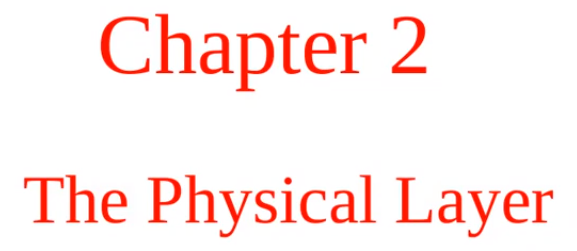
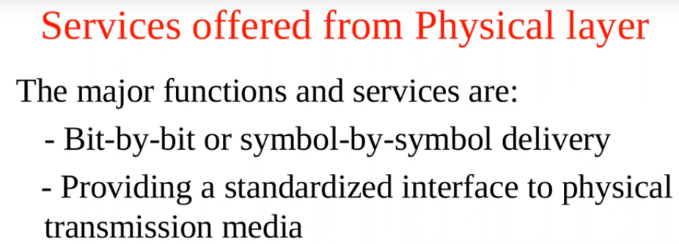
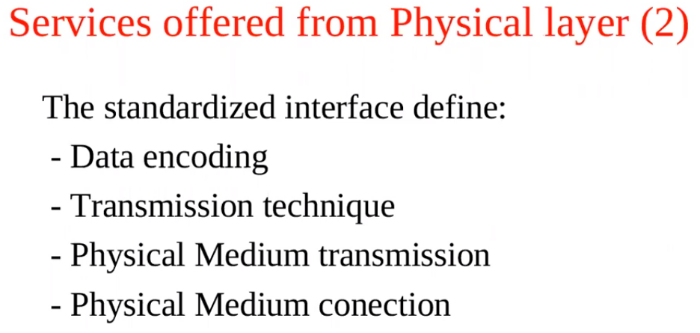
*-Clase del 10-06-2021. “Capa Física Y Enlace”*

Acá conocemos casi todos los temas.



En la capa física viajan bits o símbolos, o luz o una onda electromagnética polarizada o no.

La función o servicio que brinda la capa superior, la de Enlace, es la transmisión o entrega de estos bits, y de alguna manera proveer una interface física común estandarizada. Ej: modulación, tipo de cable y ficha o conector, etc.

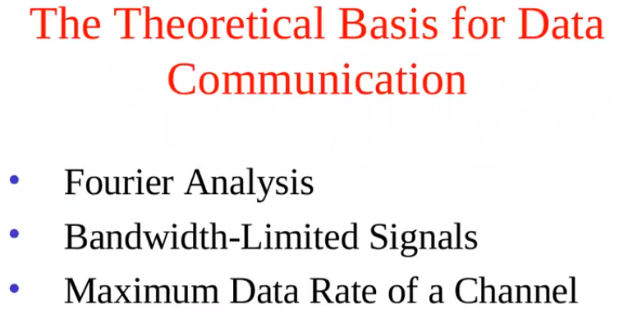
La interface define:

-Que codificación usar

-Que técnica de transmisión. Ej: técnica de modulación

-Medio físico de transmisión. Ej: aire, cobre, fibra óptica

-Conexión. Ej: conector, ficha, cantidad de pines, niveles de voltaje, etc.

 Relacionado a temas de comunicaciones, que se ven en otras materias.

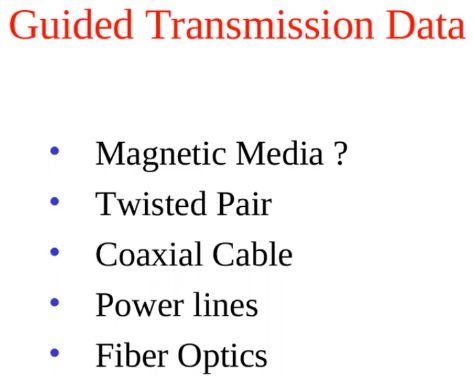
¿Qué ancho de banda puedo tener??

¿Hasta qué velocidad puedo transmitir en un canal determinado?

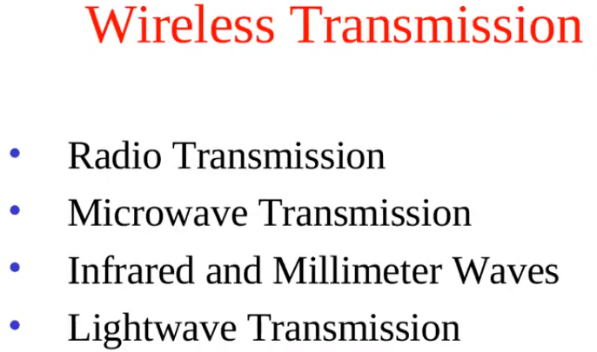
Hay ecuaciones que se pueden aplicar de Sistemas de Comunicaciones para ver cuál sería la tasa máxima de transmisión.

Todo esto se ve en capa física.

***Distintos canales posibles de comunicación.***

Hay distintos medios de transmisión físicos.

Par trazado

 Hay distintos métodos de transmisión Inalámbricos.

Radio

-Micro ondas

- Infra – milímetro Raroooo?

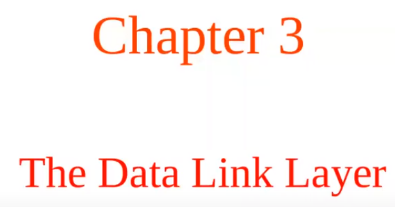
-Enlaces con Modulación de Luz …. ¿Inalámbrico? ¿Será algo usando láser?

De largo alcance usando ondas electromagnéticas.

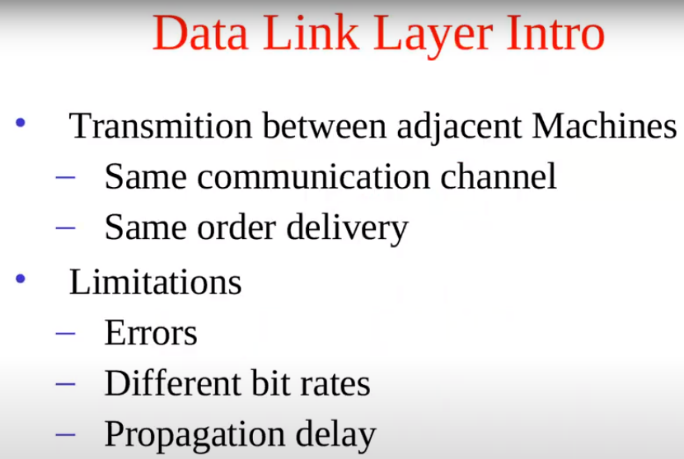
Satélites.

Básicamente el capítulo habla genéricamente de estas cosas, asique el profe prefiere saltearlo. Y arrancar utilizando la capa física. O sea, viendo que tomaría la Capa de Enlace.

***“El servicio que brinda la capa física es transmitir de un extremo a otro. De la entrega de bit por bit, o símbolo por símbolo. Y toma que si estandarizo la capa física no importa el fabricante de los equipos o placas, porque todos vas usar el mismo tipo de modulación, niveles de señal, etc.”***



“CAPA DE ENLACE”

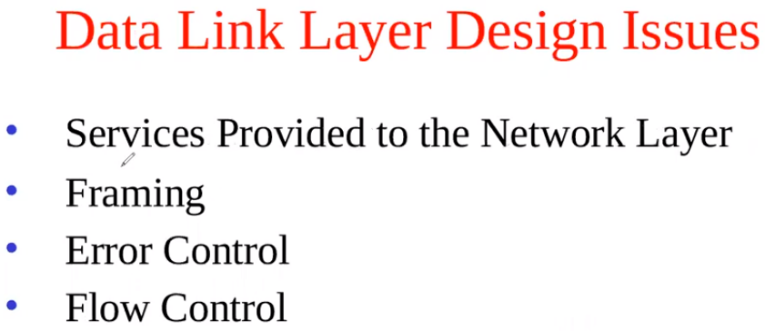
Esta capa se encarga de la transmisión entre máquinas de la misma RED.

Ósea, van a usar el mismo canal de comunicación, la misma tecnología por la definición de RED, y al estar la información en un cable, la entrega de la misma seria ordenada. Ej: sale un bit A llega un bit A, sale un bit B llega un bit B, NO SE MESCLARIAN.

Problemas que la capa física no resuelve, o no nos brinda ese servicio:

* ¿Qué pasa si tengo errores?
* ¿Qué pasa si el Emisor y Receptor tienen velocidades distintas?
* ¿Tengo en cuenta la el retardo de propagación?

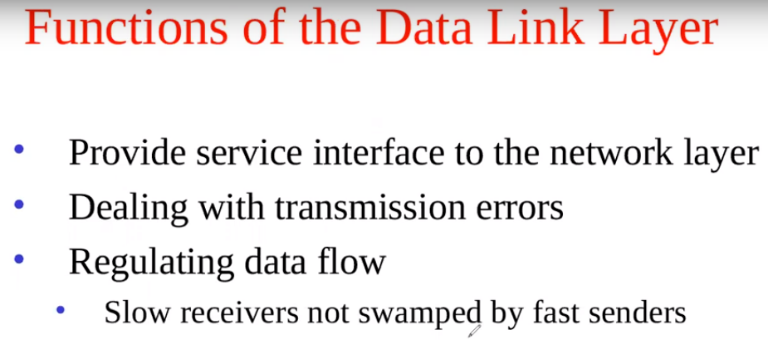
Hay varias situaciones en las que vamos a profundizar, que la capa de Enlace debería solucionar para brindarle un servicio a la capa superior, la de Red.

¿Qué servicios le vamos a brindar a la capa de Red?

-Entramado.

-Control de errores: Podría ser detección o corrección. Gestión de errores.

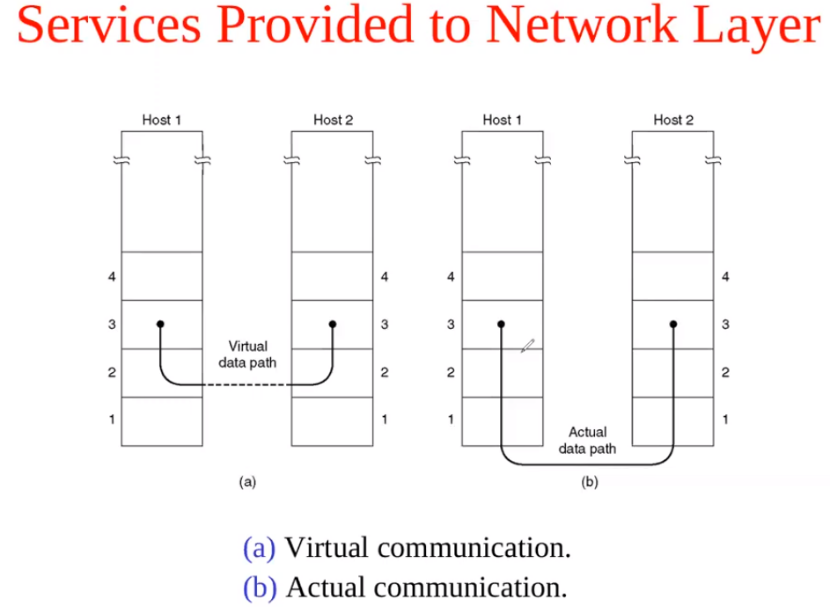
- Control de flujo.

Funciones de la capa de Enlace:

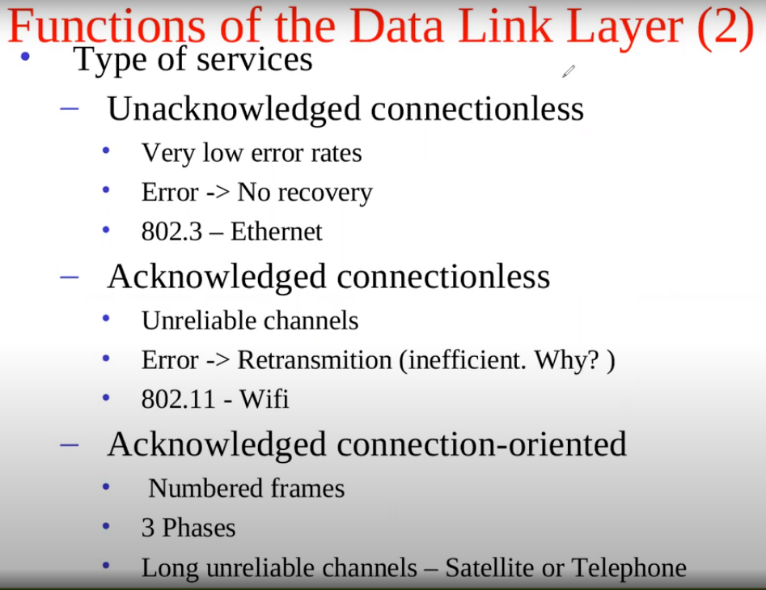
-Brindarle los servicios mencionados arriba a la capa de red.

-Debería ver como gestiona los errores

-Como maneja el control de flujo: Ósea si tengo un receptor lento, debería de alguna manera hacer que el emisor no lo aturda, no lo abrume o agobie.

 Camino de la comunicación real

Virtualmente la Capa de Red entiende que le entrega sus datos a la capa de enlace y la capa de enlace llega de extremo a extremo. Virtualmente la capa de red no sabe cuántas capas más hay abajo, solo que le pide servicio a la de enlace y listo.

Esto estamos hablando entre 2 entidades de la misma capa de enlace

De acuerdo al tipo de servicio, la C. de Enlace Puede ser:

**-Sin confirmación (acuse de recibido), sin conexión.**

Esto se aplica a capas que tienen una baja tasa de errores, entonces si ocurre un error no se hace nada. Depende de la aplicación y del medio.

Tipo de enlace. Y de en qué capa me conviene dar un servicio o no, Puede ser que la gestión de error conviene hacerlo en una capa superior.

Mando el Dato nada más, no hay handshake o inicialización de la comunicación, tampoco hay confirmación del receptor. Se usa en canales con baja tasa de error.

Ej: del teléfono fijo. Ethernet. Y 802.3

**-CON confirmación (acuse de recibido), sin conexión.**

Para canales no muy confiables. Si hay un error el Emisor retransmite. No es muy eficiente según el autor, porque si parto un mensaje grande en pedacitos (tramas), y la confirmación es sobre todo el mensaje completo, tengo que reenviar todas las partes de nuevo, no solo el pedacito que llego con error. Ej: Wifi

**-Con confirmación (acuse de recibido), Orientada a conexión**.

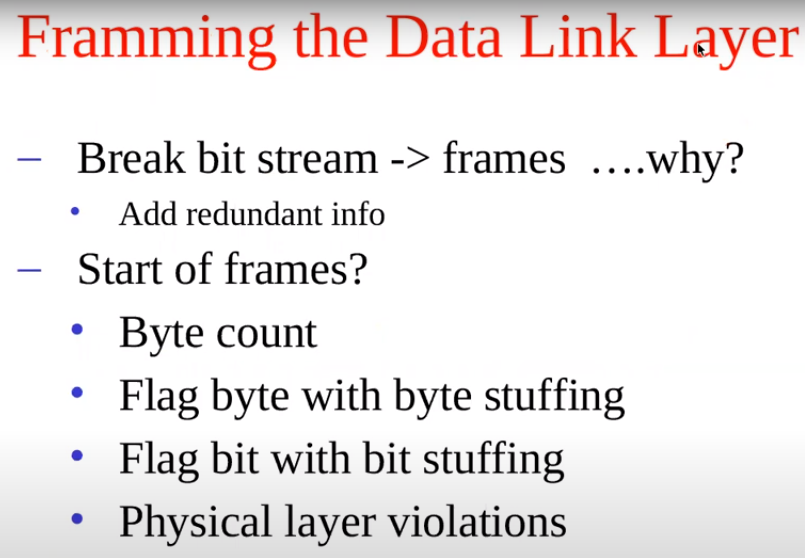
Para evitar la ineficiencia del caso anterior, cada trama esta numerada, si hay error retransmito solo esa.

Esto es bueno para canales no confiables y con un alto diley, canales largos. Con lo cual retransmitir todo es perder mucho tiempo.

Obviamente ante de mandar los datos hay que entablar o inicializar la comunicación, hacer un handshake

-Sin confirmación (acuse de recibido), Orientada a conexión.

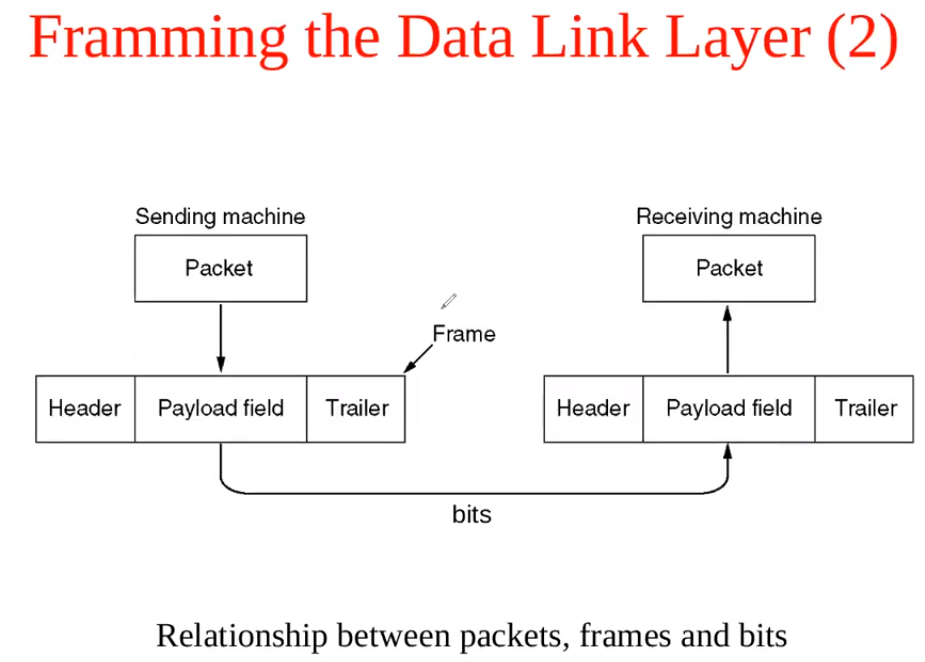
Esta combinación no tiene sentido.

Vamos a ver el 1er servicio que brinda a la capa superior la Capa de Enlace.

Entramado o Framming: Básicamente parte un mensaje en pedacitos llamados tramas. Y agrega información redundante, que puede ser de control de flujo o gestión de errores.

¿Por qué lo parte?

Si tengo un mensaje que es muy largo, no sé cuándo empieza y termina. Como hago para agregar un Head y un Trailer sino conozco el inicio y el fin. Como le agrego la información que me sirve para bríndale los servicios a la capa superior.

Entonces lo parte en pedazos y a cada uno le agrega información de control. Ya no depende del tamaño del mensaje. Ej un mail o compartir un DVD.

¿Cómo se cuándo inicia una trama? Hay barias formas.

-Conteo de “Byte”: Es malo no se usa, llego a tener un error, se corre y no se sincroniza más.

- “Byte” bandera: es más usado.

- ***bit*** bandera:

-Violaciones de la capa física.

En cada capa la información o mensaje tiene un nombre:

En capa de RED, el mensaje se llama Paquete.

En capa de Enlace, hablamos de tramas o frames.

**Conteo de Byte**: Las tramas no tienen siempre el mismo tamaño

Tiene un encabezado, que tendría un byte, que me indica la cantidad de caracteres de la trama.

El Problema: es que pasa si tengo un error en el encabezado. Se pierde la sincronización entre encabezados y los que es información, en el destino la capa de enlace lee y pasa a la capa de red cualquier cosa, no el mensaje correcto.

**Byte de Bandera**: Cada trama arranca y termina con un Byte que tiene un valor predeterminado. Siempre el mismo.

¿Qué pasa si dentro del Payload hay un byte que tiene la misma combinación de bits que el byte bandera?

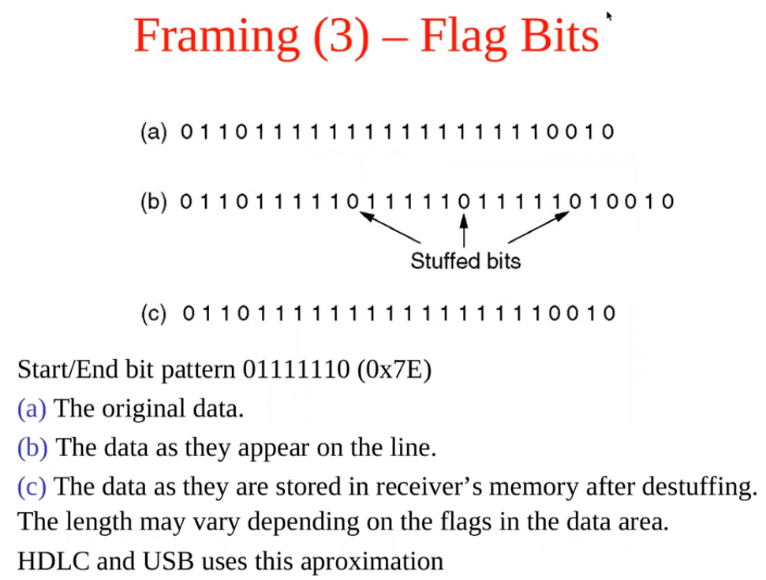


Caracteres originales (dentro del Payload). Ej: 3 byte

Lo que hace el emisor antes de transmitir todos los bytes es agregar un byte de escape, transmite 4 en vez de 3. Para que el receptor, si ve un escape sepa interpretar que el siguiente es un byte de información y no una bandera.

After Stuffing = después de rellenar

Los esc que lee el receptor los descarta, indican que el siguiente byte es dato.

El tema acá, es que si un Byte luego de la transmisión, por error se cambia algún bit y ahora el receptor lo detecta como una bandera. Piensa que termino …. Lo de más lo detecta como ruido hasta la siguiente bandera, por lo que se pierde parte de la información de esa trama, pero se vuelve a sincronizar con la siguiente bandera. No se desincroniza para siempre como el anterior.

Una crítica a este protocolo es que agrega muchos más bytes de control. Tiene más overhead, menos eficiencia.

Ej: Protocolo PPP

**“Bits” de Bandera:** Cada trama inicia con un BYTE, igual que el anterior, la diferencia es que en vez de realizar un relleno de Bytes, rellena con bits.

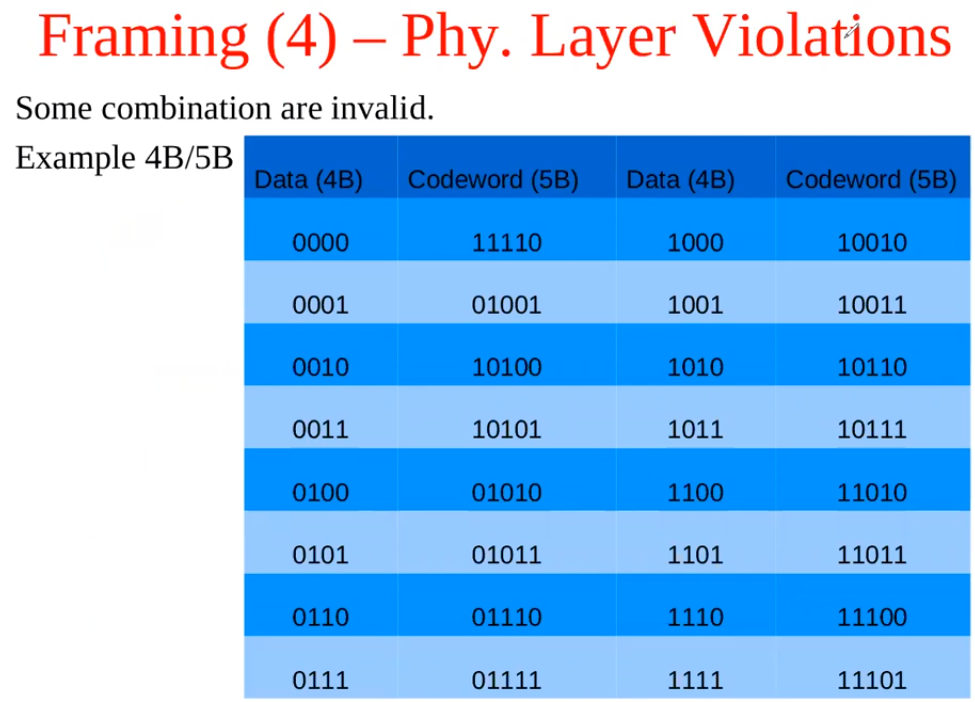
Cuando en el Payload (paquete que viene de la capa de red) encuentra 5 bits seguidos en 1, le agrega un 0, para que no halla 6 como en la bandera. En el receptor saca el 0 cada 5 unos.

Este es más eficiente, tiene menos overhead. Se agregan bits en vez de bytes de redundancia, pero otra vez si tenes un ruido que lleva al receptor a detectar la bandera por error, entonces corta antes o se pierde la trama, pero con la siguiente se vuelve a sincronizar.

En este y en el anterior, la longitud del mensaje puede variar dependiendo de la cantidad de los bits o byte de relleno que se agreguen.

No hemos visto todavía el tema de error, solo overhead y sincronización.

En la capa de enlace, originalmente, yo quiero dar un servicio de entramado porque justamente se desconoce el largo del mensaje que viene de la capa de red, y quiero hacer un control de errores, entonces no voy a esperar a mandar todo el mensaje para ver si está mal o bien, la idea es partirlo en pedacitos, tramas, e ir controlando cada uno de esos. Pero esto implica saber dónde empieza y termina cada pedacito, entonces necesito agregarle algo que me marque el inicio de las tramas.

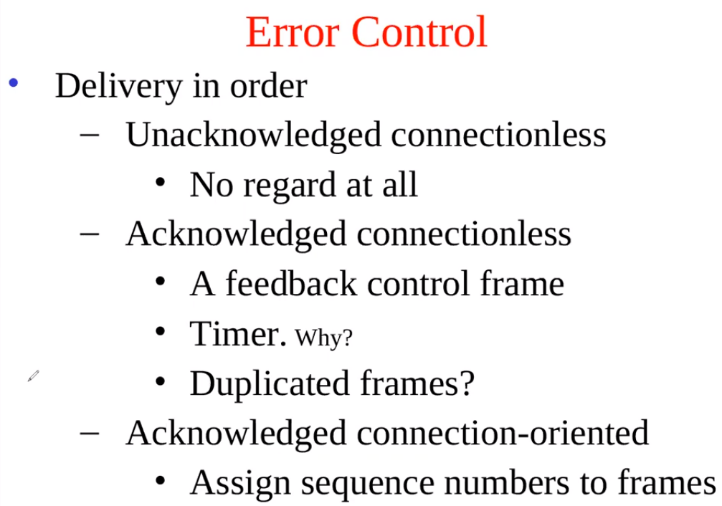
**Violaciones de capa física:** se basa en combinaciones de Bits que no podrían existir. Este es medio ineficiente.

EJ: una combinación de 4 bits la codifico con 5 bits, entonces algunas combinaciones no van a ser validas, y esos las utilizo como inicio o fin de trama.

4bits = 16 combinaciones

5 bits = 32 combinaciones: 16 combinaciones se usan para representar dato y las otras 16 no se usan. Una de esas puede ser la bandera.

Acá desperdicio por cada byte 2 bits.

En cuanto a control de error, si bien la entrega es ordenada, porque si alguien manda una trama, no se mescla en el camino, hay un único medio, canal o camino. En el caso de Red si pueden tomar distintos caminos, y hay que ordenarlos, en capa de enlace no.

Cada trama que se envía se recibe en el mismo orden.

**-Sin confirmación (acuse de recibido), sin conexión.** No tiene control de flujo. Y el emisor no detecta nada en cuanto a los errores que le llegan al receptor.

**-Con confirmación, sin conexión**. Acá si puedo hacer control de flujo. ¿Cuánto espera el emisor antes de reenviar la trama si no hubo confirmación? Y si tengo una confirmación yo podría guardar los datos en un buffer de salida, y hasta que no me diga que llego bien no lo descarto. Utilizo un timer.

¿Cómo hace control de flujo?: Por lo general se utilizan dos métodos. En el primero, el **control de flujo basado en retroalimentación**, el receptor regresa información al emisor autorizándolo para enviar más datos o indicándole

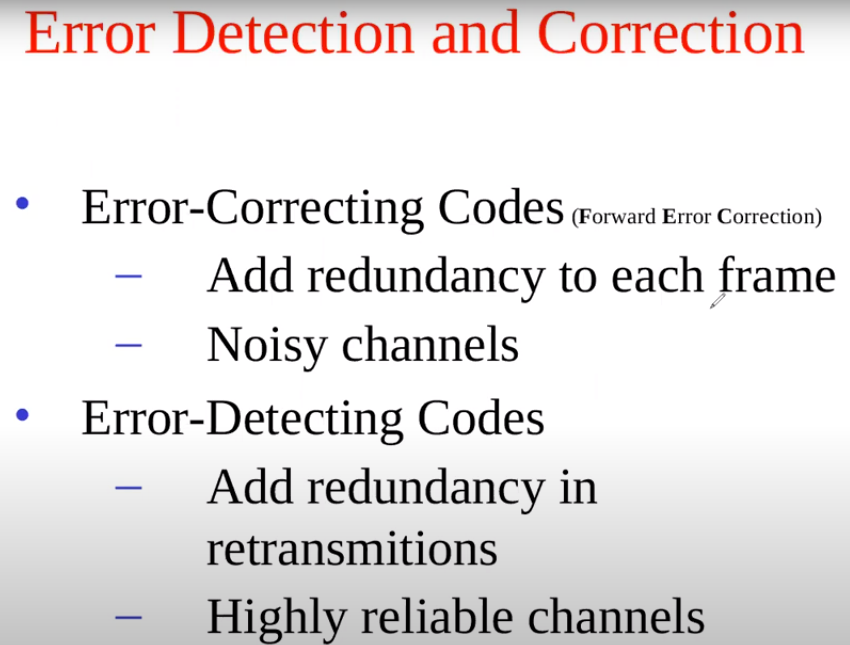
su estado. Ejemplo de esto puede ser Ventana deslizante, fija (para aprovechar mejor el canal - BW) o variable en el tiempo (agrega la funcionalidad del control de flujo)

En el segundo, el **control de flujo basado en tasa**, el protocolo tiene un mecanismo integrado que limita la tasa a la que el emisor puede transmitir los datos, sin recurrir a retroalimentación por parte del receptor. Ejemplo de esto puede ser el preámbulo de Ethernet, donde la onda cuadrada sincroniza los host destino y origen.

¿Qué pasa si lo que se perdió fue la respuesta de confirmación del receptor? Entonces reenvió el mismo mensaje y en el receptor llega duplicado.

**-Con confirmación, orientado a conexión**:acá de alguna manera Emisor y receptor se ponen de acuerdo, y las tramas se enumeran. Y acá el emisor solo retransmite la que no llego al receptor.

Esto es el mejor de los casos, pero tiene un costo en cuanto a mensajes de control y overhead. Es menos eficiente en el uso del canal, mando mucho byts de control y menos de información.

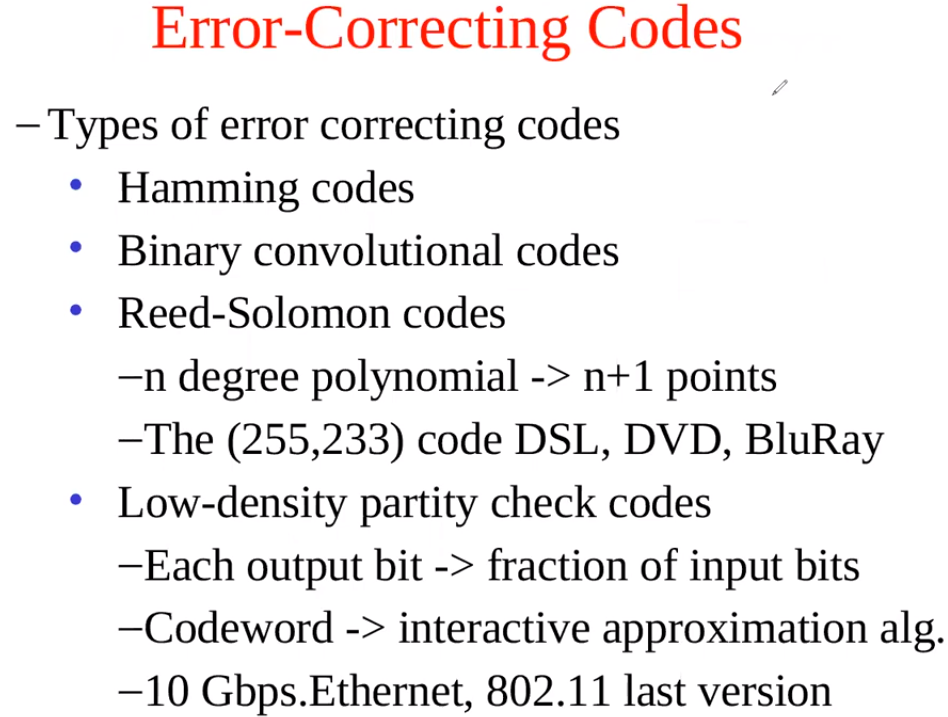
Adicionalmente, podríamos tener detección y corrección de errores.

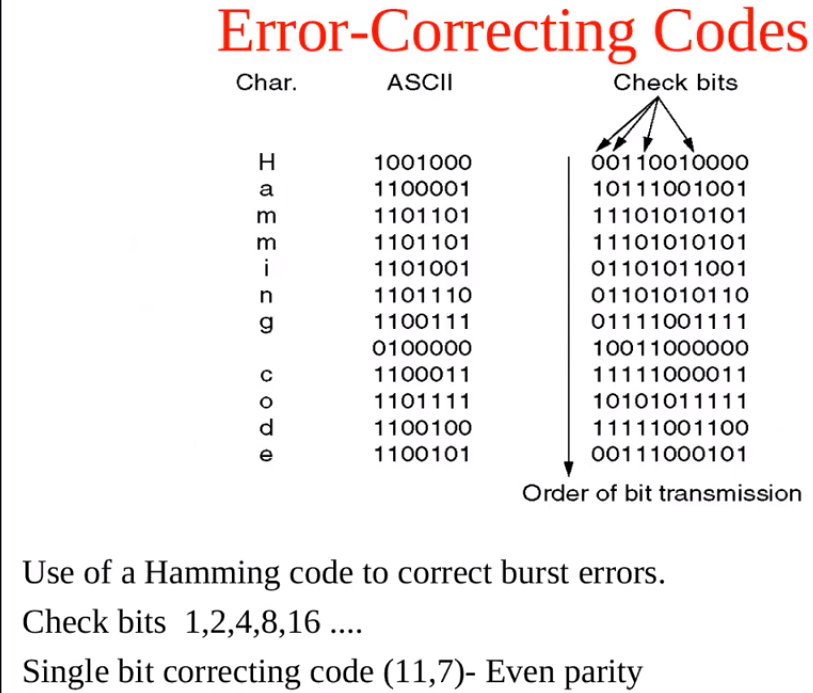
Se añade redundancia, ósea bytes de control a la información que se envía.

La corrección de errores se usa cunado tengo canales muy ruidosos, y que generalmente tengo errores. Lo que se hace es siempre añadir información de control que le permita arreglar los errores al receptor y no tener la necesidad de re transmitir, porque es probable que vuelva haber un error.

En canales más confiables o con menor tasa de error, es preferible usar detección, si hay un error el receptor avisa y se retransmite todo. Porque como la probabilidad de error es baja, vale la pena retransmitir todo lo que está mal, pero una vez cada 1000 tramas, por ejemplo.

Hay que verlo desde el punto de vista de la eficiencia.

Hay varios códigos de corrección de error. El libro explica algunos, otros solo los presenta.



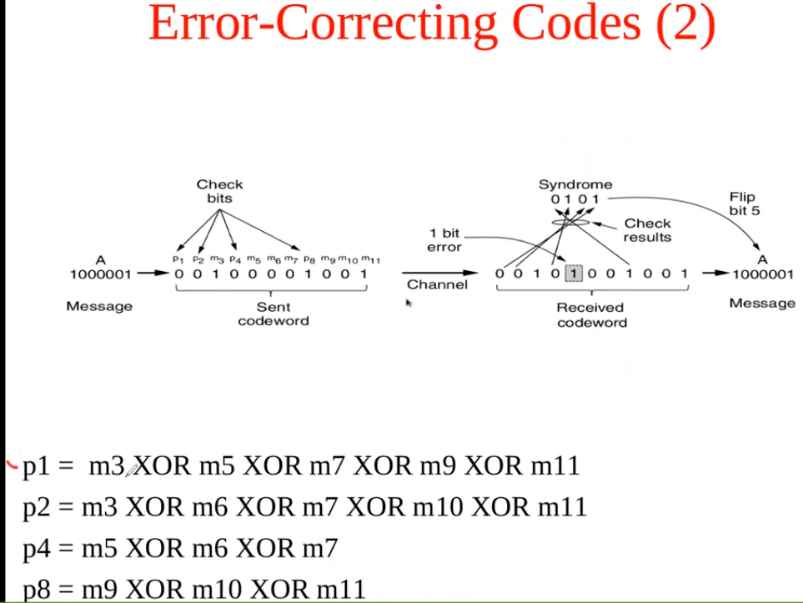
**-Código de Hamming** básicamente siempre agrega información adicional.

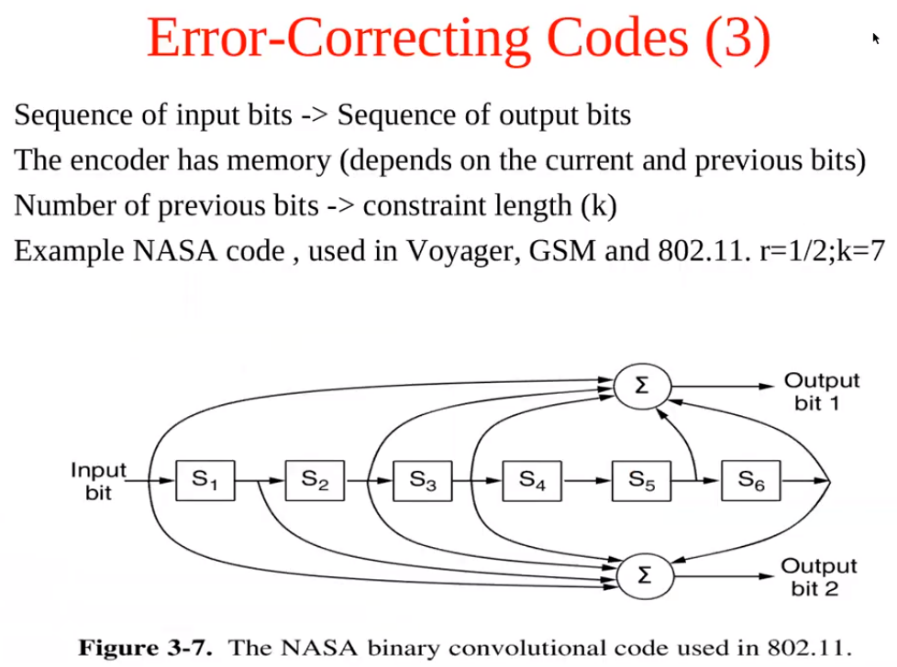
Ej: (11,7) manda 7 bits de datos y agrega 4. Los cuales tienen un peso, no se agregan en cualquier lado.

Los bits agregados se completan en base a operaciones binarias entre los bits de mensaje.

En el emisor corro el algoritmo y completo los bits agregados. Y el receptor corro el mismo algoritmo y comparo con los bits, si esta todo bien debería darme 0, caso contrario me da la posición del bit con error.

No es necesaria una retrasmisión.



**-Código convolucional binario:** a diferencia de otros que trabajan sobre bloques de Byts, este trabaja sobre bits individuales.

Tiene registros de desplazamiento. Memoria. La salida de un bit depende de la entrada, pero también de los estados anteriores.

Se uso bastante.

Códigos correctores de errores se justifica para lugares ruidosos, o mensajes caros.

Estos 2 códigos se implementan sobre hardware.

Algunos basados en temas matemáticos.

-**Red Salomon**: Básicamente define un polinomio de grado “n”, entonces lo que se hace es contrastar contra eso.

Se usa sobre medios de cobre o cuando se rayan los CD.

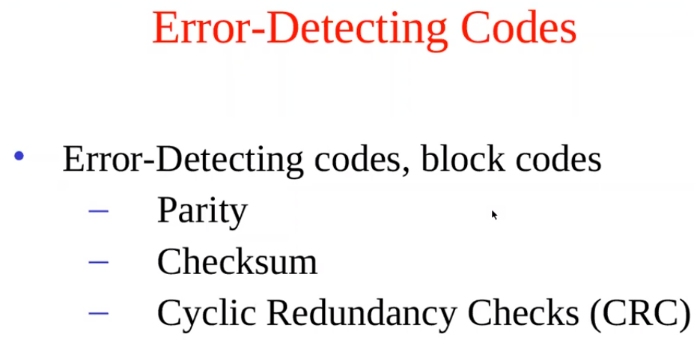
-**Código de chequeo de paridad de baja densidad.**

Básicamente se arma como una matriz, y de todos los bits que entran a la matriz solo una fracción salen. La matris se llama Codeword y utiliza un algoritmo de aproximación iterativa.

El profe dijo:

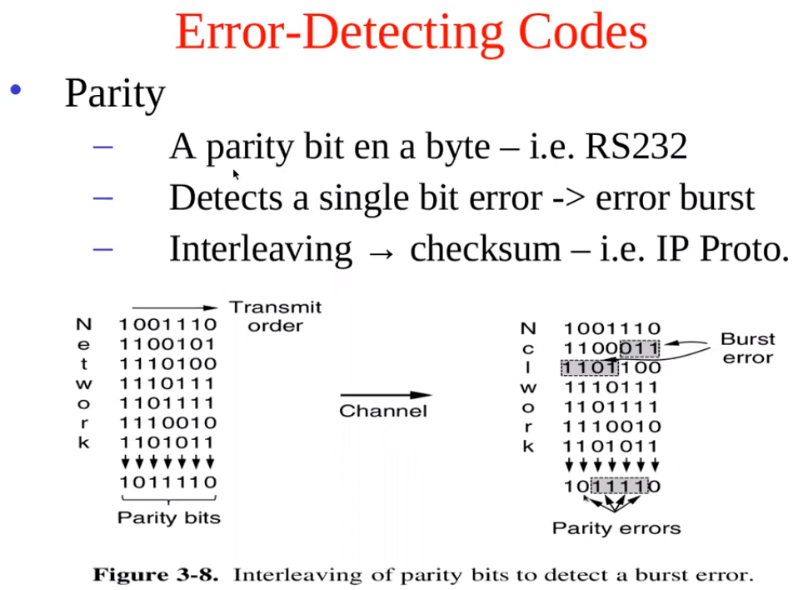
NO tiene sentido profundizar meterse en detalle, porque tienen un componente importante de temas matemáticos.

Se destaca que es para corrección de errores y ejemplos de aplicación.

** Códigos detectores de Errores:**

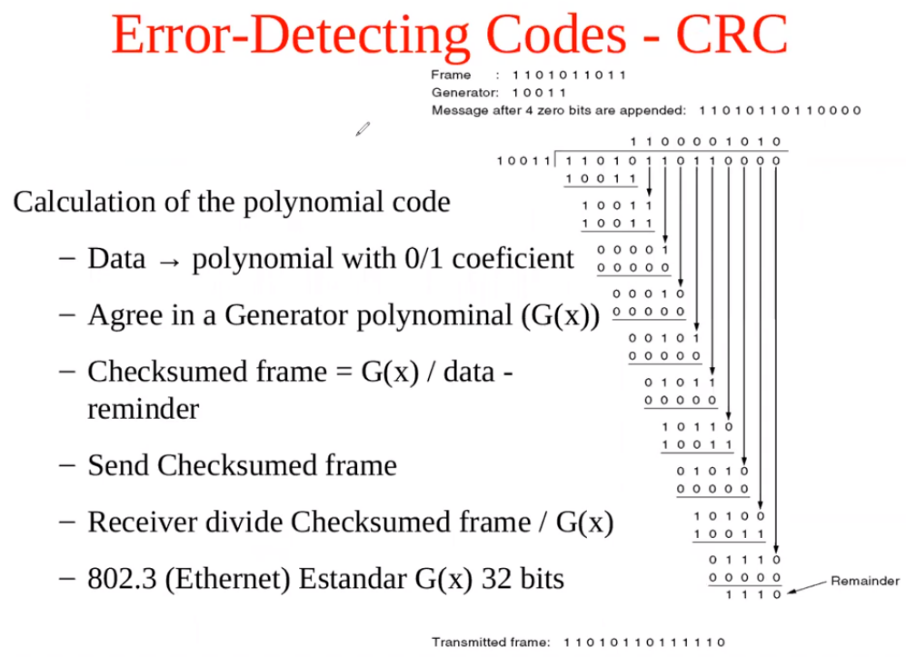
No corrige, a lo sumo no manda la confirmación, o informa que llego mal para no esperar el timer de retransmisión del emisor. No más de eso.

-Paridad: siempre es por bloque, a un byte le agrego un bit de paridad par o impar.

El tema es que normalmente no cambia un bit, sino una ráfaga de bits. Entonces podría ser que me cambie algunos bits pero me dé bien la paridad.

Checksum: Entonces una manera de mejorar esta detección es hacer este chequeo de paridad, pero en un bloque más grande de un Byte.

El emisor manda la información y el resultado del checksum y el receptor lo calcula y compara si le dio igual o no.

**Código de Redundancia cíclica:** Básicamente lo que hace es operaciones matemáticas. Se define un polinomio de grado “n” con coeficientes 0 y 1.

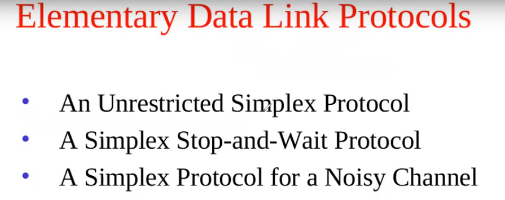
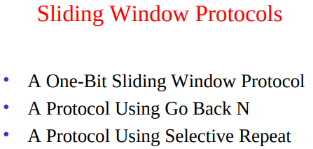
Tomo el mensaje y le agrego una cantidad de ceros igual a la cantidad de bits que tiene el polinomio y hago una operación de división.

El resto generado lo mando al final junto con el mensaje. En el receptor hago la operación inversa y debería dar cero.

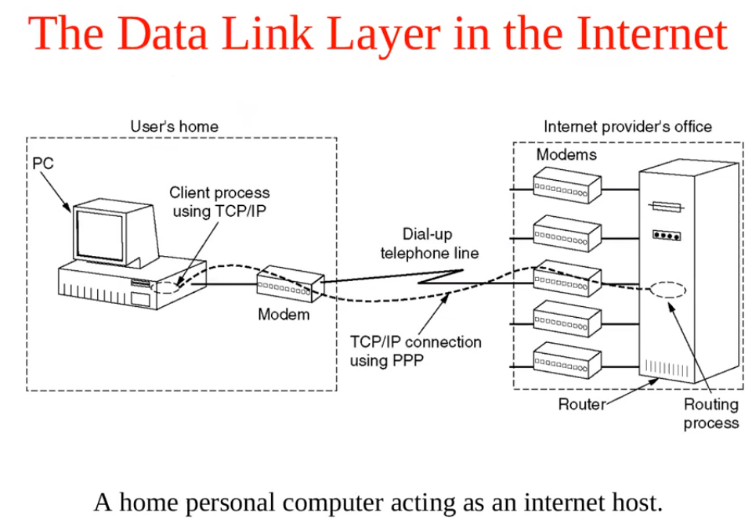
El polinomio es conocido por el emisor y el receptor.

Se usa para bloques de información más grandes que la paridad o el checksum, ej: la trama Ethernet tiene 1kB

Acá tengo que entrar a una tabla, o hacer operaciones, si lo hiciera por cada byte sería muy lento.

Acá hay una parte que no se ve de este capítulo. Porque se repite en otros capítulos.

salta varias filminas el profe, algo de una ventana. Confirmado que estos temas no van.



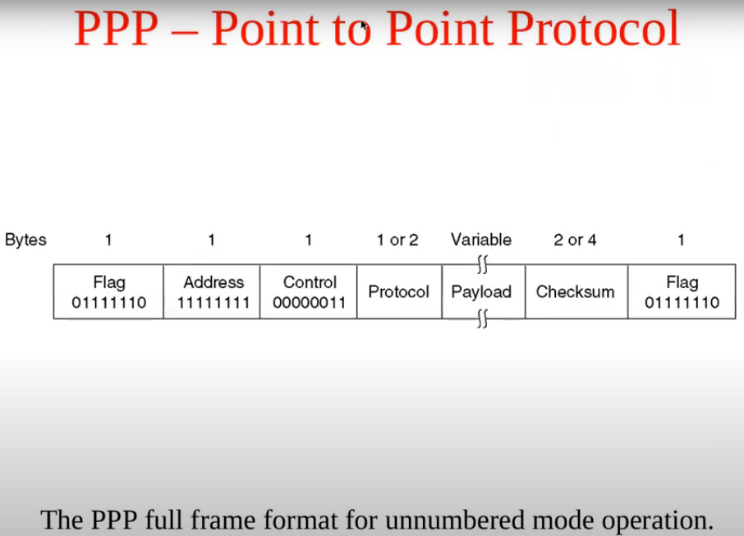
Caso de uso de la capa de enlace. Un protocolo de capa de enlace PPP. Que cosas usa, como es el formato del protocolo.

Si bien PPP se usa para capa de enlace, también tiene unos bits que son de control que también sirven para otras capas. Pero nosotros nos enfocamos en capa de enlace.

-Inicio y fin de trama.

-Que hace con los errores.

Este protocolo en las viejas épocas nos conectaba a internet.

Utiliza un inicio de trama que es 7E, que una bandera de bits. Utiliza relleno de bits no de Bytes.

Utiliza solo detección de errores, no corrección. Usa Checksum

Otra cosa que la capa de enlace debería solucionar a veces, no siempre, es como se a quien le mando los datos. Si tengo varias máquinas en la misma red. En PPP solo se conectan 2 máquinas, asique el campo dirección se pone todo en alto, no interesa.

**Protocol**: Me dice a qué protocolo de la capa de RED le estoy mandando los datos, hay varios. Y si el emisor tiene uno y el receptor otro hay problemas.

Generalmente siempre vamos a encontrar, que un protocolo tiene alguna información de a que protocolo de capa superior le tiene que llevar los datos.

Hasta acá solo hemos hablado de protocolo que son punto a punto, podría ser que en una red tenga más de 2 máquinas, por lo tanto, un tema a ver son las direcciones, como se a quien le mando los datos. Hay varios temas para redes de difusión, ósea que no son punto a punto, en capa de enlace.

Esto se llama Sub capa de acceso al medio. Hay que ver toda la problemática de acceder a un medio compartido. Esto dentro de capa de enlace, un ej es Ethernet.

**Campo “Control”:** cuyo valor predeterminado es 00000011. Este valor indica una trama no numerada. En otras palabras, PPP no proporciona de manera predeterminada transmisión confiable usando números de secuencia y confirmaciones de recepción

PPP realiza detección de errores, soporta múltiples protocolos, permite la negociación de direcciones de IP en el momento de la conexión, permite la autenticación y tiene muchas otras funciones.

PPP proporciona tres características:

1. Un método de entramado que delinea sin ambigüedades el final de una trama y el inicio de la siguiente. El formato de trama también maneja la detección de errores.
2. Un protocolo de control de enlace para activar líneas, probarlas, negociar opciones y desactivarlas ordenadamente cuando ya no son necesarias. Este protocolo se llama LCP (Protocolo de Control de Enlace). Admite circuitos síncronos y asíncronos y codificaciones orientadas a bits y a caracteres.
3. Un mecanismo para negociar opciones de capa de red con independencia del protocolo de red usado. El método escogido consiste en tener un NCP (Protocolo de Control de Red) distinto para cada protocolo de capa de red soportado.