proporciona un identificador único para el contenido del mensaje.
- **Content-Transfer-Encoding**: Indica cómo se codificó el contenido para que se pueda transmitir a través de la red.

  La codificación más usada se llama **base64 encoding**. En este esquema, grupos de 24 bits se dividen en cuatro unidades de 6 bits, y cada unidad se envía como un carácter ASCII legal. Para mensajes que son casi completamente ASCII pero con algunos caracteres no ASCII, se utiliza un esquema de codificación conocido como **quoted-printable encoding**.

  Finalmente, cuando hay razones válidas para no usar uno de estos esquemas, es posible especificar una codificación definida por el usuario.
- **Content-Type**: Indica el tipo de contenido que contiene el mmensaje:

Tipo	Ejemplos de Subtipo	Descripción
text	plain, html, xml, css	Texto plano en varios formato
image	gif, jpeg, tiff	Imagenes
audio	basic, mpeg, mp4	Audio
video	mpeg, mp4, quicktime	Video MPEG
model	vrml	Modelos 3D
application	octet-stream, pdf, js, zip	Datos producidos por aplicaciones
message	http, rvf822	Mesnaje encapsulado
multipart	mixed, alternative, parallel, digest	Combinación de varios tipos

14.3.2. Transferencia de mensajes

El correo electrónico se entrega haciendo que la computadora de envío establezca una conexión TCP al puerto 25 de la computadora de recepción. Escuchando este puerto hay un servidor de correo que habla SMTP (Protocolo simple de transferencia de correo). Este servidor acepta conexiones entrantes, sujeto a algunas verificaciones de seguridad, y acepta mensajes para su entrega. Si un mensaje no se puede entregar, se devuelve un informe de error que contiene la primera parte del mensaje no entregable al remitente.

Una vez que el servidor SMTP recibe un mensaje del agente de usuario, lo entregará al servidor SMTP que corresponde a la dirección del remitente del mail. Para identificar este servidor se

utiliza una consulta DNS, se crea una conexión TCP entre ambos servidores parar transferir el mensaje usando SMTP y luego se cierra la conexión.

Cuando el usuario remitente quiera leer su correo, el agente de usuario se conecta al servidor de correo de recepción usando POP3 (Protocolo de oficina de correos versión 3) o IMAP (Protocolo de acceso a mensajes de Internet).

14.3.3. IMAP - Internet Message Access Protocol

Uno fde los protocolos más usados en para realizar el delivery final al remitente es IMAP:

El cliente inicia un conexión seguro al servidor de mails y luego inicia sesión o se autentique de algun manera. Una vez iniciada la sesión, hay muchos comandos para enumerar carpetas y mensajes, recuperar mensajes o incluso partes de mensajes, marcar mensajes con banderas para su eliminación posterior y organizar mensajes en varias badnejas de entrada.

IAMP es un mejor de POP3, un protocolo más simple y menos seguro en el cual cada mail se descargaba en la computadora del usuario, en vez de mantenerse en los servidores de mail.

Resumen IMAP y POP3

Característica	**POP3**	**IMAP**
En dónde se define el protocolo	RFC 1939	RFC 2060
Puerto TCP utilizado	110	143
En donde se almacena el email	PC del usuario	Servidor
Como se lee el correo electrónico	Sin conexión	
Tiempo de conexión requerido	Breve	Largo
Uso de recursos de servidor	Bajo	Alto
Múltiples buzones de entrada	No	Si
Quien resplalda los buzones	Usuario	Servidor
Bueno para los usarios moviles	No	Si
Control del usuario sobre la descarga	Poco	Mucho
Descargas parciales de mensajes	No	Si
Sencillo de implementar	Si	No
Soporte amplio	Si	No

15. HTTP: Hypertext Transfer Protocol

Desde el punto de vista de los usuarios, la Web consiste en una vasta colección mundial de contenido en forma de páginas web. Cada página puede contener enlaces a otras en cualquier parte del mundo. Los usuarios pueden seguir un enlace haciendo clic en él, lo que los lleva a la página a la que apunta. La idea de tener una página que apunte a otra es llamada **hipertexto**. Un pedazo de texto, icono, imagen, etc. asociado con otra página se llama **hipervínculo**.

Cada página se obtiene enviando una solicitud a uno o más servidores, que responden con el contenido de la página. El protocolo de solicitud-respuesta para obtener páginas es un protocolo

de texto simple que se ejecuta sobre TCP, tal como era el caso de SMTP. Se llama **HTTP (Protocolo de transferencia de hipertexto)**. El contenido puede ser simplemente un documento que se lee de un disco, o el resultado de una consulta a una base de datos y la ejecución de un programa. La página es una **página estática** si es un documento que es el mismo cada vez que se muestra. En contraste, si se generó a pedido por un programa o contiene un programa, es una **página dinámica**.

Cada página se le asigna una **URL (Localizador de recursos uniforme)** que efectivamente sirve como el nombre mundial de la página. Las URL tienen tres partes: el protocolo (también conocido como el esquema), el nombre DNS de la máquina en la que se encuentra la página y la ruta que indica de forma única la página específica (un archivo para leer o programa para ejecutar en la máquina). En el caso general, la ruta tiene un nombre jerárquico que modela una estructura de directorio de archivos.

Cuando un usuario hace clic en un hipervínculo, el navegador realiza una serie de pasos para buscar la página a la que apunta:

1. El navegador determina la URL (viendo lo que se seleccionó).
2. El navegador hace una consulta DNS, la dirección IP del servidor donde está la página guardada.
3. El navegador establece una conexión TCP con el servidor en el puerto 80, el puerto conocido para el protocolo HTTP.
4. El navegador envía una solicitud HTTP pidiendo la página /index.html.
5. El servidor envía la página solicitada.
6. Si la página contiene hipervínculos, el navegador repite el proceso para cada uno de ellos usando la misma conexión TCP.
7. En HTTP1.0 la conexión se cierra y el navegador debe volver a realizar todos los pasos anteriores para pedir un nuevo recuro. En HTTP 2.0, la conexión TCP se mantiene abierta mientras el navegador sigue pidiendo recursos y se cierra despues de un tiempo de que se deja usar.

15.1. Cookies

Navegar por la Web como lo hemos descrito hasta ahora implica una serie de recuperaciones de páginas independientes. No hay concepto de una sesión de inicio de sesión. El navegador envía una solicitud a un servidor y recibe un archivo. Luego, el servidor olvida que alguna vez ha visto a ese cliente en particular.

Este problema se resuelve con un mecanismo a menudo criticado llamado cookies.

Cuando un cliente solicita una página web, el servidor puede proporcionar información adicional en forma de una cookie junto con la página solicitada. La cookie es una cadena de nombre pequeño (de como máximo 4 KB) que el servidor puede asociar con un navegador.

Esta asociación no es lo mismo que un usuario, pero es mucho más cercana y útil que una dirección IP. Los navegadores almacenan las cookies ofrecidas durante un intervalo, generalmente en un directorio de cookies en el disco del cliente para que las cookies persistan a través de las invocaciones del navegador, a menos que el usuario haya desactivado las cookies. Las cookies son solo cadenas, no programas ejecutables. En principio, una cookie podría contener un virus, pero dado que las cookies se tratan como datos, no hay una forma oficial para que el virus se ejecute y cause daños. Sin embargo, siempre es posible que algún hacker explote un error del navegador para causar la activación.

Una cookie puede contener hasta cinco campos:

- **Dominio:** El dominio indica a que sitio web pertenece la cookie.
- **Ruta:** La ruta indica a que directorio pertenece la cookie.
- **Content:** El contenido es un diccionario de de clave, valor separados por un signo igual.
- **Fecha de expiración:** La fecha de expiración indica cuando la cookie debe ser eliminada. Si este campo está ausente, entonces se dice que es una **cookie persistente**. Sino, es una **cookie no persistente**.
- **Secure:** Indica si la cookie debe ser enviada solo a través de conexiones seguras.

15.2. Mensajes HTTP

Aunque HTTP fue diseñado para su uso en la Web, se hizo intencionalmente más general de lo necesario con miras a futuros usos orientados a objetos. Por esta razón, se admiten operaciones, llamadas métodos, que no solo solicitan una página web:

- **GET:** El cliente solicita un objeto y el servidor responde con el objeto solicitado o un mensaje de error.
- **HEAD:** Se usa cuando, al cliente, solo le interesa la información contenida en el header de la respuesta. El servidor no envía el objeto, solo el encabezado.
- **POST:** El cliente envía un objeto al servidor y el último usa este objeto para realizar alguna acción, por ejemplo, modificar una base de datos, comprar algún item o ejecutar algún programa. El servidor responde con una página indicando el resultado de la operación realizada.
- **PUT:** Se usa para modifciar algún objeto ya guardado en servidor.
- **DELETE:** Se usa para eliminar algún objeto guardado en el servidor.
- **TRACE:** Se usa para que el servidor devuelva el mensaje recibido, esto se usa para debuggear.
- **OPTIONS:** Se usa para que el servidor devuelva los métodos que soporta.

- **CONNECT:** Se usa para establecer un túnel TCP/IP con el servidor.

Todas las respuestas están formadas por un header y un body. El header contiene información sobre la respuesta y el body contiene el objeto solicitado. El header contiene los siguientes campos:

- **Status line:** Contiene el protocolo, la versión y el código de estado de la respuesta. El código de estado es un número de tres dígitos que indica el resultado de la operación:

Código	Descripción	Ejemplo
1xx	Información	100 Continue
2xx	Éxito	200 OK
3xx	Redirección	301 Moved Permanently
4xx	Error del cliente	404 Not Found
5xx	Error del servidor	500 Internal Server Error

- **User-Agent:** Indica el nombre del navegador que realizó la solicitud.
- **Content-Type:** Indica el tipo de contenido del cuerpo del mensaje.
- **Accept:** Indica los tipos de contenido que acepta el cliente.
- **If Modified Since:** Indica la fecha de la última modificación del objeto solicitado.
- **Host:** Indica el nombre del host al que se le está haciendo la solicitud.
- **Authorization:** Es necesario para páginas que están protegidas. En este caso, el usuario debe probar que tiene las credenciales necesarias para realizar las operaciones que está solicitando.
- **Location:** Indica la URL a la que se debe redirigir el cliente.

16. CDN: Content Distribution Network

La idea es distribuir geográficamente una colección de servidores que almacenan copias de páginas web. Cuando un usuario quiere acceder a una página, se le redirige al servidor más cercano a su ubicación. Esto reduce la latencia y la congestión de la red.

Esto servidores se pueden ver como cachés. Si no tienen una página que haya sido solicitada por un cliente, le piden al servidor de back-end. En la práctica, sin embargo, los servidores de back-end replican proactivamente sus datos en los sustitutos en lugar de esperar a que los sustitutos lo soliciten bajo demanda. También es el caso de que solo las páginas estáticas, en oposición al contenido dinámico, se distribuyen en los sustitutos. Los clientes tienen que ir al servidor de back-end para cualquier contenido que cambie con frecuencia (por ejemplo, resultados deportivos y cotizaciones de acciones) o se produzca como resultado de algún cálculo (por ejemplo, una consulta de base de datos).

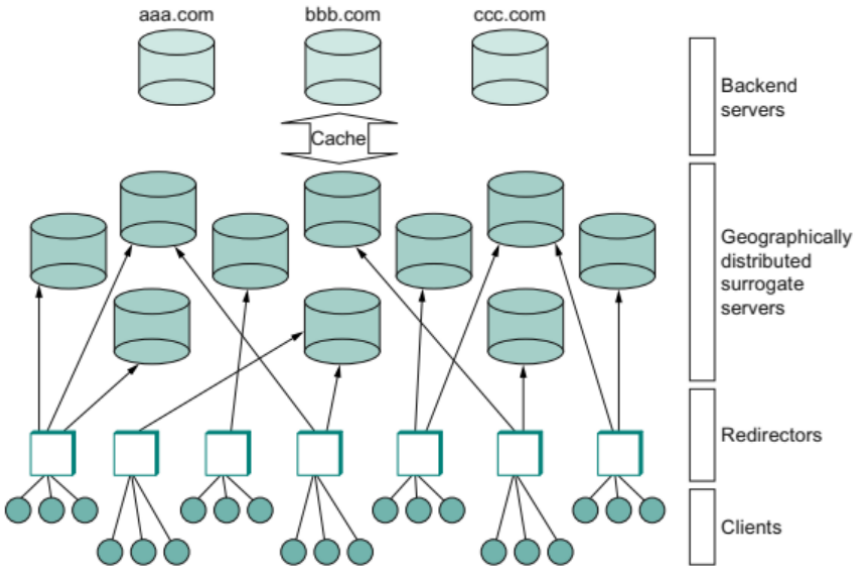

Figura 33: CDN

Las CDN también necesitan proporcionar un conjunto de redireccionadores que reenvíen las solicitudes de los clientes al servidor más apropiado. El objetivo principal de los redireccionadores es seleccionar el servidor para cada solicitud que resulte en el mejor tiempo de respuesta para el cliente. Un objetivo secundario es que el sistema en su conjunto procese tantas solicitudes por segundo como el hardware subyacente (enlaces de red y servidores web) pueda admitir. El número promedio de solicitudes que se pueden satisfacer en un período de tiempo determinado, conocido como el rendimiento del sistema, es principalmente un problema cuando el sistema está bajo una carga pesada.

16.1. CDNs Comerciales

Las CDN comerciales usan una combinación de reescritura de URL y redirección basada en DNS. Por razones de escalabilidad, el servidor DNS de alto nivel primero apunta a un servidor DNS de nivel regional, que responde con la dirección real del servidor. Para responder a los cambios rápidamente, los servidores DNS ajustan el TTL de los registros de recursos que devuelven a un período muy corto, como 20 segundos. Esto es necesario para que los clientes no almacenen en caché los resultados y, por lo tanto, no vuelvan al servidor DNS para obtener el último mapeo de URL a servidor.

17. Seguridad

Un protocolo provee **cofidencialidad** si encripta los mensajes para prevenir que un atacante pueda leerlos. Si además oculta la cantidad de información o el destino del mensaje, entonces ofrece **Confidencialidad de tráfico**.

Un atacante que no puede leer el contenido del mensaje encriptado sigue siendo capaz de modificar los paquetes nterceptados a través de una red y generar contenido válido. Si el protocolo implementa técnicas para detectar si un paquete fue modificado en el camino, entonces se dice que ofrece **Integridad**.

Otra amenaza para el cliente es ser dirigido sin saberlo a un sitio web falso. Esto puede resultar de un **DNS Attack**, en el que se ingresa información falsa en un servidor DNS o en la caché del DNS de la computadora del cliente. Esto lleva a traducir una URL correcta en una dirección IP incorrecta. Un protocolo que garantiza que realmente está hablando con quien cree que está hablando se dice que proporciona **autenticación**. La autenticación implica integridad, ya que no tiene sentido decir que un mensaje proviene de un cierto participante si ya no es el mismo mensaje.

Si el atacante en vez de crear un sitio falso, logra modificar los archivos de uno existente, Ese es un problema de **control de acceso**: hacer cumplir las reglas sobre quién está autorizado a hacer qué. Los sitios web también han sido objeto de ataques de denegación de servicio (DoS), durante los cuales los clientes potenciales no pueden acceder al sitio web porque está siendo abrumado por solicitudes falsas. Afecta el nivel de **disponibilidad**.

17.1. Principios de cifrado

El cifrado transforma un mensaje de tal manera que se vuelve ininteligible para cualquier parte que no tenga el secreto de cómo revertir la transformación. El remitente aplica una función de cifrado al mensaje de texto plano original y envía el resultado a través de la red. El receptor aplica una función de descifrado al mensaje que le llega para recuperar el mensaje original.

La transformación representada por una función de encriptado y su función de descifrado correspondiente se llama **cipher**.

Las funciones de cifrado y descifrado son de conocimiento público pero están parametrizadas por una clave secreta. Una razón para este principio es que si se depende de que el cifrado se mantenga en secreto, entonces hay que retirar el cifrado (no solo las claves) cuando sea vulnerado. Esto significa cambios potencialmente frecuentes de cifrado, lo cual es problemático, ya que toma mucho trabajo desarrollar uno nuevo. Además, una de las mejores formas de saber que un cifrado es seguro es usarlo durante mucho tiempo, si nadie lo rompe, probablemente lo sea.

Los mejores algoritmos de cifrado pueden evitar que el atacante deduzca la clave incluso cuando este conoce tanto el texto plano como el texto cifrado. Esto deja al atacante sin otra opción que probar todas las claves posibles usando fuerza bruta. La fuerza bruta puede ser computacionalmente ineficiente si se eligue un espacio de claves lo suficientemente grande y si las operaciones de cifrado y descifrado son lo suficientemente costosas. Una de las idifcultades de

esta elección es que las velocidad de computo aumentan con el tiempo, por lo que lo que ahora puede ser computacionalmente imposible, en el futuro quizas sea posible.

17.1.1. Cifrado en bloques

La mayoría de los cifrados son **cifrados de bloque**: se definen para tomar como entrada un bloque de texto plano de un cierto tamaño fijo, típicamente de 64 a 128 bits. El uso de un cifrado de bloque para encriiptar cada bloque de forma independiente, conocido como **cifrado de libro electrónico (Electronic Cipher Book - ECB)**, tiene la debilidad de que un valor de bloque de texto plano dado siempre dará como resultado el mismo bloque de texto cifrado. Por lo tanto, los valores de bloque recurrentes en el texto plano son reconocibles como tales en el texto encriptado, lo que hace que sea mucho más fácil para un criptoanalista romper el cifrado.

Para evitar esto, los cifrados de bloque siempre se amplían para hacer que el texto resultate para un bloque varíe según el contexto. Las formas en que se puede ampliar un cifrado de bloque se llaman **modos de operación**. Un modo de operación común es el **Cipher Block Chaining (CBC)**, en el que cada bloque de texto plano se XORea con el texto cifrado del bloque anterior antes de ser encriptado. El resultado es que el texto cifrado de cada bloque depende en parte de los bloques anteriores. Dado que el primer bloque no tiene precedente, se lo xorea con un número random llamado **vector de inicialización** que se acuerda al principio de la comunicación para que el receptor pueda descifrarlo.

17.1.2. Cifrados de clave secretas

En un cifrado de clave secretas o clave simetrica, ambos participantes de la comunicación comparten la misma clave. Osea que el mensaje es encriptado usando una clave particular y luego es desencriptado usando la misma clave. El problema es que la clave debe ser compartida entre los participantes de la comunicación, lo que puede ser difícil de lograr. Además, si la clave se ve comprometida, entonces todos los mensajes encriptados con esa clave también se ven comprometidos.

El Instituto Nacional de Estándares y Tecnología (NIST) de los EE. UU. ha emitido estándares para una serie de cifrados de clave secreta. El **Estándar de cifrado de datos (Data Encryption Standard - DES)** fue el primero, y ha resistido la prueba del tiempo en el sentido de que no se ha descubierto ningún ataque criptanalítico mejor que la búsqueda de fuerza bruta. Sin embargo, las claves de DES son relativamente cortas (56 bits), lo que significa que un atacante con suficiente poder de cómputo puede probar todas las claves posibles en un tiempo razonable. Por lo tanto, DES ya no se considera seguro.

Se actualizó este estandar con el cifrado **3DES** que aplica tres claves DES (168 bits) para encriptar el mensaje de la siguiente manera:

1. Usa la primer clave para encriptar el mensaje original.
2. La segunda se usa para desencriptar el resultado del paso anterior. Osea se aplica la función de descifrado de DES para obtener un nuevo mensaje.

3. La tercer clave se usa para encriptar el resultado del paso anterior. Osea se aplica la función de cifrado de DES para obtener el mensaje final.

El desencriptado implica hacer la inversa de cada uno de esos pasos: Desencriptar con la tercer clave, encriptar con la segunda y desencriptar con la primera.

Actualmente, 3DES está siendo reemplazado por el estándar **Advanced Encryption Standard (AES)** emitido por NIST que soporta claves de 128, 192 y 256 bits.

17.2. Cifrados de clave pública

Una alternativa a los cifrados de clave secreta son los cifrados de clave pública. En lugar de una sola clave compartida por dos participantes, un cifrado de clave pública utiliza un par de claves relacionadas, una para el cifrado y otra diferente para el descifrado. El par de claves es "propiedad"de un solo participante. El propietario mantiene la clave de descifrado en secreto para que solo él pueda descifrar los mensajes; esa clave se llama **clave privada**. Además, hace pública la clave de cifrado para que cualquiera pueda cifrar mensajes que le estén destinados; esa clave se llama **clave pública**.

Para que un esquema de este tipo funcione, no debe ser posible deducir la clave privada a partir de la pública.

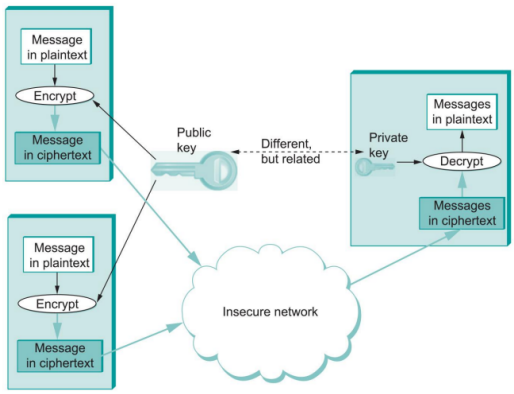

Figura 34: Encriptación con clave pública

Una clave para un cifrado de clave secreta proporciona un canal bidireccional entre dos participantes: cada participante tiene la misma clave (simétrica) que cualquiera puede usar para encriptar o desencriptar mensajes en cualquier dirección. Un par de claves pública / privada, en contraste, proporciona un canal unidireccional y de muchos a uno: desde todos los que tienen la clave pública hasta el único propietario de la clave privada.

Una propiedad adicional importante de los cifrados de clave pública es que la clave privada de "descifrado"se puede usar con el algoritmo de cifrado para cifrar mensajes de modo que solo se puedan descifrar usando la clave pública de çifrado". Esta propiedad, si bien no nos

ofrece confidencialidad, ya que cualquiera podría descifrar el mensaje, nos permite autenticar el mensaje. Si el mensaje se descifra correctamente, entonces el receptor pued estar seguro que el mensaje fue enviado por la persona que lo encriptó. Esto se llama **firma digital**.

El algoritmo de clave pública más conocido es el RSA (Rivest, Shamir, Adleman). Este algoritmo aprovecha el alto costo computacional de factorizar números grandes. El problema de encontrar una manera eficiente de factorizar números es uno en el que los matemáticos han trabajado sin éxito desde mucho antes de que apareciera RSA en 1978, y la resistencia posterior de RSA a la criptoanálisis ha fortalecido aún más la confianza en su seguridad. Desafortunadamente, RSA necesita claves relativamente grandes, al menos 1024 bits, para ser seguro. Las claves RSA son mas grandes que las claves para los cifrados de clave secreta porque es más rápido romper una clave privada RSA factorizando el número grande en el que se basa el par de claves que buscando exhaustivamente el espacio de claves.

En general, los cifrados de clave pública son usados principalmente para autenticar y distribuir claves secretas (simétricas) para su uso durante la comunicación entre dos entidades. Esto se debe a que los cifrados de clave pública son computacionalmente costosos en comparación con los cifrados de clave secreta.

17.2.1. Autenticadores

Un autenticador es un valor, que se incluye en un mensaje transmitido, que puede ser usado para verificar simultaneamente la autenticidad y la integridad de los datos de un mensaje.

Las sumas de verificación (checksums) y las comprobaciones de redundancia cíclica (CRC) son piezas de información agregadas a un mensaje para que el receptor detecte cuando el mensaje ha sido modificado inadvertidamente por errores de bits. Un concepto similar se aplica a los autenticadores, con el desafío adicional de que la corrupción del mensaje probablemente sea realizada deliberadamente por alguien que quiera que la corrupción pase desapercibida.

Para admitir la autenticación, un autenticador incluye alguna prueba de que quien creó el autenticador conoce un secreto que solo conoce el supuesto remitente del mensaje.

Cryptographic checksum: Combina encriptación y una función hash criptográfica. Los algoritmos hash criptográficos se tratan como conocimiento público, como con los algoritmos de cifrado. Una función hash criptográfica (también conocida como suma de verificación criptográfica) es una función que produce suficiente información redundante sobre un mensaje para exponer cualquier manipulación. El resultado de esta función se llama **Message Digest** y se agrega al final del mensaje antes de encriptar. El receptor puede verificar la integridad del mensaje descifrando el mensaje y luego aplicando la función hash criptográfica al mensaje descifrado. Si el resultado coincide con el Message Digest recibido, entonces el mensaje no se ha modificado.

Todos los message digests generados por la función son del mismo tamaño, idependientemente de la longitud del mensaje original. Dado que el espacio de mensajes posibles es más grande que el espacio de message digests, habrá mensajes distintos que tienen el mismo valor de message digest. Para que la función de hash sea segura en necesario que tenga la propiedad **one-way**:

Debe ser computacionalmente imposible encontrar un mensaje que produzca el mismo message digest que el mensaje original.

Una condición necesaria para que esto se cumpla es que el output de esta función debe estar distribuido uniformemente en el espacio de message digests. Si no lo está, entonces habrá regiones del espacio de message digests que son más probables que otras y, por lo tanto, más fáciles de encontrar.

Ha habido varios algoritmos hash criptográficos comunes a lo largo de los años, incluidos Message Digest 5 (MD5) y la familia Secure Hash Algorithm (SHA). Las debilidades de MD5 y las versiones anteriores de SHA han sido conocidas por algún tiempo, lo que llevó al NIST a recomendar el uso de SHA-3 en 2015.

17.2.2. Firma Digital

Un digest encriptado con un algoritmo de clave pública pero usando la clave privada se llama firma digital porque proporciona no repudio como una firma escrita. El receptor de un mensaje con una firma digital puede demostrar a cualquier tercero que el remitente realmente envió ese mensaje, porque el tercero puede usar la clave pública del remitente para verificarlo por sí mismo.

17.2.3. Message Authentication Code (MAC)

Se usa una clave secreta (conocida solo por el remitente y el receptor) para crear un código MAC que se concatena al mensaje de texto plano (antes de encriptar). Cuando el recptor recibe el mensaje, calcul su propio MAC usando el texto del mensaje que le llegó y la clave secreta. Si el código MAC calculado coincide con el código MAC recibido, entonces el receptor puede estar seguro de que el mensaje no se ha modificado y que fue enviado por alguien que conoce la clave secreta.

Una variación de este esquema es el **HMAC** en la que el mensaje se concatena la clave secreta y se pasa a una función de hash criptográfica. El resultado que devuelve está función es el codigo que se concatena al mensaje antes de ser enviado.

17.3. Distribución de claves

Para poder usar autenticadores o cifrado de clave secreta, los participantes deben saber qué claves usar.

Session Key: Una clave secreta que se genera para su uso en una sola sesión de comunicación, es válida durante un pequeño intervalo de tiempo. Cada sesión tiene un clave diferente y esta es determinada por medio de algún protocolo de protocolo especial.

Predistributed Keys: Son claves que se distribuyen antes de que se necesiten. En general, son manejadas por un centro de administración de claves que permite a los usuarios establecer sesiones de comunicación segura entre ellos. En general, este tipo de claves solo se utiliza al

comienzo de una comunicación para establecer una clave de sesión entre los participantes para luego pasar a usar esa clave de sesión para el resto de la comunicación.

Hay dos razones para que estas claves se usen de esta forma:

- Limitar la cantidad de veces que se usa una clave resulta en menos tiempos para ataques que sean intensivos computacionalmente, menos texto cifrado que un atacante pueda interceptar y menos información expuesta si la clave se ve comprometida.
- Las claves públicas son superiores a las claves secretas en términos de autenticación y establecimiento de clave, pero es demasiado lento cifrar mensajes completos con ellas.

17.3.1. Predistribución de claves públicas

Los algoritmos para crear un par de claves publica/privadas son de público conocimiento y es facil de econtrar software que lo haga.

Supongamos que Alice quiere usar cifrado asimétrico, entonces podría generar su propio par de claves y publicar la clave publica. El problema es como publicar la clave de tal maner que otros usuarios puedan estar seguros de que la clave publica realmente le pertenece a ella.

Public Key Infrastructure (PKI)

Una PKI comienza con la capacidad de verificar identidades y vincularlas a claves fuera de banda. Por "fuera de banda", nos referimos a algo fuera de la red y de las computadoras que la componen, por ejemplo, si Alice y Bob son individuos que se conocen, entonces podrían reunirse en la misma habitación y Alice podría darle su clave pública a Bob directamente, tal vez en una tarjeta de presentación.

Establecer claves fuera de banda no parece que escalaría bien, pero es suficiente para arrancar una PKI. El conocimiento de Bob de que la clave de Alice es X puede ser ampliamente, difundido de forma escalable usando una combinación de firmas digitales y un concepto de confianza.

Una declaración firmada digitalmente de una vinculación de clave pública se llama **certificado de clave pública**, o simplemente certificado. Este certificado debe contener:

- La identidad de la entidad que está siendo certificada.
- La clave pública de esa entidad.
- La identidad del que firma el certificado
- La firma digital
- Un identificador del algoritmo de firma digital que se usó para crear la firma.

Autoridades de certificación

Si X certifica que una cierta clave pública pertenece a Y, y luego Y certifica que otra clave pública pertenece a Z, entonces existe una cadena de certificados de X a Z, incluso si X y Z nunca se han conocido.

Una **autoridad de certificación** o **autoridad de certificados (CA)** es una entidad que se afirma que es confiable para verificar identidades y emitir certificados de clave pública.

Hay CA comerciales, CA gubernamentales e incluso CA gratuitas. Para usar una CA, se debe conocer su clave, la cual se puede aprender la clave de una CA si puede obtener una cadena de certificados que comienza con una CA cuya clave ya conoce. Luego puede creer cualquier certificado firmado por esa nueva CA.

Una forma común de construir tales cadenas es organizarlas en una jerarquía estructurada en árbol. En la parte superior de la jerarquía se encuentra una CA raíz, que es una CA que se considera confiable sin necesidad de un certificado. La CA raíz emite certificados para otras CA, que a su vez emiten certificados para otras CA, y así sucesivamente.

Hay algunos problemas importantes con la construcción de cadenas de confianza. Lo más importante es que, incluso si se está seguro de que se tiene la clave pública de la CA raíz, debemos asegurar que cada CA, desde la raíz hacia abajo, esté haciendo su trabajo correctamente. Si solo una CA en la cadena está dispuesta a emitir certificados a entidades sin verificar sus identidades, entonces lo que parece ser una cadena válida de certificados se vuelve insignificante.

Firefox e Internet Explorer vienen pre-equipados con certificados para un conjunto de CA; en efecto, el productor del navegador ha decidido que estas CA y sus claves pueden ser confiables. Un usuario también puede agregar CA a las que su navegador reconoce como confiables. Estos certificados son aceptados por Secure Socket Layer (SSL) / Transport Layer Security (TLS), el protocolo más utilizado para asegurar las transacciones web.

Web of trust

Un modelo alternativo de confianza es la red **Pretty Good Privacy (PGP)**: es un sistema de seguridad para correo electrónico, en el cual las direcciones de email son las identidades a las que se vinculan las claves y por las que se firman los certificados. En consonancia con las raíces de PGP como protección contra la intrusión del gobierno, no hay CA. En cambio, cada individuo decide en quién confía y cuánto confía en ellos; en este modelo, la confianza es una cuestión de grado. Un certificado puede incluir un nivel de confianza que indique cuán confiado está el firmante de la vinculación de claves reclamada en el certificado, por lo que un usuario determinado podrá decidir esperar varios certificados que atestigüen la misma vinculación de claves antes de estar dispuesto a confiar en ella.

En resumen, PGP reconoce que el problema de establecer la confianza es un asunto bastante personal y les da a los usuarios la materia prima para tomar sus propias decisiones en lugar de asumir que todos están dispuestos a confiar en una sola estructura jerárquica de CA.

Revocación de certificados

En el caso de que un certificado haya sido vulnerado, las CA agrega este certificado a una **Lista de certificados revocados (Certificate Revocation List - CRL)** que firma digitalmente. Los usuarios pueden descargar la CRL y verificar si un certificado dado está en ella. Si lo está,

entonces el certificado se ha revocado y no debe confiarse. Además, cada certificado tiene un tiempo de vida, después del cual se considera inválido y son eliminados de esta lista.

17.3.2. Predistribución de claves secretas

Existen entidades conocidas como **Key Distribution Centers (KDC)** que se encargan de distribuir claves secretas. Un KDC es un servidor que comparte una clave secreta con cada participante de la comunicación. Cuando Alice quiere comunicarse con Bob, se comunica con el KDC usando claves asimétricas indicandole sus intenciones. El KCD genera una clave secreta que le envia a Alice y a Bob. Luego Alice y Bob pueden comunicarse usando esa clave secreta.

17.4. Protocolos de autenticación

Reply Attack: Un ataquente retransmite una copia de un mensaje que ya se envió. A pesar de que no es la primera encarnación del mensaje, el autenticador es válido y como no fue modificado puede ser correctamente descifrado. Sin embargo, el mensaje puede no ser válido en el contexto actual. Por ejemplo, si el mensaje es una solicitud de transferencia de fondos, entonces el atacante podría retransmitir la solicitud para que se ejecute dos veces. Necesitamos una solución que provea **Originalidad**.

Una veración de este ataque es el **supress-replay attack** en el que el atacante intercepta un mensaje y lo demora para que sea recibido en algún momento que no sea el apropiado. En este caso, aunque el mensaje es original, no es oportuno. Necesitamos una solución que provea **Timeliness**.

17.4.1. Técnicas de Orginialidad y Timeliness

Un enfoque es incluir una marca de tiempo en el mensaje. La marca de tiempo en sí debe ser a prueba de manipulaciones, por lo que debe estar cubierta por el autenticador. La principal desventaja de las marcas de tiempo es que requieren sincronización de relojes distribuidos. Dado que nuestro sistema dependería de la sincronización, la sincronización de relojes en sí misma debería defenderse contra las amenazas de seguridad, además de los desafíos habituales de la sincronización de relojes.

Otro enfoque es incluir, en el mensaje, un **nonce**, un número aleatorio que se usa solo una vez. Los participantes pueden detectar ataques de repetición verificando si un nonce se ha usado previamente. Desafortunadamente, esto requiere que se mantenga un log de los nonces ya usados, lo que puede ser difícil de escalar.

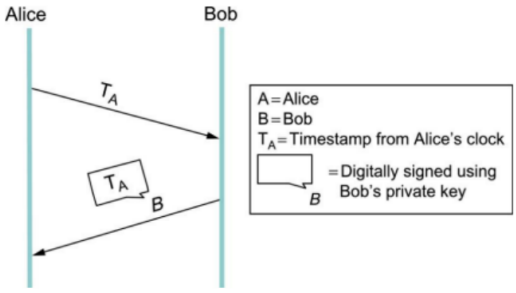

Figura 35: Protocolo response-challenge basado en timestamps

Protocolos de autenticación de clave asimétrica

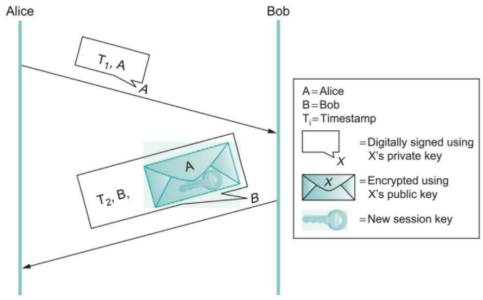

Figura 36: Protocolo de autenticación con clave asimétricas con sincronización de clocks

Alice envía a Bob un mensaje con un timestamp y su identidad en texto plano más su firma digital. Bob usa la firma digital para autenticar el mensaje y el timestamp para verificar su frescura. Bob envía un mensaje de vuelta con una timestamp y su identidad en texto plano, así como una nueva clave de sesión encriptada (para confidencialidad) usando la clave pública de Alice, todo firmado digitalmente. Alice puede verificar la autenticidad y la frescura del mensaje, por lo que sabe que puede confiar en la nueva clave de sesión. Para lidiar con la sincronización imperfecta del reloj, las marcas de tiempo podrían aumentarse con nonces.

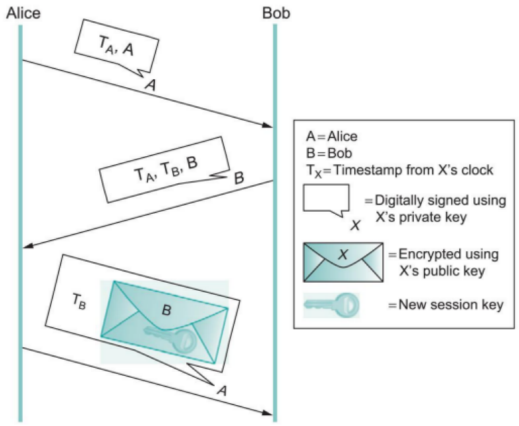

Figura 37: Protocolo de autenticación con clave asimétricas sin sincronización de clocks

En este protocolo, Alice nuevamente envía a Bob un mensaje firmado digitalmente con una marca de tiempo y su identidad. Debido a que sus relojes no están sincronizados, Bob no puede estar seguro de que el mensaje sea nuevo. Bob envía un mensaje firmado digitalmente con la marca de tiempo original de Alice, su propia marca de tiempo y su identidad. Alice puede verificar la frescura de la respuesta de Bob comparando su tiempo actual con la marca de tiempo que se originó con ella. Luego le envía a Bob un mensaje firmado digitalmente con su marca de tiempo original y una nueva clave de sesión encriptada usando la clave pública de Bob. Bob puede verificar la frescura del mensaje porque la marca de tiempo provino de su reloj, por lo que sabe que puede confiar en la nueva clave de sesión. Las marcas de tiempo esencialmente sirven como nonces convenientes, e incluso este protocolo podría usar nonces en su lugar.

Protocolos de autenticación de clave simétrica

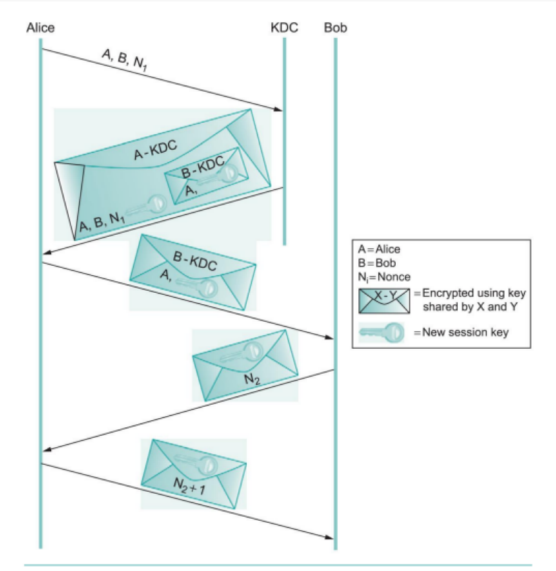

Figura 38: Protocolo de autenticación con KDC

El nonce en los primeros dos mensajes es para asegurarle a Alice que la respuesta del KDC es fresca. Los segundos y terceros mensajes incluyen la nueva clave de sesión y la identificación de Alice, encriptados juntos usando la clave maestra de Bob. Es una especie de versión de clave secreta de un certificado de clave pública; es, en efecto, una declaración firmada por el KDC (porque el KDC es la única entidad además de Bob que conoce la clave maestra de Bob) que la clave de sesión adjunta es propiedad de Alice y Bob.

Notar que el KDC no autentica realmente el mensaje inicial de Alice y no se comunica con Bob en absoluto. En cambio, utiliza su conocimiento de las claves públicas de Alice y Bob para construir una respuesta que sería inútil para cualquier persona que no sea Alice (porque solo Alice puede descifrarla) y que contiene los ingredientes necesarios para que Alice y Bob realicen el resto del protocolo de autenticación ellos mismos.