Capa de Enlace de Datos

Redes de Computadores

FIEC04705

Sesión 05

Agenda

- Terminología
- Conceptos básicos
- Protocolos orientados a bytes
- Protocolos orientados a bits
- Detección de errores

Terminología

Terminología

- **Frame:** unidad de información intercambiada por protocolos de bajo nivel.
- **Protocolo orientado a bytes**: Protocolo que trata a los frames como flujos de bytes.
- **Protocolo orientado a bits**: Protocolo que trata a los frames como flujos de bits.
- **Byte stuffing**: Inserción de un byte extra para evitar una mala interpretación de un byte de dato como un byte de control

Conceptos básicos

Conceptos básicos

- Un **protocolo** es un conjunto de reglas que necesitan ser implementadas en software y ser ejecutado por los dos nodos envueltos en un intercambio de datos a nivel de la capa de enlace de datos.
- La capa de enlace de datos transforma la capa física en un enlace responsable de la comunicación de nodo a nodo.

Conceptos básicos

- Responsabilidades de la capa de enlace de datos son:
  - Framing
  - Direccionamiento
  - Control de flujo
  - Control de errores
  - Control de acceso al medio
- Esta capa divide el flujo de bits recibidos de una capa de red en unidades de datos manejables llamados **frames**.

Conceptos básicos

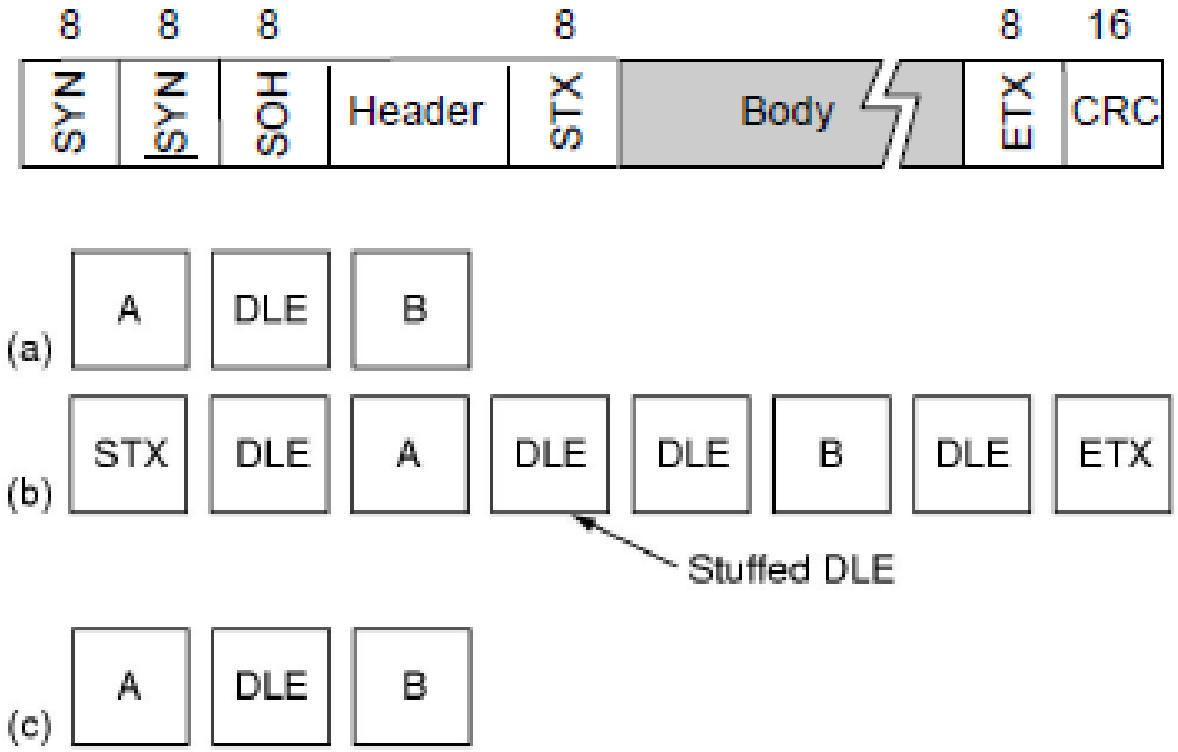
- La capa de enlace de datos agrega una cabecera al frame para definir la dirección del remitente y el destinatario del frame.
- Impone un mecanismo de control de flujo en caso que el destinatario tenga una capacidad de recepción de datos menor que la de producción de datos del remitente.
- Implementa mecanismos de detección y retransmisión de frames perdidos, duplicados o dañados a fin de agregar confiabilidad sobre la capa física.

Protocolos orientados a bytes

Protocolos orientados a bytes

- Ejemplos:
  - **IBM BISYNC**,
  - ARPANET IMP-IMP,
  - DEC DDCMP
- La principal desventaja de los protocolos orientados a bytes es que están estrechamente relacionados a los códigos de caracteres.

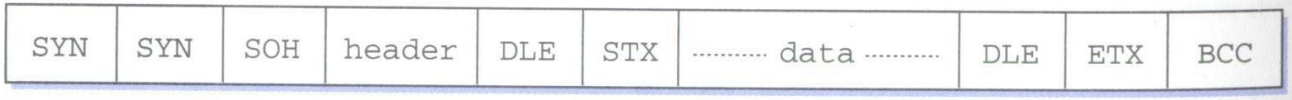
Protocolos orientados a bytes



Protocolos orientados a bytes

Figure 9.9 BSC Frame Formats

(a) Nontransparent data



(b) Transparent data

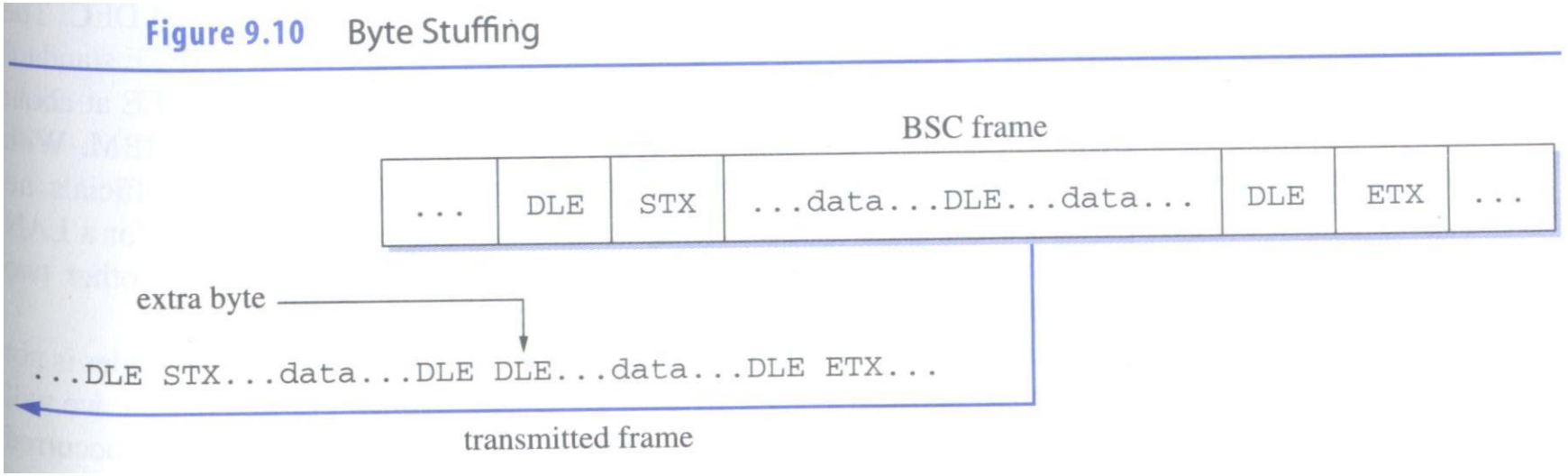
(c) Control frame

* It may be interesting to note that BSC can also be used with a lesser-known 6-bit code called Transcode. (Then again, maybe not.)

Protocolos orientados a bytes

- Binary synchronous communication protocol
- También conocido como BSC o bysinc protocol
  - Dos **SYN** characters: permiten al receptor del frame dividir el flujo de bits en bytes
  - Seguidos de bytes de control
  - **SOH**: start of header
  - **STX**: Start of Text
  - **DLE**: Data Link Escape – archivos binarios con patrones de bits aleatorios
  - **ETX**: End of Text
- Existen **control frames** y **data frames**

Protocolos orientados a bytes



Protocolos orientados a bits

Protocolos orientados a bits

- Ejemplos:
  - IBM SDLC,
  - ISO HDLC,
  - CCITT X.25 LAPB

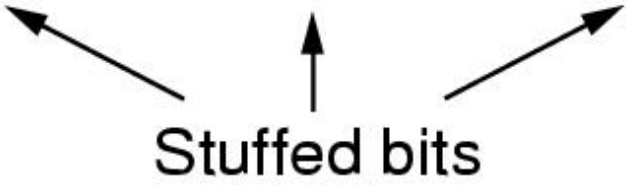
Protocolos orientados a bits

- Flujos de bits de longitud variable
- Remitente:
  - Encapsula el paquete (bit stream): 01111110
  - Agrega un 0 después de cada 11111 en el cuerpo (bit staffing)
- Receptor, cuando recibe 011111:
  - Siguiente bit 0: stuffed bit es removido
  - Siguiente bit 1:
    - Si el siguiente bit es 0 (01111110): marcar el fin del frame
    - Si el siguiente bit es 1 (01111111): error

Protocolos orientados a bits

(a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

(b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0



(c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Detección de errores

Chequeo de paridad

- Un bit adicional para hacer el número de 1s par (o impar).
- Detección de impar (par) bit de errores: número de errores pares (impares) no son detectados (50% de burst errors)

Chequeo de paridad

- **Burst error** significa que dos o más bits en la unidad de datos han sido corrompidos.
- Los bits corruptos no necesariamente son consecutivos.
- La ocurrencia de un burst error es más probable que el error de un simple bit: La duración del ruido es mayor que la duración de un bit
- Si un impar número de bits cambia, el chequeo de paridad funciona
- Si un número par de bits cambia, el chequeo de paridad no detectará el error
- Por lo tanto, el chequeo de paridad encontrará alrededor del 50% de los burst errors, lo cual no es bueno para comunicaciones.

Chequeo de paridad

- **Burst error detection**
  - Idea básica: Distribuir los bits de un mensaje en diferentes frames
  - Garantiza la detección de errores para aquellos burst en los que su duración es menor que el tiempo de envío de un frame
  - Detecta burst errors con una probabilidad $1-(1/2)^n$

Chequeo de paridad

FIGURE 4.1 Detecting Single Bit Errors Using Parity Checking

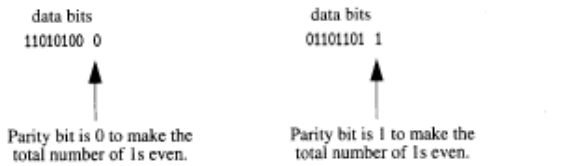


FIGURE 4.2 Detecting Consecutive Double-Bit Errors Using Parity Checking

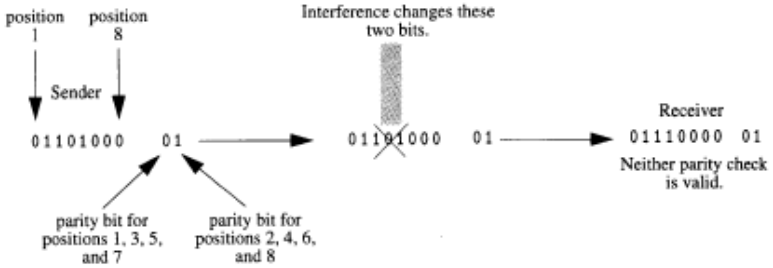


FIGURE 4.3 Detecting Burst Errors Using Parity Bits

Sender				Receiver		
Row (frame) number		Parity bit for one row		Row number		Parity bit for one row
1	01101	1		1	01101	1
2	10001	0		2	10001	0
3	01110	1		3	01100	1*
4	11001	1		4	11001	1
5	01010	0	Burst error occurs and destroys column four, making it all zeroes.	5	01000	0*
6	10111	0		6	10101	0*
7	01100	0		7	01100	0
8	00111	1		8	00101	1*
9	10011	1		9	10001	1*
10	11000	0		10	11000	0
Column number	12345	6		Column number	12345	6

* Parity bit is not correct

Cyclic Redundancy Checks: CRC

- Es una poderosa técnica de **detección de errores** basada en divisiones polinomiales.
- $b_n\ b_{n-1}\ b_{n-2} \ldots b_1\ b_0 \rightarrow b_n\ x^n + b_{n-1}\ x^{n-1} + b_{n-2} x^{n-2} + \ldots b_1\ x^1 + b_0$
- Pasos:
  - Dada una cadena de bits B, añadir un número [igual al grado de G(x)] de ceros al final: B(x)
  - Dividir B(x) por un generador polinomial G(x) acordado y obtener el resto R(x)
  - Definir T(x) = B(x) – R(x) [Nota: T(x)/G(x) genera un resto de 0]
  - Transmitir T -> T(x) el cual arribará como T'(x)
  - Si T'(x)/G(x) genera un resto 0 entonces OK. Caso contrario error!

Principio del CRC

CRC principle

- append certain number of bits to the message
- send
- receive
- use appended bits to detect possible errors

$G(x)$ … generator polynomial (known to sender and receiver – agreed beforehand)
$M(x)$… message to be delivered from sender to receiver
$M'(x)$ … data leaving sender
$N'(x)$ … data arriving into receiver

Cyclic Redundancy Checks: CRC

How to get $M'(x)$ from $M(x)$?

- append string of 0s of the length equal to the degree of $G(x)$ to $M(x)$. Resulting string is $M''(x)$.
- do modulo 2 of $\frac{M''(x)}{G(x)}$
- if the result is 0 then $M'(x) = M''(x)$ otherwise change $M''(x)$ by altering appended bits in order to get 0 as a result and this is your $M'(x)$.

How to get $M(x)$ from $N'(x)$ on the receiver side?

- take received $N'(x)$ and do modulo 2 of $\frac{N'(x)}{G(x)}$
- if the result is **not 0** then discard $N'(x)$ and request retransmission
- if the results is 0 then remove number of the least significant bits equal to the degree of $G(x)$ and what is left is your $M(x)$.

Puntos para recordar

- Protocolos orientados a bits vs. a bytes
- Mecanismos de detección de errores

Próxima Sesión

- Corrección de errores
- Control de flujo
- El problema de la asignación del canal