Nivel 2 ISO/OSI (Link level)

- **Servicio utilizado: Nivel físico, transmisión de bits**
- **Servicio ofrecido: Transferencia de bloques (frames) de bits entre dos equipos conectados a través de un vínculo físico**
- **Variantes del servicio:**
  - **Orientado o no orientado a la conexión**
  - **Confiable o no confiable**
- **Combinaciones clásicas:**
  - **Orientado a la conexión, confiable**
  - **No orientado a la conexión, no confiable**
- **Protocolos más comunes**
  - **PPP (point to point protocol)**
  - **LLC (Logical Link Control)**
- **Dividido en dos subniveles cuando aparecieron las redes broadcast (MAC y LLC —IEEE 802)**

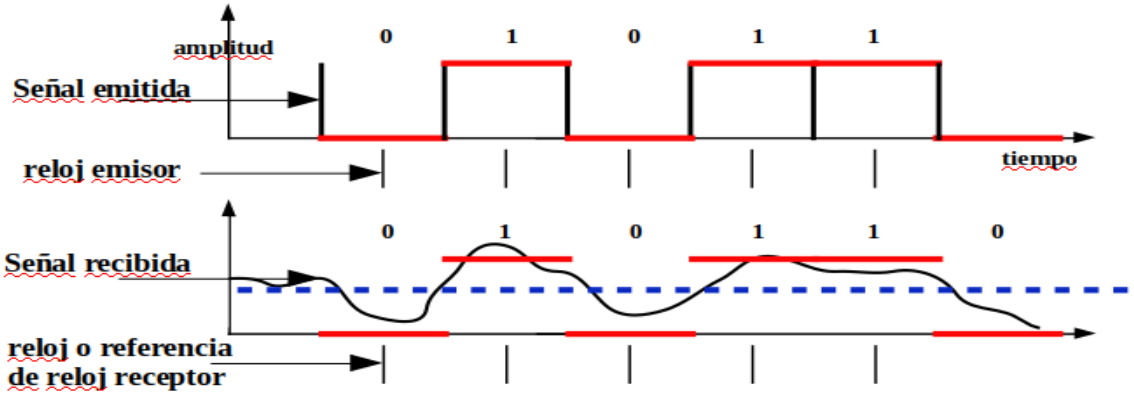
Características generales de los protocolos de nivel 2

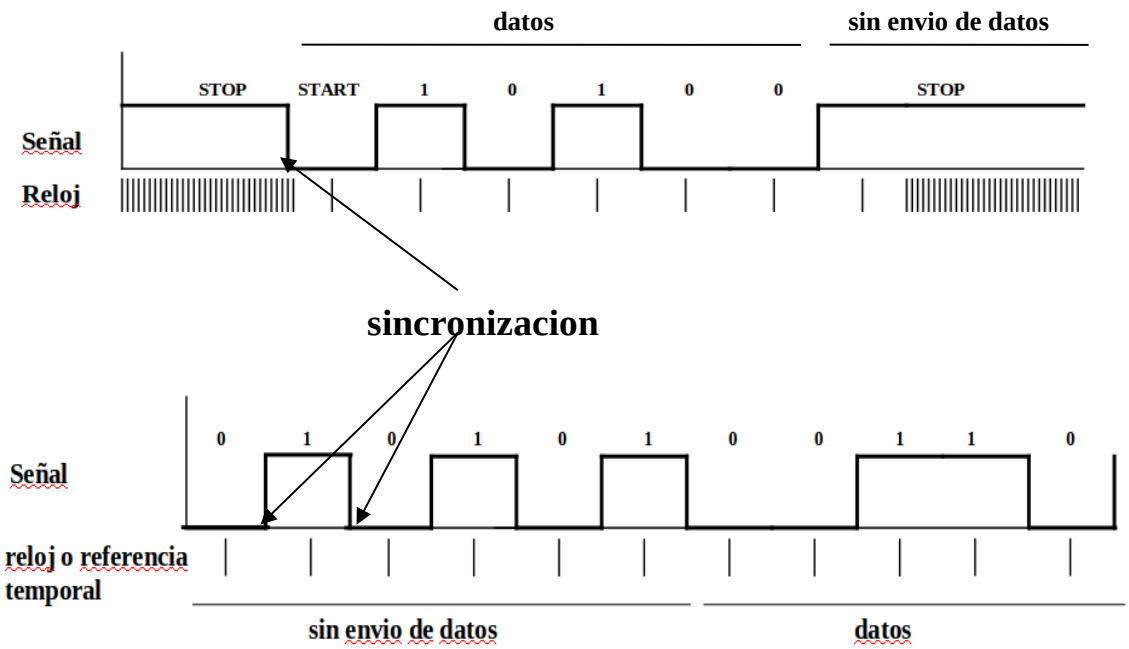
- Independientes del código
- Soportan transmisión sincrónica y asincrónica
- Eficientes
- Funciones en hardware
- Frames con información de control posicional
- Derivan de HDLC (High level data link control)

Funciones del nivel 2

- **Configuracion inicial del link**
- **Direccionamiento**
- **Delimitación de bloque**
- **Control de errores**
- **Control de flujo**

Transmision de bits





Transmision asincronica
-Cada caracter se transmite independiente mente
-Canal en reposo, senal alta (stop)
-Sincronizacion: start
-Reloj en emisor y receptor
-Relojes independientes

Transmision sincronica
-Los bits se transmiten continuamente
-Canal en reposo, se envia secuencia de pulsos altos y bajos (0 y 1)
-Sincronizacion: en cada transicion
-Reloj en emisor y receptor, sincronizados
-Reloj en emisor, el receptor sincroniza con info de reloj enviada en la senal

Delimitacion de bloque (Framing)

- **Permite distinguir grupos de bits (frames) que son tratados como una unidad a nivel 2**

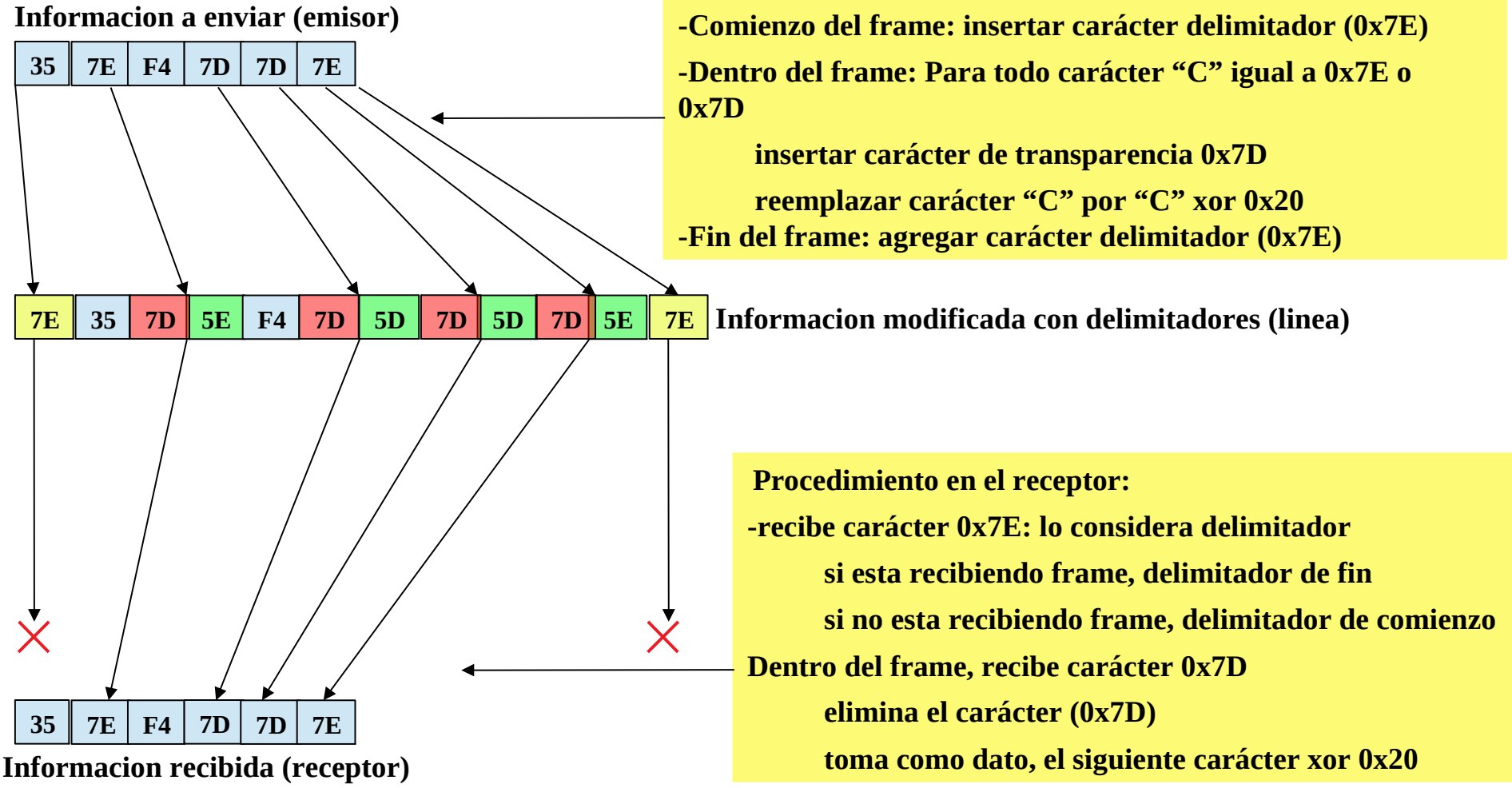
- **Alternativas**
  - **Delimitacion por caracteres**
    - **Se indica comienzo y fin de frame con un carácter -delimitador-**
    - **Mecanismo de insercion de carácter (character stuffing): permite enviar el carácter delimitador como informacion.**
  - **Delimitacion por secuencia de bits**
    - **Se indica comienzo y fin de frame con una secuencia de bits -delimitador-**
    - **Mecanismo de insercion de bits (bit stuffing): permite enviar como dato la secuencia de bits de delimitacion**
  - **Delimitacion por senales especiales (Violación de código)**
    - **Se indica comienzo y fin de frame con señales que no son ni cero ni uno**
    - **Es necesario que el metodo de codificacion de bits en senales lo permita**
  - **Delimitacion por cuenta de caracteres**
    - **Se destina un campo en el frme para indicar la longitud del mismo**
    - **Solo se usa como complemento de los metodos anteriores**
    - **Se arrastra la perdida de sincronismo en frames consecutivos**

Delimitacion por caracteres

- Utilizado en transmisión asincrónica
- Overhead significativo
- Por ejemplo, delimitacion utilizada por PPP en líneas asincronicas:

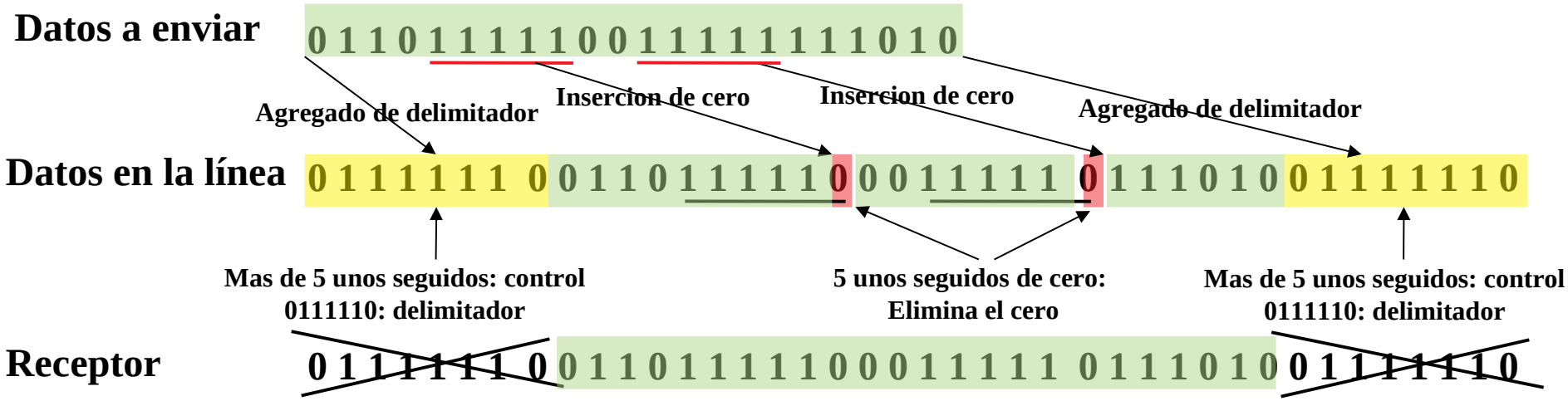
Delimitador: 7E

Relleno: 7D



Delimitacion por bits: inserción de cero

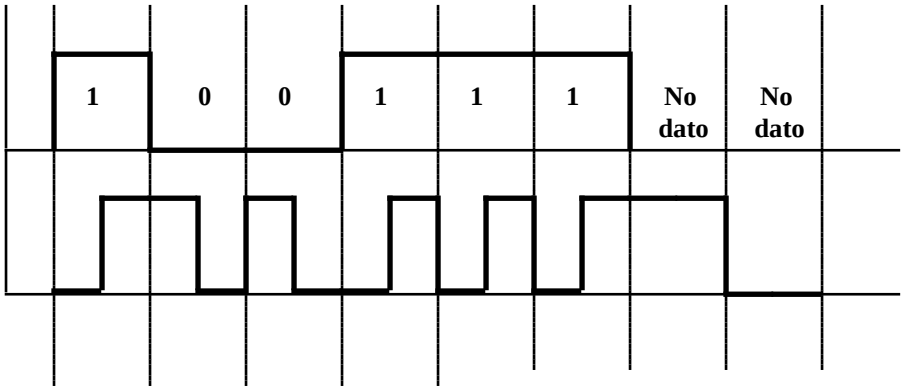
- Utilizado por PPP en líneas sincrónicas
- Bajo overhead
- Eficiente (implementado en hardware)
- Delimitación de bloque: Flag 0 1 1 1 1 1 1 0
- Transparencia: Mecanismo de inserción de cero
  - Emisor
    - En modo datos, cuando detecta 5 unos seguidos, inserta cero
  - Receptor
    - Cuando detecta cinco unos seguidos, si el siguiente es cero, lo elimina
    - Más de cinco unos seguidos, significado especial (p.ej. flag)



Delimitación por violación de código

- **Violación de código**
  - **Aplicable en los casos en que la codificación de bits en señales contiene redundancia (p.ej. Manchester en Ethernet de 10Mbps)**

Ejemplo: Manchester



Control de errores

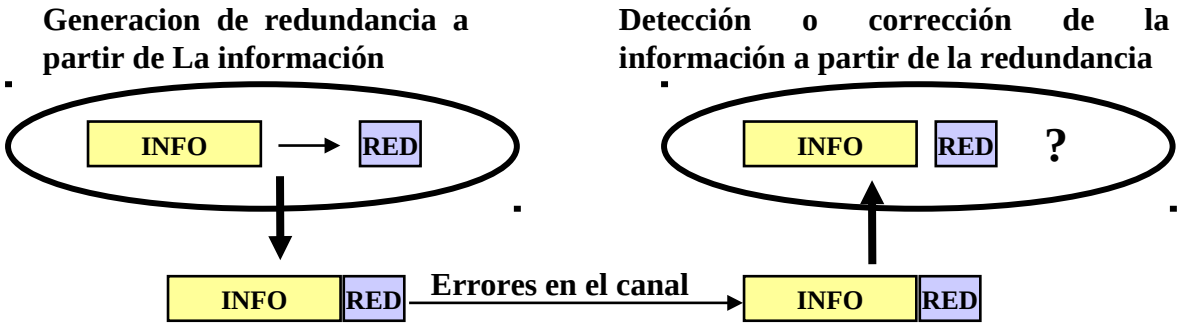
- **Objetivos**
  - **Detectar errores: impide que datos erroneos sean procesados por las aplicaciones**
  - **Corregir errores: permite que las aplicaciones puedan disponer de los datos necesarios**

- **Tipos de errores**
  - **Uno o mas bits erroneos dentro de un frame**
  - **Frames perdidos**
  - **Frames duplicados**

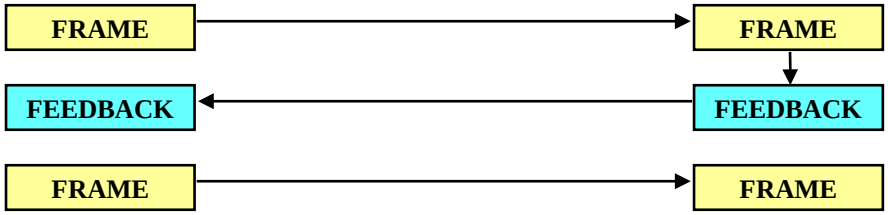
- **Aspectos**
  - **Detección**
  - **Recuperación**

Técnicas para control de errores

- **Codificación de redundancia**
  - **Para detección**
  - **Para corrección**



- **Retransmisión**



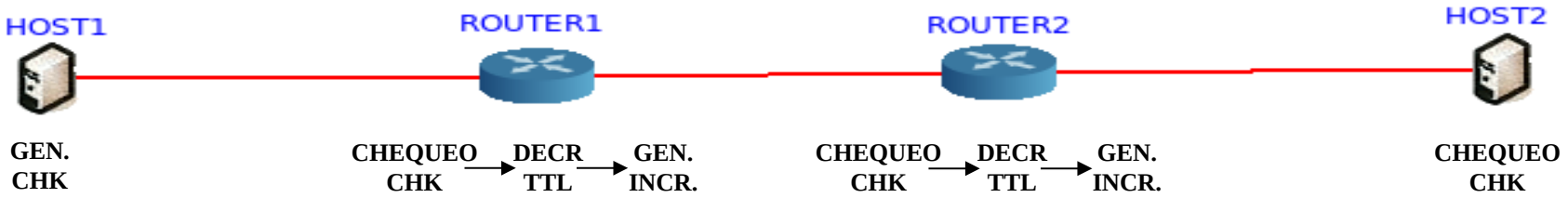
- **Tratamiento de errores**
  - **Casos extremos**
    - **Sólo redundancia: código autocorrector**
    - **Sólo retransmisión: (comparación) eco en una terminal**
  - **Uso normal: Redundancia para detección y retransmisión para corrección**

Control de errores por redundancia (2)

- **Codigos correctores de errores**
  - **Usos**
    - **Imposibilidad de retransmision**
    - **Retransmision muy costosa**
    - **Errores persistentes**
  - **Utilizados en 802.11, GSM, Wimax, Ethernet 10GB**

- **Codigos detectores de errores**
  - **Bit de paridad**
    - **Baja capacidad de detecciion**
    - **Utilizados en hardware por cada byte (transmision asincronica)**
  - **Codigos ciclicos**
    - **Implementados en hardware y software**
    - **Muy eficientes**
    - **Buena capacidad de deteccion de errores**
    - **Utilizados en Ethernet, 802.11, HDLC, etc**
  - **Checksum IP**
    - **Muy simple y rapido (aritmetica modulo 1)**
    - **Incremental**
    - **Indiferente al orden de los bytes en palabras**
    - **Puede realizarse en paralelo**
    - **Capacidad de deteccion limitada**
    - **Utilizado en las niveles IP y transporte de la arquitectura TCP/IP (IP, ICMP, UDP, TCP)**

Control de errores por redundancia (2)



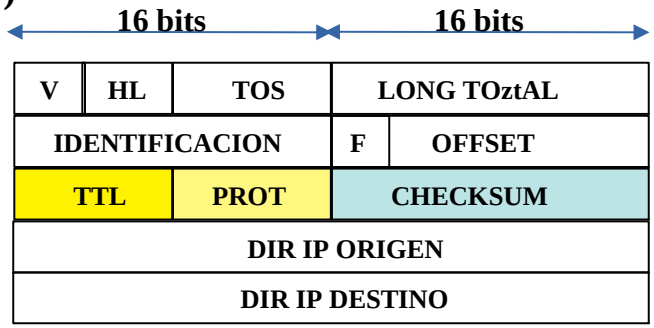
Checksum IP

- **Generacion de checksum**
  - **Agrupar los bytes en grupos de a 2 (16 bits). Si hay un numero impar rellenar con 0**
  - **Poner campo checksum en cero**
  - **Suma modulo 1 de los grupos de 16 bits**
  - **Complementar a 1 el resultado y colocarlo en el campo checksum**

- **Chequeo de checksum**
  - **Sumar en complemento a 1 la totalidad de grupos de 16 bits (incluido checksum)**
  - **Si el resultado es todos 1, el resultado es correcto**

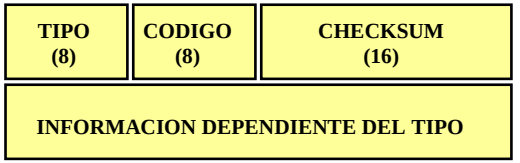
- **Calculo Incremental para cambio de un grupo de 16 bits (de valor_anterior a valor_nuevo)**
  - **nuevo_CHK = ~ (~ CHK_anterior + ~ valor_anterior + nuevo_valor)**

Ejemplo:
Llega datagram IP que contiene un ICMP (PROT = 1), con TTL = 9 y checksum = 0x1234
Checksum anterior: 0x1234
Valor anterior: 0x0901
Nuevo valor: 0x0801
Nuevo checksum: ~ (~ 0x1234 + ~ 0x0901 + 0x0801)

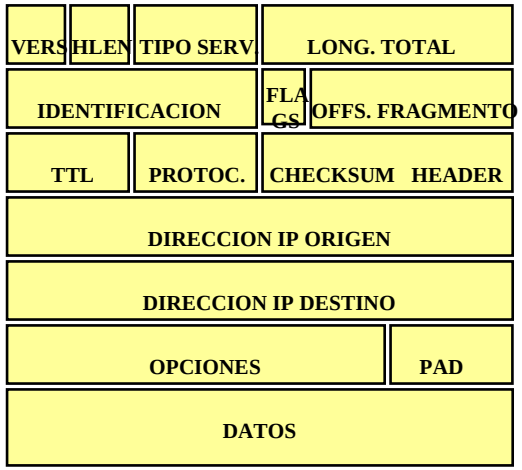


Uso de redundancia en algunos protocolos

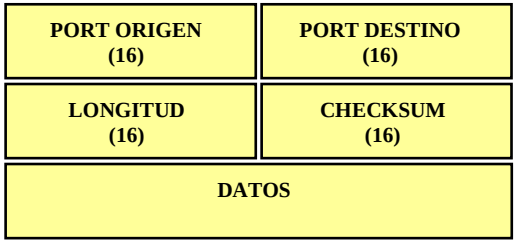
Ethernet



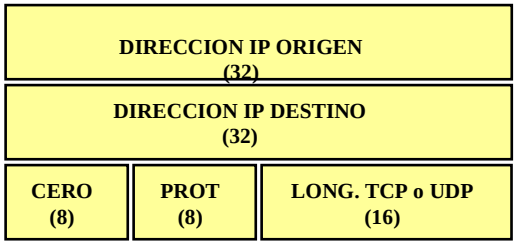
ICMP



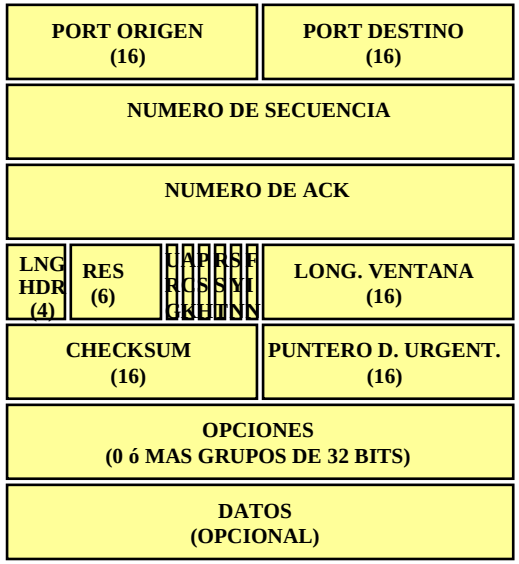
IP



UDP



Pseudoheader



TCP

Ethernet: codigo ciclico, cubre desde D. DEST hast FCS inclusive
IP: checksum IP, cubre solo el header IP
ICMP: checksum IP, cubre el frame ICMP completo
UDP: checksum IP sobre todo el segmento UDP mas el pseudoheader (uso opcional)
TCP: checksum IP sobre todo el segmento TCP mas el pseudoheader

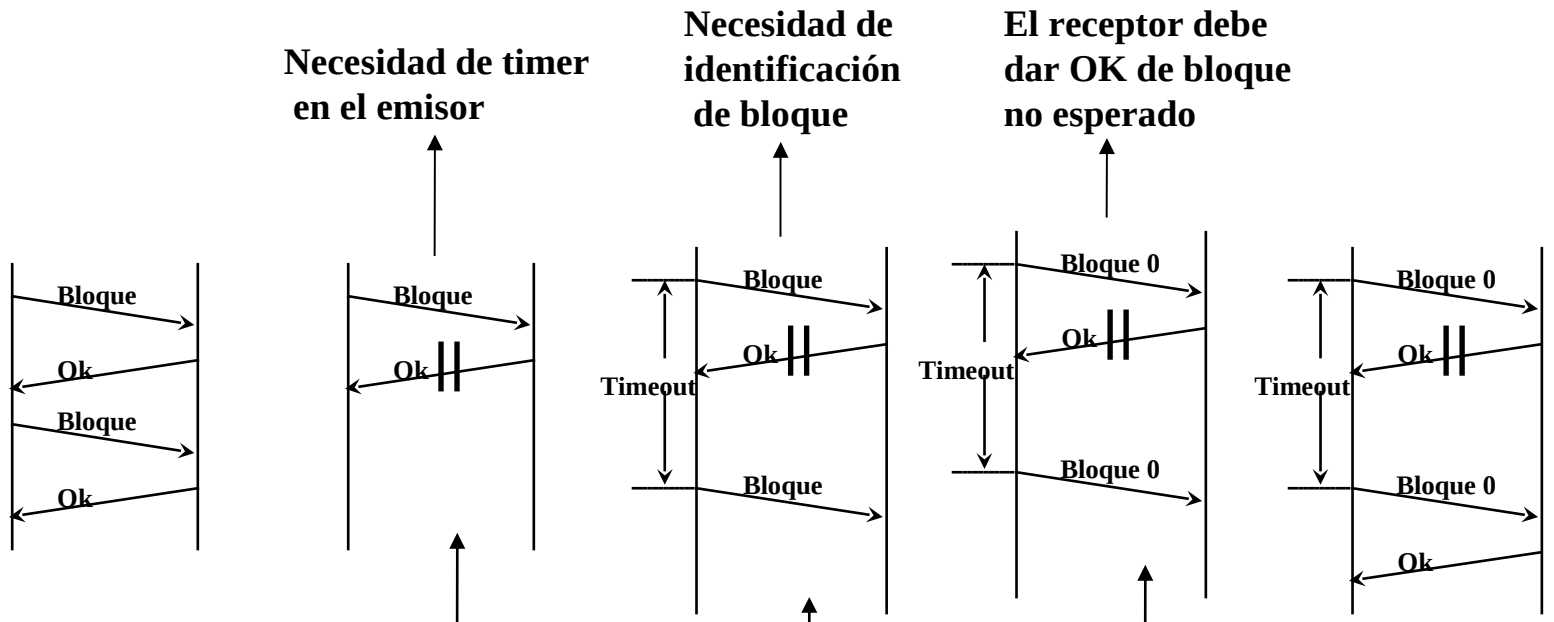
Control de errores-retransmisión (1)

Mecanismo basado en un acuerdo entre el emisor y el receptor, para que el primero reenvíe ciertos frames

- **Combinación de retransmisión y detección de errores por redundancia**
  - **Elementos necesarios**
    - **Generación y chequeo de redundancia**
    - **Indicación del receptor respecto a reenvío**
    - **Timers en el emisor**
    - **Identificación de bloques**
  - **Aspectos**
    - **Cuando reenviar**
      - **Receptor envía OK**
      - **Receptor envía OK y NAKs**
      - **Vencimiento de timer**
    - **Cuántos bloques reenviar**
      - **Depende del mecanismo de control de flujo utilizado**

Control de errores-retransmisión (2)

Situaciones de error



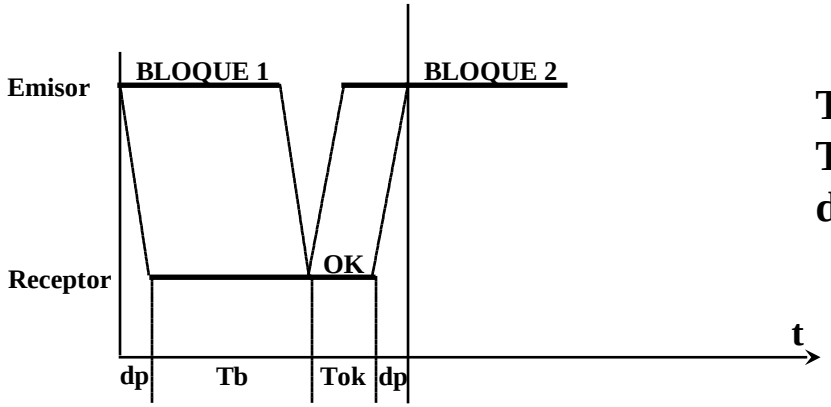
Control de flujo

- **Objetivo: El emisor no debe sobrepasar la capacidad del receptor**
- **Complejidad: Nivel 2: parámetros constantes (demoras, buffers, etc.)**
- **Evaluación**
  - **Recursos utilizados (proceso, buffers)**
  - **Aprovechamiento del canal (eficiencia)**

- **Mecanismos de control de flujo**
  - **Parada y espera**
  - **Ventanas**
    - **Fijas**
    - **Deslizantes**
      - **De longitud fija**
        - **Retransmisión continua (HDLC –REJ-)**
        - **Retransmisión selectiva (HDLC-SREJ-)**
      - **De longitud variable**
        - **Retransmision continua (TCP)**
        - **Retransmision selectiva (TCP con opcion SACK)**

Control de flujo. Parada y espera

- **Parada y espera**
  - **El emisor envía un bloque y espera respuesta**
  - **Utilizable en vínculos bidireccionales alternativos**
  - **Simplicidad en el protocolo y uso de un único buffer**
  - **Poco eficiente**
  - **e = Tb / (Tb + Tok + 2 * dp)**

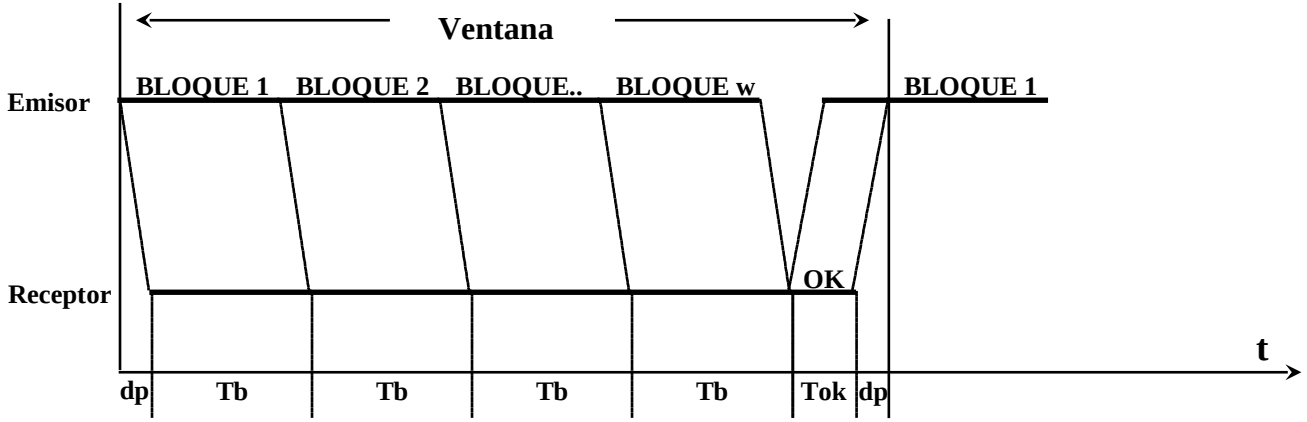


Tb :Tiempo de transmisión de un bloque
Tok:Tiempo de transmisión del asentimiento
dp : Demora de propagación

*** Significado del asentimiento: Se considera que sólo indica capacidad de procesar bloqu**

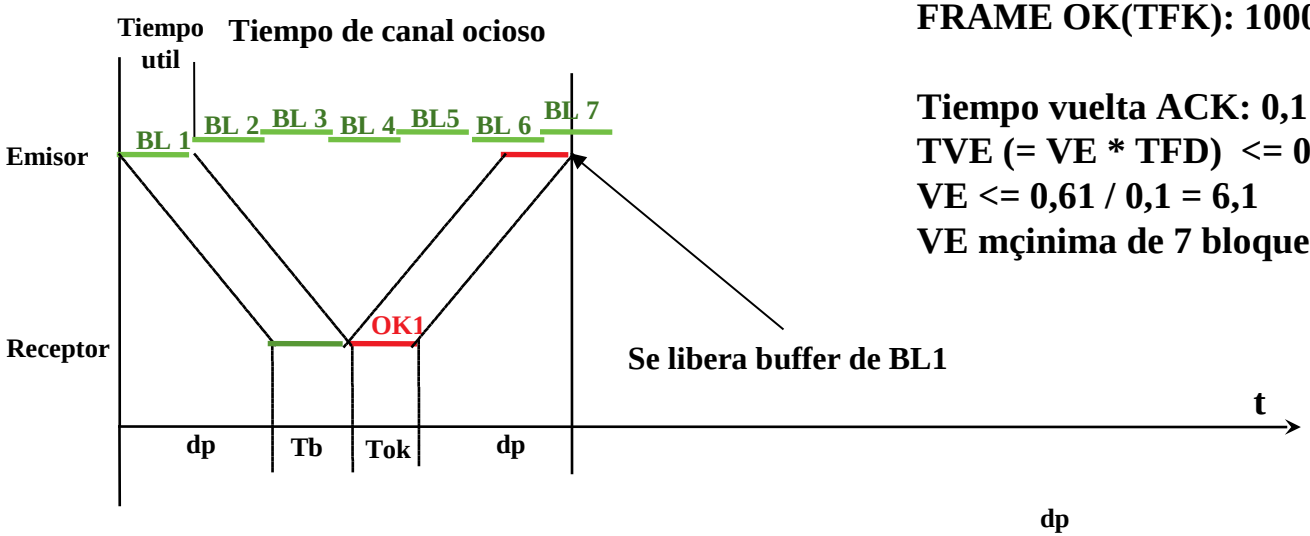
Control de flujo. Permisos al fin de la ventana

- **Ventana fija, permisos al fin de la ventana**
  - **Un permiso para todos los bloques de la ventana**
  - **Mayor capacidad de buffers (W: cantidad de bloques de la ventana)**
  - **Puede utilizarse en vínculos bidireccionales alternativos**
  - **Mejora la eficiencia de parada y espera**
  - **e = Tb * W / (Tb * W + Tok + 2 * dp)**



Control de flujo. Ventana deslizante (1)

- **Ventana deslizante**
  - **El mecanismo más utilizado**
  - **Capacidad de buffers : W (tamaño ventana) en emisor, uno o más en receptor**
  - **Permisos (acumulativos) enviados por cada bloque recibido**
  - **Debe disponerse de un vínculo bidireccional simultáneo**
  - **Se logra transmisión continua de bloques si 2*dp+Tok <= Tb*(W-1)**
  - **Más flexible respecto al tratamiento de errores que ventana fija con asentimiento al comienzo**



Ejemplo cálculo de VE
Demora propag: 0,25 seg.
Veloc. Transm: 100,000 bps
FRAME DATOS (TFD): 10,000 bits (0,1 seg)
FRAME OK(TFK): 1000 bits (0,01 seg)

Tiempo vuelta ACK: 0,1 + 0,01 + 2 * 0,25 = 0,61 seg
TVE (= VE * TFD) <= 0,61
VE <= 0,61 / 0,1 = 6,1
VE mçinima de 7 bloques

Control de flujo. Ventana deslizante (2)

- **Ventana deslizante**
  - **El mecanismo más utilizado**
  - **Capacidad de buffers : W (tamaño ventana) en emisor, uno o más en receptor**
  - **Permisos (acumulativos) enviados por cada bloque recibido**
  - **Debe disponerse de un vínculo bidireccional simultáneo**
  - **Se logra transmisión continua de bloques si 2*dp+Tok <= Tb*(W-1)**
  - **Más flexible respecto al tratamiento de errores que ventana fija con asentimiento al comienzo**