### **Data Link Layer(7)**

#### **Example:**



parity bit 1 wrong (1,3,5,7,9,11,13,15,17,19,21 contain altogether 5 Ones)parity bit 2 correct (2,3,6,7,10,11,14,15,18,19, contain altogether 6 Ones)parity bit 4 wrong (4,5,6,7,12,13,14,15,20,21 contain altogether 5 Ones)parity bit 8 correct (8,9,10,11,12,13,14,15 contain altogether 2 Ones)parity bit 16 correct (16,17,18,19,20,21 contain altogether 4 Ones)---> Bit No. 5 is wrong and has to be inverted!

#### Use of ECCs :

Heavily disturbed transmission channels (wireless) or those with long propagation times (space missions)Data transmission in military (secret service) applications

•Secondary storage media (e.g. CD's), to correct reading errors instantaneously (without timing delay)

Vorlesung "Kommunikation und Netze", SS'09 E. Nett

## **Data Link Layer(8)**

#### Use of the Hamming - Code to correct burst errors:

Char.	ASCII	Check bits
		$\wedge$
н	1001000	00110010000
a	1100001	10111001001
m	1101101	11101010101
m	1101101	11101010101
i	1101001	01101011001
n	1101110	01101010110
g	1100111	11111001111
•	0100000	10011000000
с	1100011	11111000011
0	1101111	00101011111
d	1100100	11111001100
е	1100101	00111000101
	Order of bit transmission	

#### (eindimensionaler) Paritätscode:



Vorlesung "Kommunikation und Netze",

SS'09

E. Nett

## **Example for Checksumming: Cyclic Redundancy Check (CRC)**

- view data bits, **D**, as a binary number
- choose r+1 bit pattern (generator), **G**
- goal: choose r CRC bits, **R**, such that
  - <D,R> exactly divisible by G (modulo 2)
  - receiver knows G, divides <D,R> by G. If non-zero remainder: error detected!

$$\begin{array}{c} \bullet & \bullet & \bullet & \bullet & \bullet \\ \hline D: \text{ data bits to be sent } R: CRC \text{ bits } & bit \\ pattern \\ D*2^{r} XOR R & mathematical \\ formula \end{array}$$

# **CRC** Example



Vorlesung "Kommunikation und Netze",

## Data Link Layer(12)

**Examples for (generator) polynomials that have become international standards:** 

CRC-12	$= x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$	(110000001111)
CRC-16	$=x^{16}+x^{15}+x^2+1$	(1100000000000101)
CRC-CCITT	$=x^{16}+x^{12}+x^{5}+1$	(1000100000100001)
CRC-32		(100000100110000010001110110110111)

### **Properties:**

They detect

- all single and double errors
- all errors with an odd number of bits
- all burst errors of length *r* or less
- all burst errors of length greater than r with probability 1  $0.5^r$ 
  - $\rightarrow$  the higher r, the better the protection
- e.g. for CRC-16:
  - 99.997% of 17-bit error bursts
  - 99.998% of 18-bit and longer bursts
- widely used in practice (802.11 WiFi, ATM)

Vorlesung "Kommunikation und Netze",

## Data Link Layer(13)

### **Providing Flow Control**

### Simple Stop-and-Wait Protocol (rdt 2.0)

Assumptions:

No infinite amount of buffer space available on the receiver's side (otherwise: always send (rdt 1.0))

Communication channel is assumed to be error-free

Problem:

How to prevent the sender from flooding the receiver with data faster than the latter is able to process which results in losing frames at the receiver's side

Solution approach:

Having the receiver provide feedback to the sender by acknowledging each frame sent

---> the sender must wait until an *ack* frame arrives before fetching the next frame from the network layer

**Piggybacking** (if data traffic is duplex (bidirectional)) :

Delaying outgoing *acks* so that they can be hooked on the next outgoing data frame (having a separate *ack* field in its header)

Principal advantage over having separate ack frames:

Better use of the available channel bandwidth

## **Data Link Layer(14)**

### **Sliding window protocols**

Assumptions:

Communication channel is erroneous, i.e. data frames and acks could be lost or garbled

---> time-out mechanism for the sender

Problem:

---> How long is timeout? ---> Early time-out ---> synchronization problem!

Solution approach:

Having sliding window protocols were each data frame contains a sequence number

---> for a stop-and-wait protocol it needs only 2 numbers (0 and 1), i.e. a one-bit sequence number

### Illustration of the main idea of sliding windows



## **Data Link Layer(15)**

## **One Bit Sliding Window Stop-and-Wait Protocol (rdt 3.0)**

- Ack field contains the sequence number of the last frame received without error
- •Agreement with the number of the frame being sent ---> fetch the next frame from the network layer
- ■No agreement ---> no sending action

## **Resulting Quality of Service**

- No delivering of duplicate packets to either network layer
- No packet is skipped
- No deadlock

# rdt3.0 in action (1)



(a) operation with no loss



# rdt3.0 in action (2)

