# Tema 2

# 6.263 / 16.37

# La capa de enlace de datos: entramado y detección de errores

Eytan Modiano
MIT, LIDS

# Capa de enlace de datos (DLC)

- **Responsable de la transmisión fiable de paquetes en un enlace:**
  - **Entramado: determina el inicio y el final de los paquetes (apt. 2.5)**
  - **Detección de errores: determina si un paquete contiene errores (apt. 2.3)**
  - **Recuperación de errores: retransmisión de paquetes con errores (apt. 2.4)**

    **Recuperación de la capa DLC**

    **Puede producirse en un nivel más alto**

# Entramado (*framing*)

**0101001110101001001010101001110001000**

**¿Dónde están los DATOS?**

- **Tres enfoques para delimitar las tramas y los *bits* de relleno vacíos:**

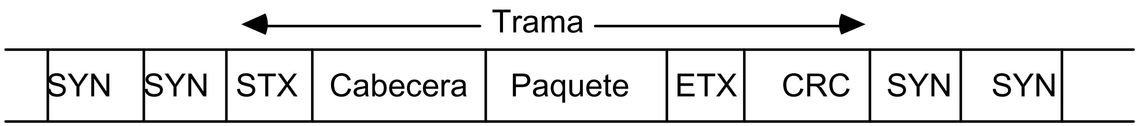
**1) Entramado orientado a caracteres**

**2) La longitud es importante:**

**- longitud fija**

**3) Protocolos orientados a *bits* (marcadores o *flags*)**

# Entramado basado en caracteres




SYN: inactividad (o vacío) síncrona
STX: inicio del texto
ETX: final del texto

- **Los códigos de caracteres estándar como ASCII y EBCDIC contienen caracteres especiales de comunicación que no pueden aparecer en los datos**
- **Toda la transmisión se basa en un código de caracteres**

# Problemas del entramado basado en caracteres

- **Depende del código de caracteres:**
  - **¿Cómo se envían datos binarios?**
- **Las tramas (*frames*) deben ser un número entero de caracteres**
- **Los errores en los caracteres de control pueden dar lugar a confusión**

**NOTA: método primario de entramado desde 1960 a ~1975**

# Enfoque del campo longitud (DECNET)

- **Utilice un campo cabecera para indicar la longitud de la trama (en *bits* o *bytes*):**
  - **El receptor puede contar hasta el final de la trama para encontrar el inicio de la trama siguiente**
  - **El receptor observa el campo longitud correspondiente en el encabezado del siguiente paquete para determinar su longitud**

- **El campo longitud debe ser $\log_2$ (Tam_Máx_Paquete) + 1 *bit*:**
  - **Con ello se restringe el tamaño del paquete que se va a utilizar**

- **Problemas con los recuentos de longitud:**
  - **Es difícil recuperarse de los errores**
  - **Es necesaria una resincronización tras detectar un error en el recuento**

# Paquetes de longitud fija (p. ej.: ATM)

- **Todos los paquetes tienen el mismo tamaño:**
  – **En las redes ATM todos los paquetes son de 53 *bytes***

- **Requiere sincronización al inicializar**

- **Problemas:**
  – **Las longitudes de los mensajes no son múltiplo del tamaño del paquete**
    **El último paquete de un mensaje debe contener un *bit* de relleno vacío (eficiencia)**

  – **Problemas de sincronización**

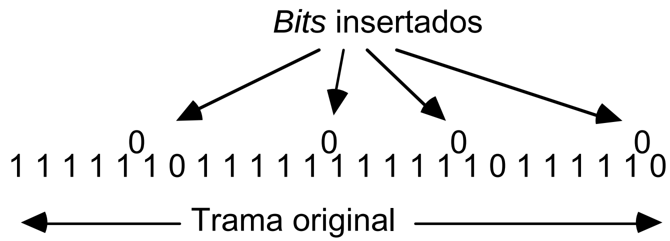
  – **La fragmentación y el reensamblaje se complican a velocidades altas**

# Entramado orientado a *bits* (marcadores o *flags*)

- Un marcador es una cadena fija de *bits* que indica el comienzo y el final de un paquete:
  - Se puede utilizar un único marcador para indicar el comienzo y el final de un paquete
- En principio, se podría utilizar cualquier cadena, pero es necesario evitar de algún modo la presencia del marcador en los datos:
  - Los protocolos estándar utilizan la cadena de 8 *bits* 01111110 como marcador
  - Utilice 01111111..1110 (<16 *bits*) como cancelación si hay errores
  - Los marcadores constantes o los 1s se consideran estados inactivos
- Así, 0111111 es la cadena de *bits* que no debe aparecer en los datos
- INVENTADO alrededor de 1970 por IBM para SDLC (protocolo síncrono de enlace de datos)

# INSERCIÓN DE *BITS* (emisor)

- **Se utiliza para eliminar el marcador de los datos originales**
- **Se inserta un 0 detrás de cada cinco 1s consecutivos en la trama original**




- **¿Por qué es necesario insertar un 0 en 0111110?**
  - **Si no se inserta, entonces:**

    **0111110111 -> 0111110111**
    **011111111 -> 0111110111**

  - **¿Cómo se diferenciaría en el receptor?**

# EXTRACCIÓN (receptor)

- **Si un 0 está precedido de 011111 en el flujo de *bits*, se elimina**
- **Si un 0 está precedido de 0111111, se trata del último *bit* del marcador**

**Ejemplo: los *bits* subrayados son los que se deben eliminar**

$$10011111\underline{0}1100111011111\underline{0}110\underbrace{\underline{01111110}}_{\text{marcador}}$$

# Rendimiento

- **Por lo general, con un marcador $01^K0$ es necesaria la inserción de *bits* siempre que $01^{k-1}$ aparezca en el flujo original de datos**
- **Para un paquete de longitud L, esto ocurrirá unas $L/2^k$ veces:**

$$E\{OH\} = L/2^k + (k+2) \text{ bits}$$

- **En el caso de un marcador de 8 *bits* OH ~ 8 + L/64:**
  - **Para la eficacia de paquetes grandes ~ 1 - 1/64 = 98,5 (o rendimiento 1,5%)**

- **Longitud óptima de marcadores:**
  - **Si los paquetes son largos requieren marcadores más largos (menor inserción)**
  - **Si los paquetes son cortos requieren marcadores más cortos (menor rendimiento debido al marcador):**

$$K_{opt} \sim \log_2(L)$$

# Errores de entramado

- **Todas las técnicas de entramado se ven afectadas por los errores:**
  - **Un error en un campo de recuento de longitud hace que la trama finalice en el sitio equivocado (y complica la búsqueda del comienzo de la siguiente trama)**
  - **Un error en DLE, STX o ETX causa los mismos problemas**
  - **Un error en un marcador o un marcador creado por error puede hacer que desaparezca una trama o que aparezca una adicional**
- **El enfoque orientado a marcadores es el menos sensible a los errores, ya que tarde o temprano aparecerá un marcador para indicar el final del siguiente paquete:**
  - **El único problema es que se crea un paquete erróneo**
  - **Este paquete erróneo se puede eliminar mediante una técnica de detección de errores**

# Técnicas de detección de errores

- Empleadas por el receptor para determinar si un paquete contiene errores
- Si se encuentran errores en un paquete, el receptor solicita al emisor que vuelva a enviarlo
- Técnicas de detección de errores:
  - Comprobación de paridad:
    - *Bit* único
    - Comprobación de redundancia horizontal y vertical
  - Comprobación de redundancia cíclica (CRC)

# Eficacia de la técnica de detección de errores

- **La eficacia de un código para la detección de errores se suele medir con respecto a tres parámetros:**

  **1) Distancia mínima del código (*d*) (núm. mín. de errores de *bit* no detectados):**

  **La distancia mínima de un código es el menor número de errores que una palabra de código puede asignar a otra. Si se producen menos de *d* errores, se detectarán siempre. Incluso si hay más de *d* errores, se suelen detectar (¡aunque no siempre!)**

  **2) Capacidad de detección de ráfagas (B) (la long. máx. de ráfaga siempre se detecta)**

  **3) Probabilidad de patrón aleatorio de *bits* mal asumido como libre de errores (buen cálculo si el núm. de errores en una trama >> *d* o B):**

  - **Útil cuando se pierde el entramado**
  - **K *bits* de información => 2k palabras de código válidas**
  - **Con r *bits* de comprobación, la probabilidad de que una cadena aleatoria de longitud k+r se asigne a una de las $2_k$ palabras de código válidas es $2^k/2^{k+r} = 2^{-r}$**

# Códigos de comprobación de paridad

| | |
|---|---|
| k *bits* datos | r *bits* de comprobación |

- **Cada comprobación de paridad es una suma módulo 2 de los bits de datos**

**Ejemplo:**

$$c_1 = x_1 + x_2 + x_3$$
$$c_2 = x_2 + x_3 + x_4$$
$$c_3 = x_1 + x_2 + x_4$$

# Código de comprobación de paridad simple

- Si la trama contiene un número impar de 1s, el *bit* de comprobación es igual a 1 si no es 0:

  1011011 -> 1011011 1
  1100110 -> 1100110 0

- Así, la trama codificada contiene un número par de 1s

- El receptor cuenta el número de unos que hay en la trama:
  – Un número par de 1s se interpreta como libre de errores
  – Un número impar de 1s indica que se debe haber producido un error:
    - Se puede detectar un error simple (o un número impar de errores)
    - No es posible detectar un número par de errores
    - No es posible corregir nada

- Probabilidad de que haya un error no detectado (errores independientes):

$$P(nodetectado) = \sum_{i \text{ par}} \binom{N}{i} p^i (1-p)^{N-i}$$

N = tam. paquete
p = prob. de error

# Paridad horizontal y vertical

$$
\begin{array}{ccccccc|c}
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\hline
1 & 0 & 1 & 1 & 1 & 1 & 1 & 0
\end{array}
$$

Comprobaciones horizontales

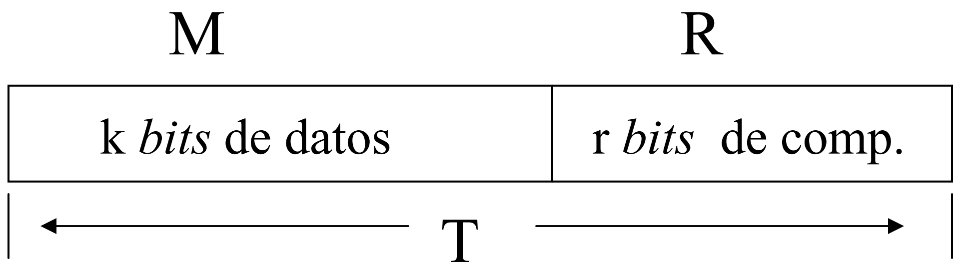
Comprobaciones verticales

$$
\begin{array}{ccccccc|c}
1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\hline
1 & 0 & 1 & 1 & 1 & 1 & 1 & 0
\end{array}
$$

- **Los datos se visualizan como una matriz rectangular (secuencia de palabras)**

- **Distancia mínima = 4, no es posible detectar 4 errores cualquiera en una configuración rectangular**

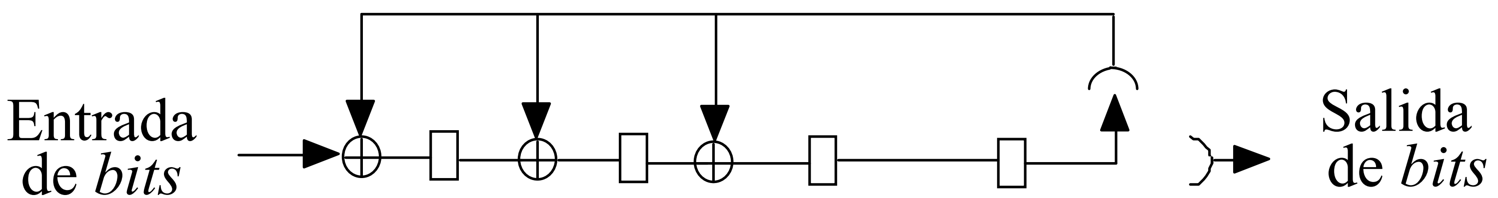
# Comprobaciones de redundancia cíclica (CRC)




M = *bits* de información
R = *bits* de comprobación
T = palabra de código

$$T = M\,2^r + R$$

- **Un CRC se implementa empleando un FSR (*feedback shift register*)**



# Comprobaciones de redundancia cíclica

$$T = M\,2^r + R$$

- ¿Cómo se calcula R (los *bits* de comprobación)?
  - Se selecciona una cadena generadora G de r+1 *bits* de longitud
  - Se selecciona R de modo que T sea un múltiplo de G (T = A*G, para algún A)
  - Entonces, cuando se divida T por G, el resto será 0 => sin errores
  - Todo se hace utilizando aritmética modular (módulo 2)

  $T = M\,2^r + R = A*G \Rightarrow M\,2^r = A*G + R$ (aritmética de módulo 2)

  Sea $R$ = resto de $M\,2^r/G$ y T múltiplo de G

- La selección de G resulta crucial para el rendimiento de un CRC

# Ejemplo

r = 3, G = 1001

M = 110101 => $M2^r$ = 110101000

```
             110011
        ______________
1001   | 110101000
         1001
         01000
          1001
          0001100
             1001
             01010
              1001
               011 = R (3 bits)
```

División de módulo 2

# Búsqueda de errores

- **Sea T' la secuencia recibida**
- **Dividir T' entre G**
  - **Si el resto = 0, dar por hecho que no hay errores**
  - **Si el resto es distinto de cero, se deben haber producido errores**

Ejemplo:
Se envía T = 110101011
Se recibe T' = 110101011
(no hay errores)

```
1001 | 110101011
       1001
       ----
       01000
        1001
        ----
        0001101
            1001
            ----
            01001
             1001
             ----
               000 => Sin errores
```

No hay forma de saber cuántos errores se han producido o qué *bits* contienen errores

# División de módulo 2 como división polinómica

# Implementación de un CRC

# Rendimiento del CRC

- **Para r *bits* de comprobación por trama y una longitud de trama menor que 2r-1, se detecta lo siguiente:**

  1) **Todos los patrones de 1, 2 ó 3 errores (d > 3)**
  2) **Todas las ráfagas de errores de r o menos *bits***
  3) **Número elevado de errores aleatorios con probabilidad $1\text{-}2^{-r}$**

- **Los DLC estándar utilizan un CRC con r = 16 y la opción de r = 32**

  – **CRC-16, $G = X^{16} + X^{15} + X^{2} + 1 = 11000000000000101$**

# Características de error de la capa física

- La mayoría de las capas físicas (canales de comunicaciones) no se describen correctamente con un solo parámetro BER
- La mayor parte de procesos de errores físicos suelen crear una mezcla de errores aleatorios y de ráfagas
- Un canal con un BER de $10^{-7}$ y un tamaño medio de ráfaga de 1.000 *bits* es muy distinto de uno con errores aleatorios independientes

- Ejemplo: para una longitud media de trama de $10^4$ *bits*:
  - Canal aleatorio: E[tasa de error de la trama] ~ $10^{-3}$
  - Canal de ráfagas: E[tasa de error de la trama] ~ $10^{-6}$

- Es mejor caracterizar un canal por su tasa de error de trama

- Esto representa un problema difícil en los sistemas reales