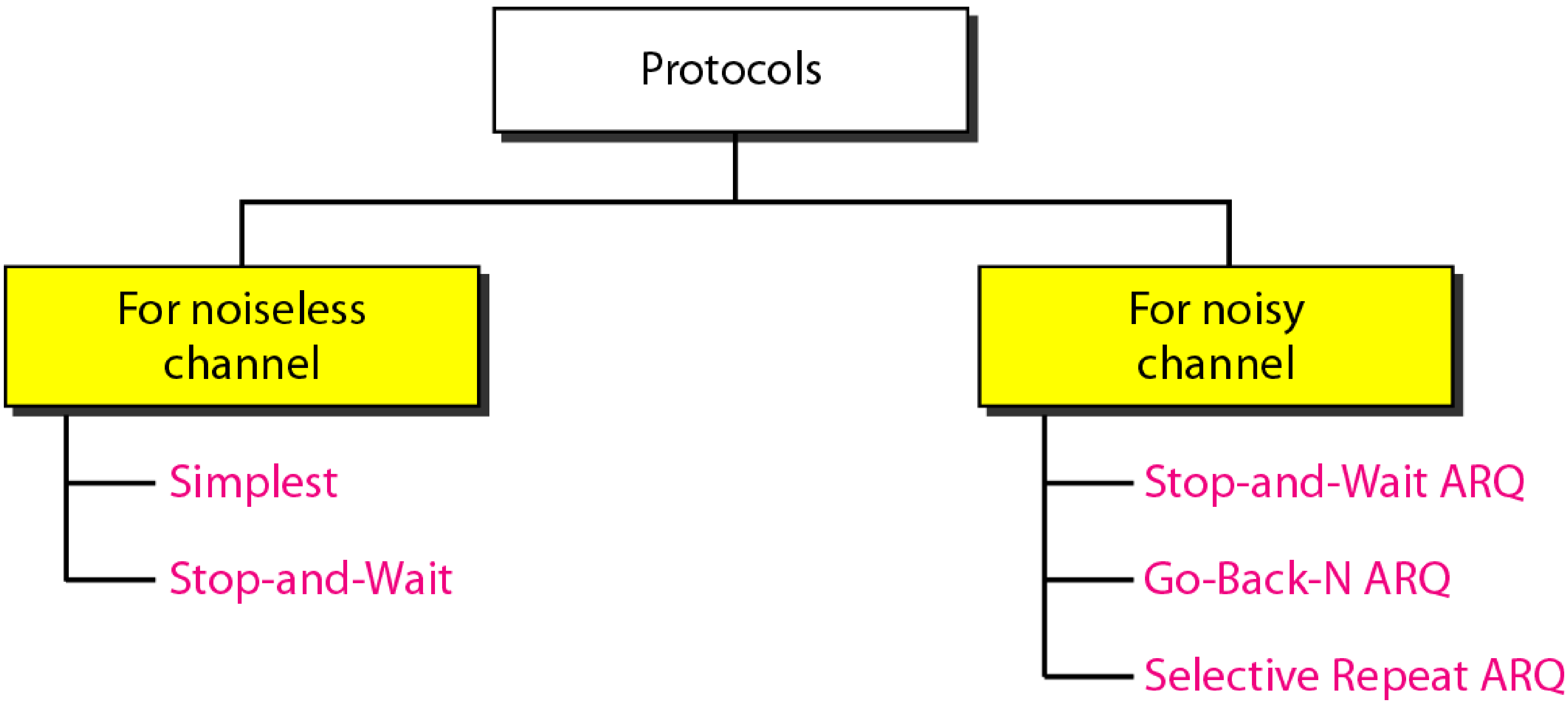
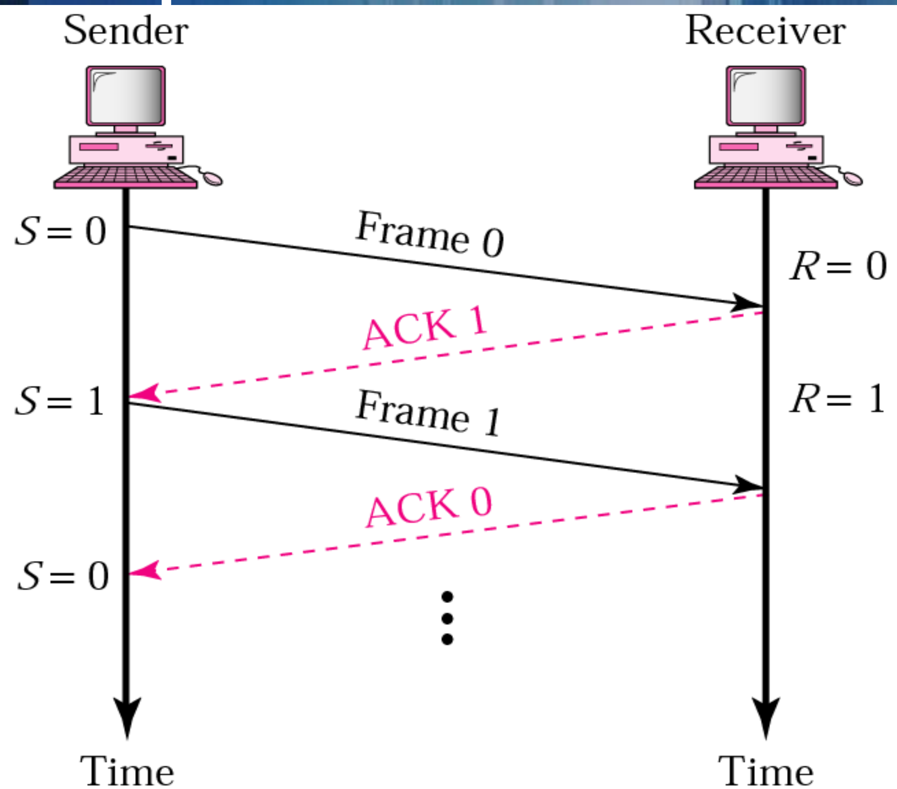
Review: Flow Control

Flow Control: Refers to the control of the amount of data that a sender can transmit **without overflowing the receiver**.



* Figure is courtesy of B. Forouzan

Stop-and-Wait ARQ

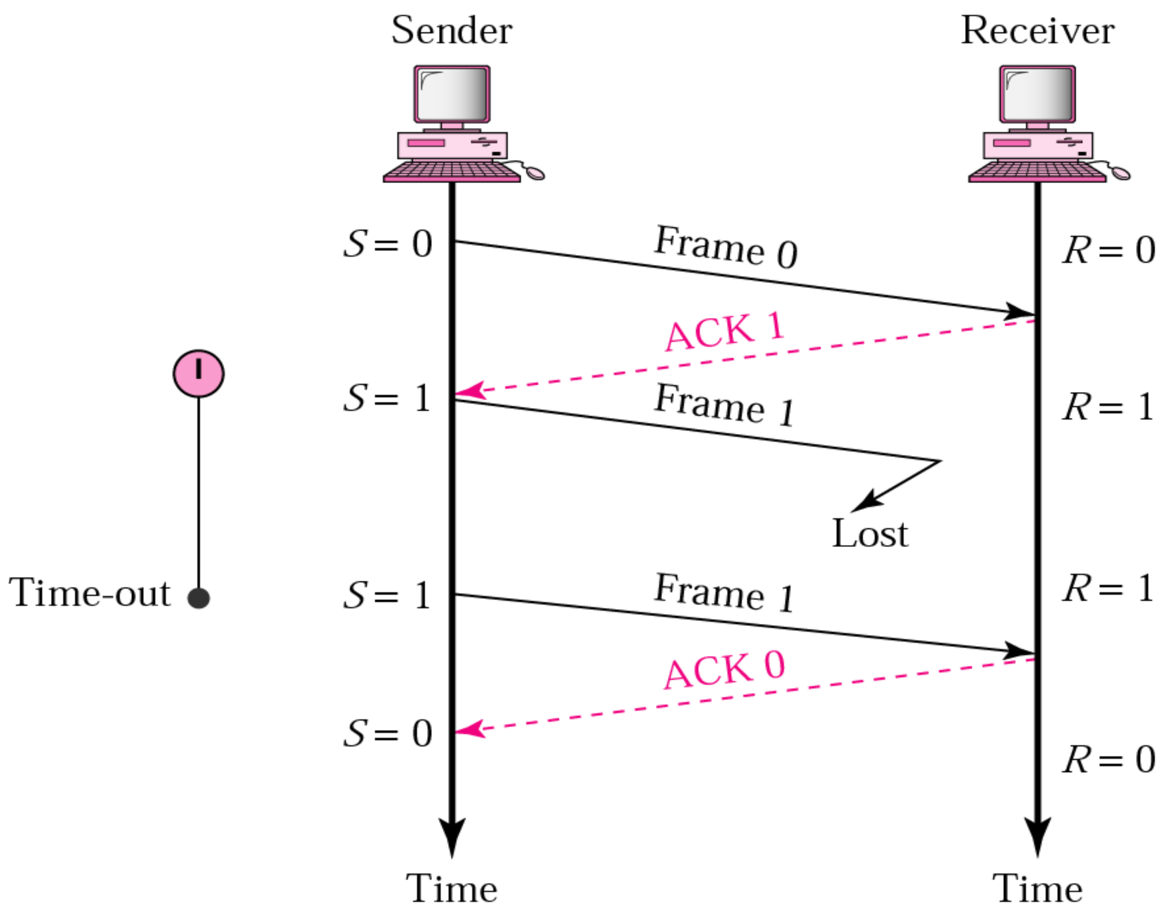


- ACK = received packet, ready to receive packet #

* Figure is courtesy of B. Forouzan

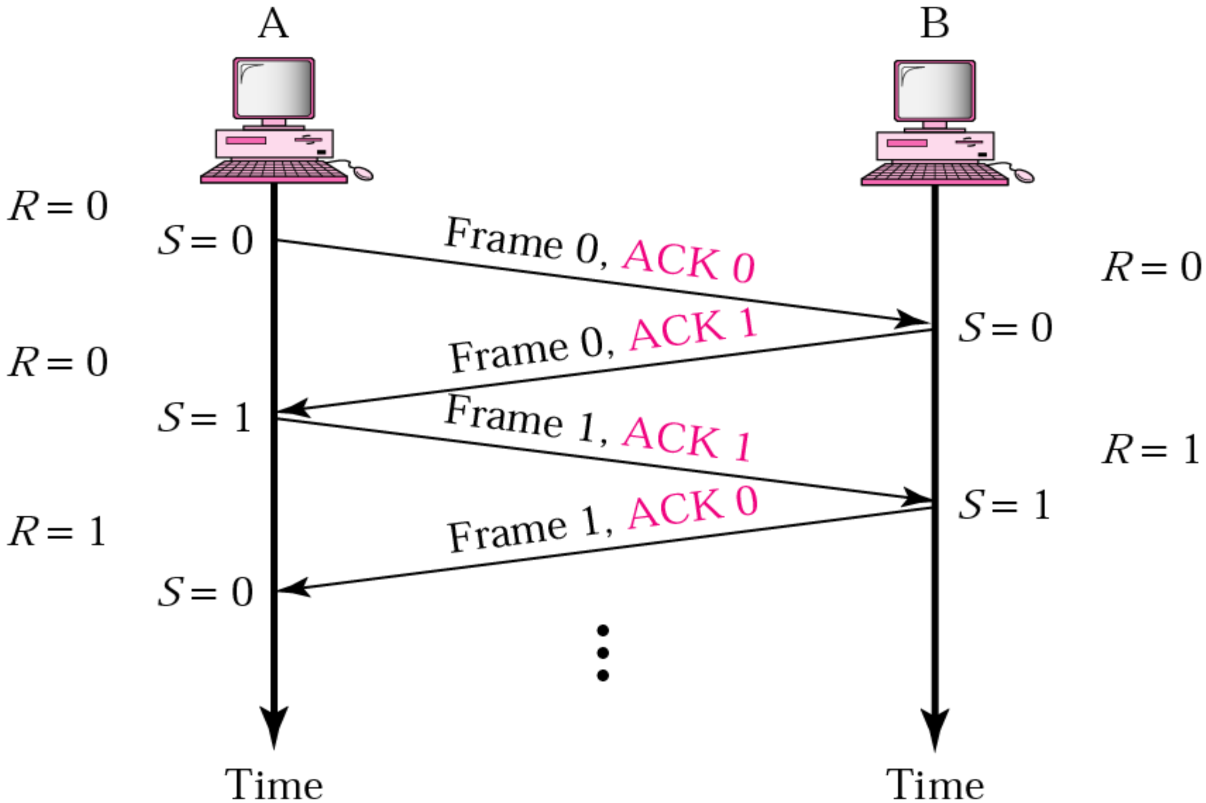
Stop-and-Wait ARQ: Time-Out

- Frame is lost during transmission



* Figure is courtesy of B. Forouzan

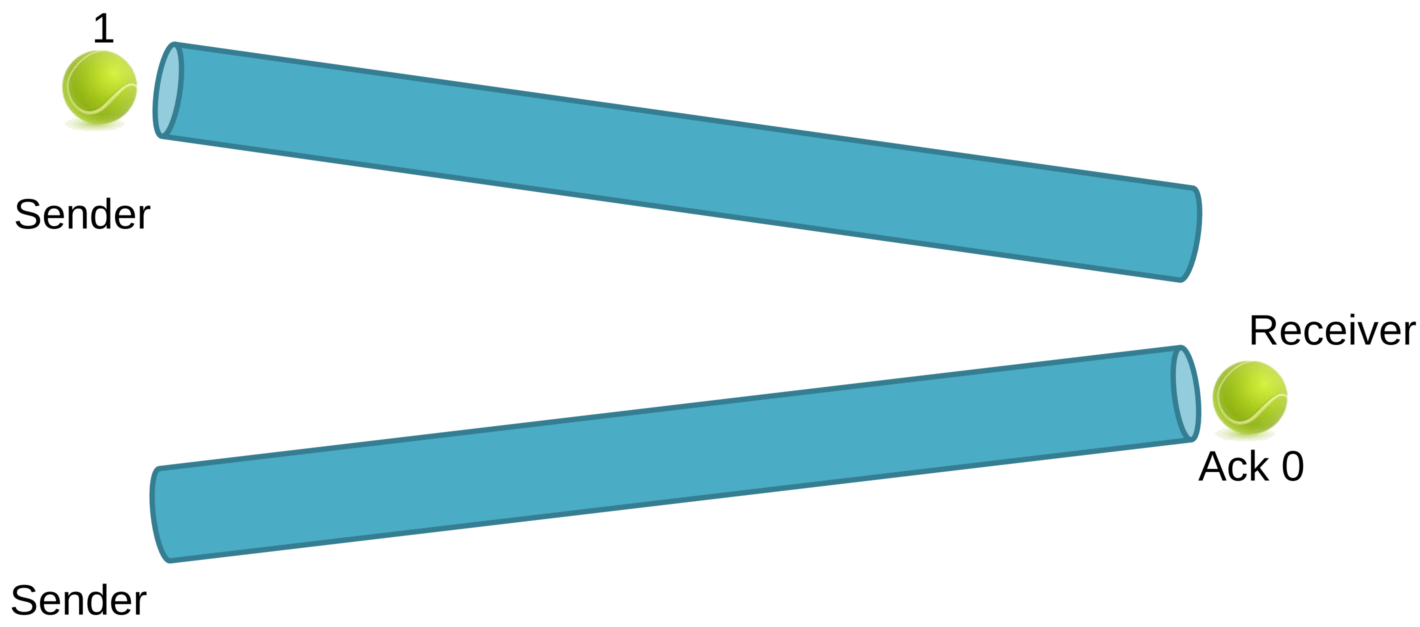
Piggybacking ACKs



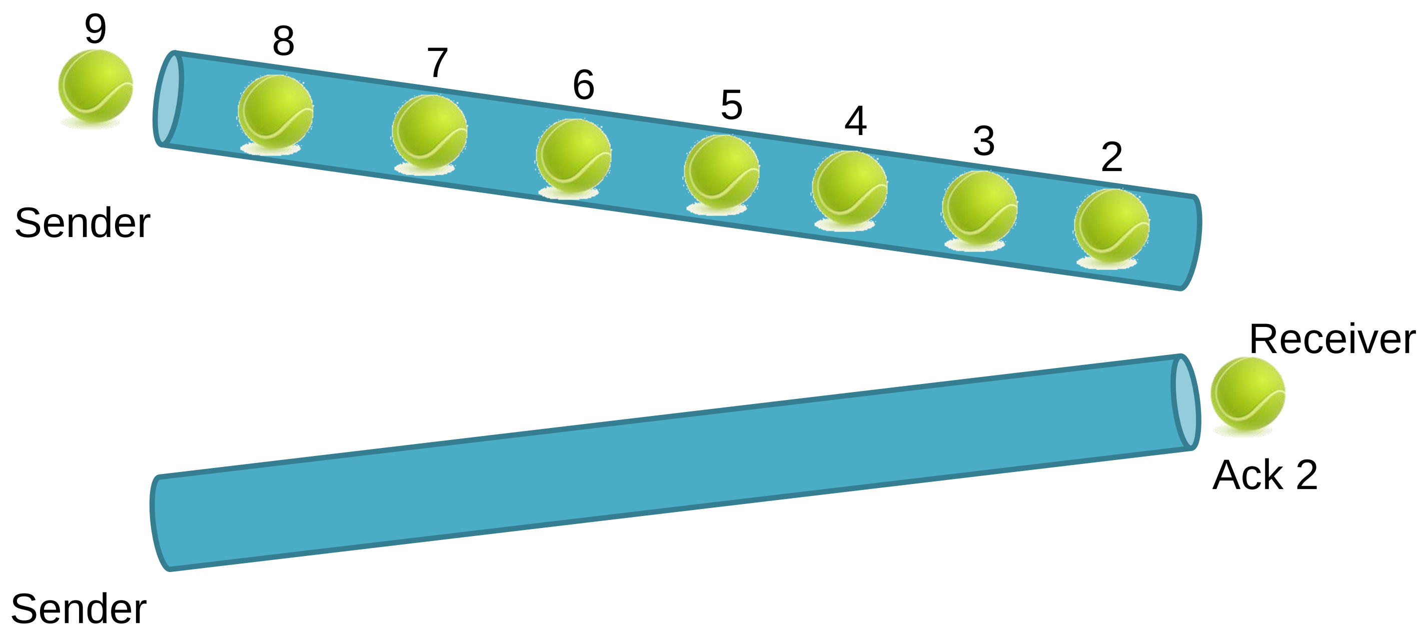
- Next data frame send carries acknowledgement for last frame received

* Figure is courtesy of B. Forouzan

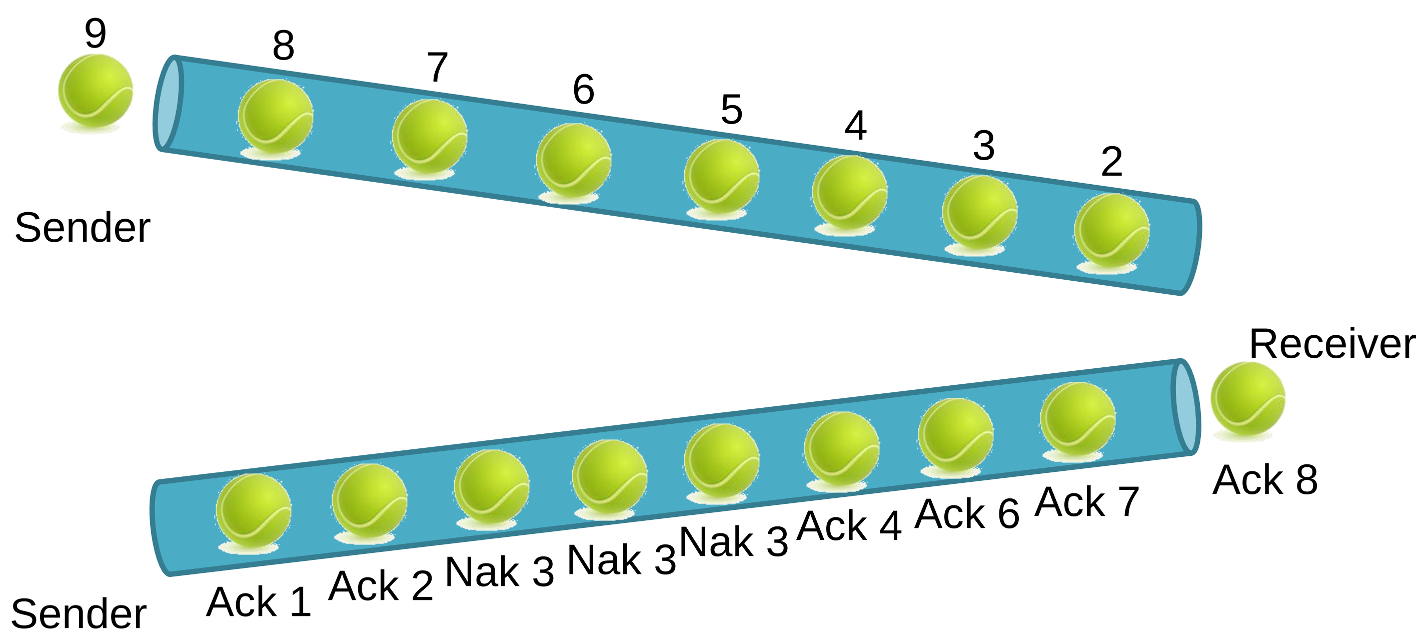
Bandwidth-Delay Product: Example



Bandwidth-Delay Product: Example

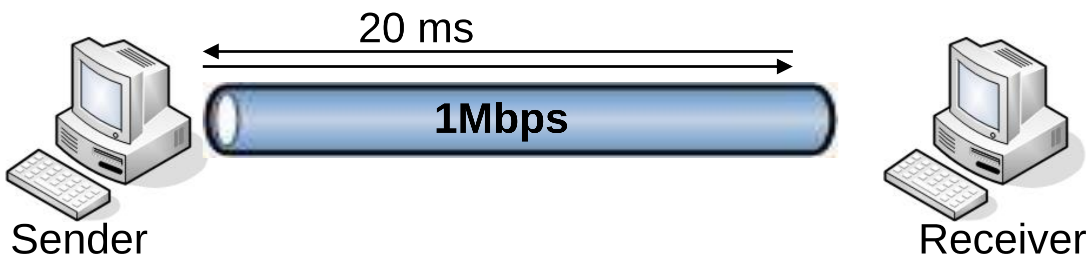


Bandwidth-Delay Product: Example



Bandwidth-Delay Product: Example

- Bandwidth × Round-Trip-Time
  - gives indication of amount of data that can be send while waiting for ACK

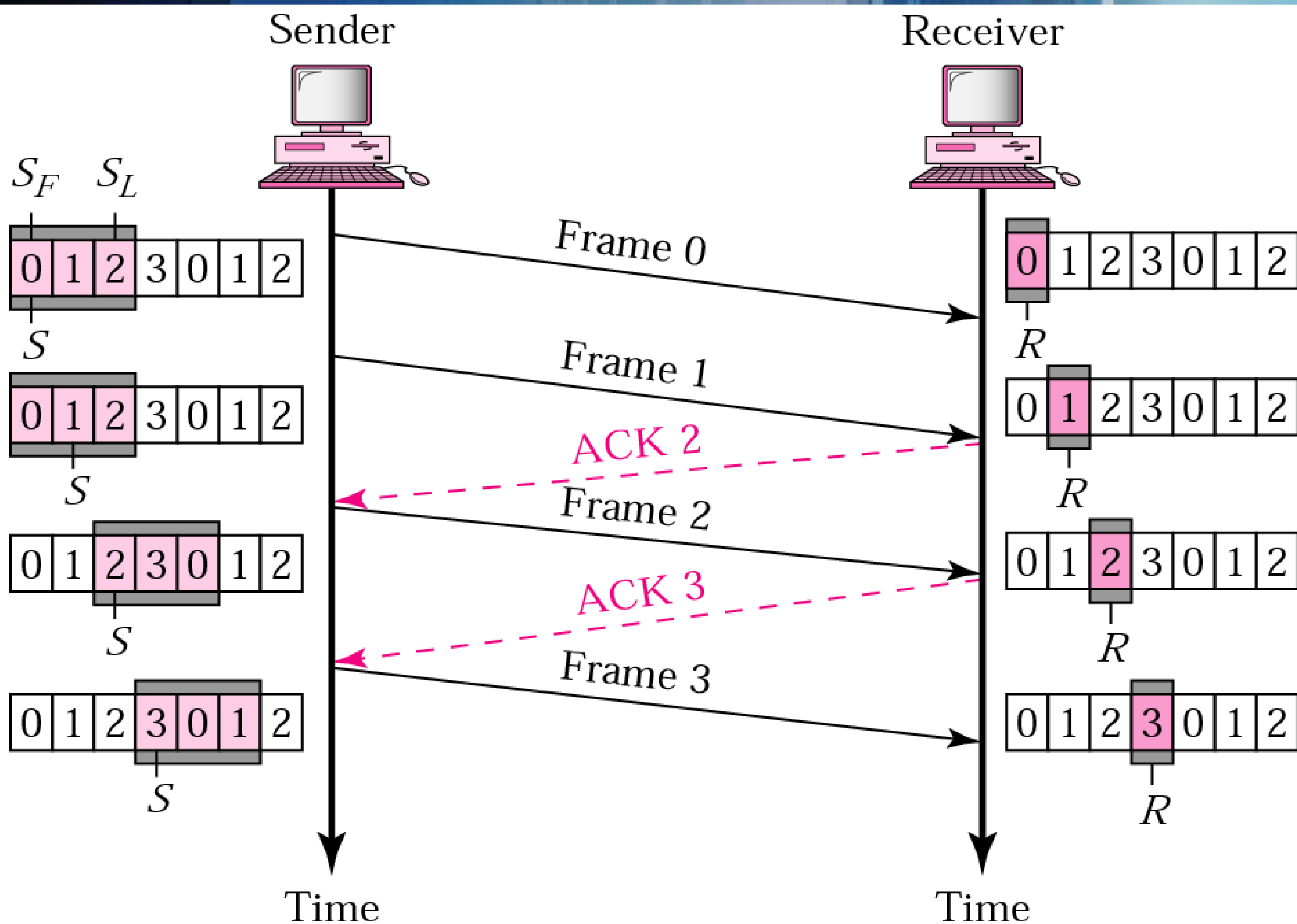


- Communication link with 1Mb/s
- Round-Trip time: 20 ms = $20*10^{-3}$ s

- How much data can you send during the time it takes for 1 bit to
- $20*10^{-3}$ s * $1*10^{6}$ b/s = 20.000 bits

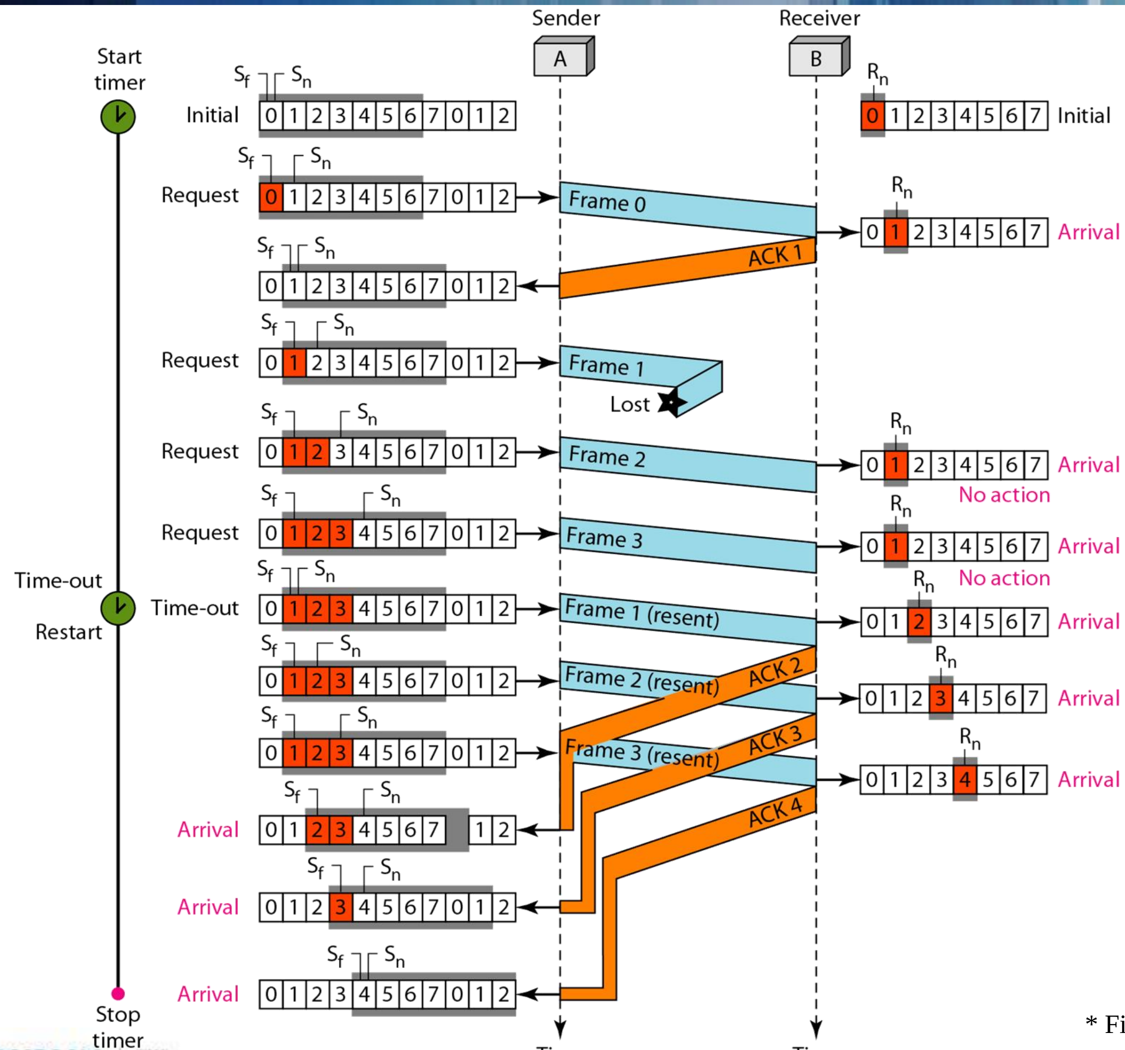
- Frame of 2000 bit ⇒ 10% of bandwidth used

Go-Back-N ARQ



* Figure is courtesy of B. Forouzan

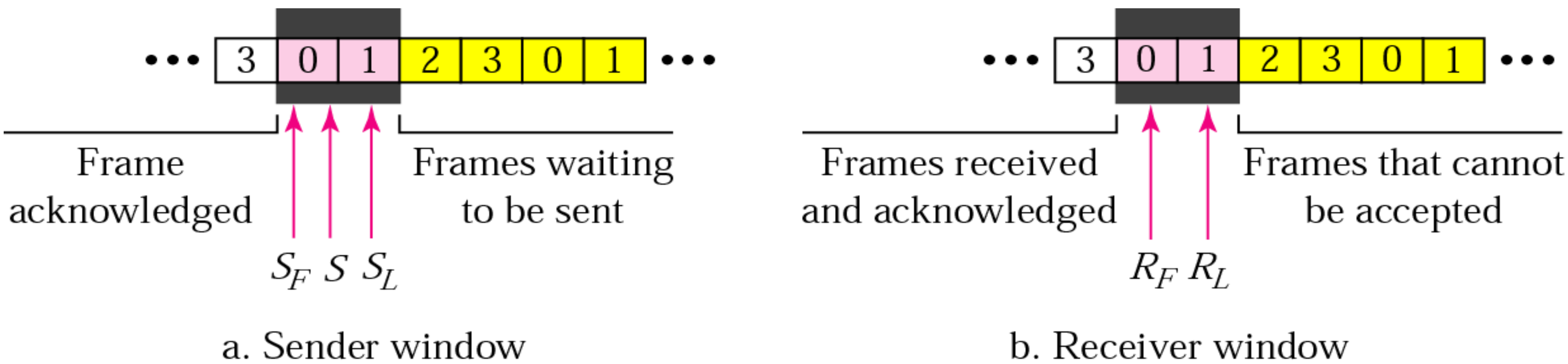
Go-Back-N ARQ: Bad Behaviour



* Figure is courtesy of B. Forouzan

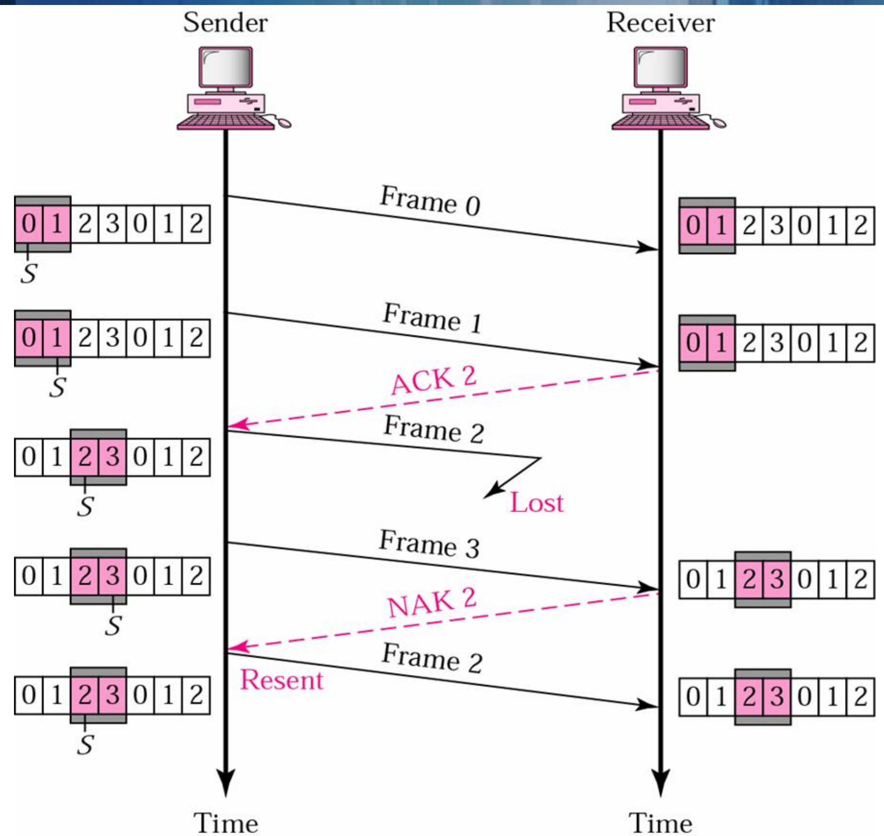
Selective Repeat

- Two Windows:
  - 1 Sender Window – 1 Receiver Window



* Figure is courtesy of B. Forouzan

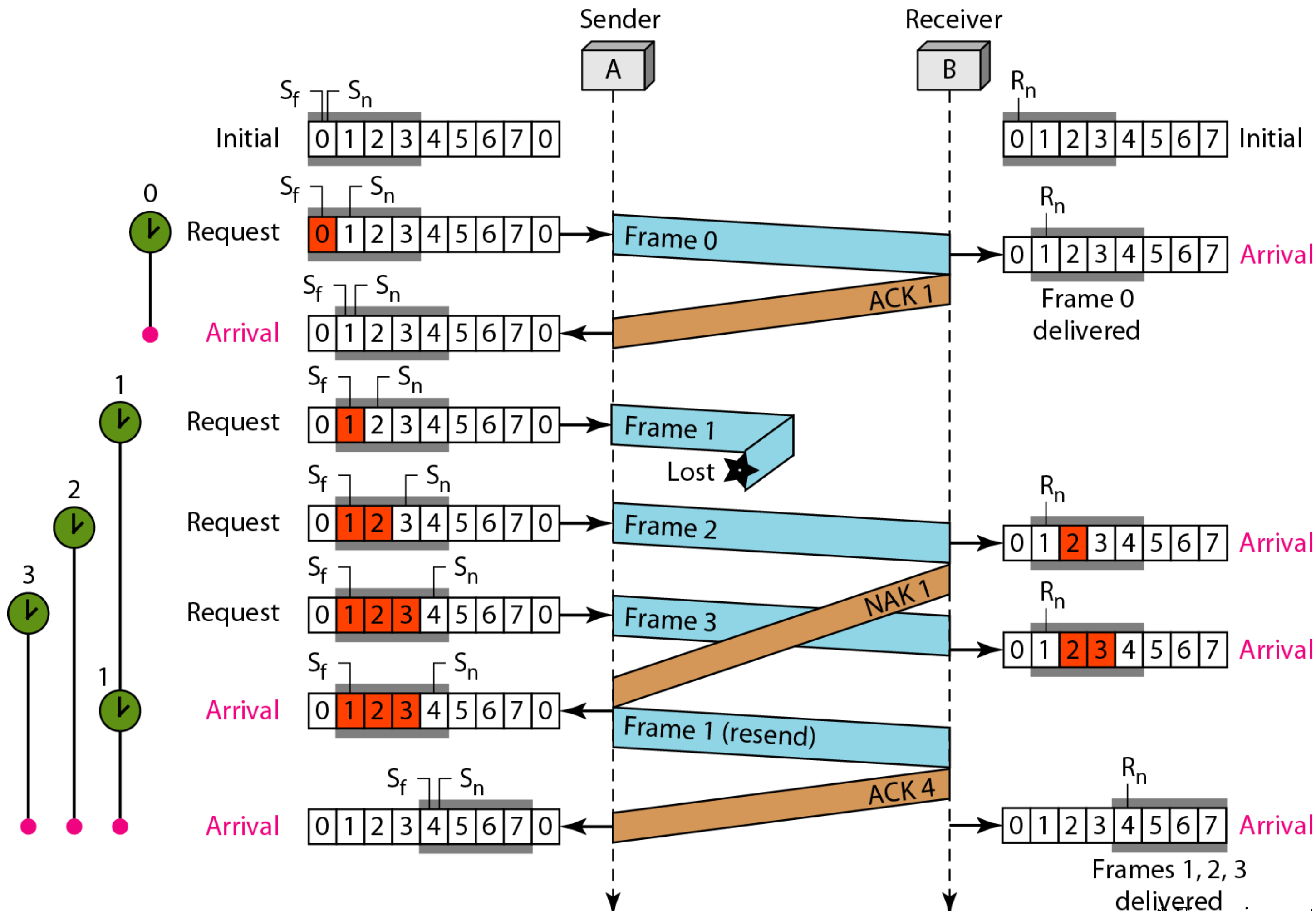
Selective Repeat ARQ: Lost Frame



- NAK = Negative Acknowledgement
- Sender still maintains timers for packets in case NAK gets lost

* Figure is courtesy of B. Forouzan

Selective Repeat ARQ



* Figure is courtesy of B. Forouzan

Sliding Windows

- Allow multiple frames to be in transit

- Receiver has buffer w long
- Transmitter can send up to w frames
  - without ACK

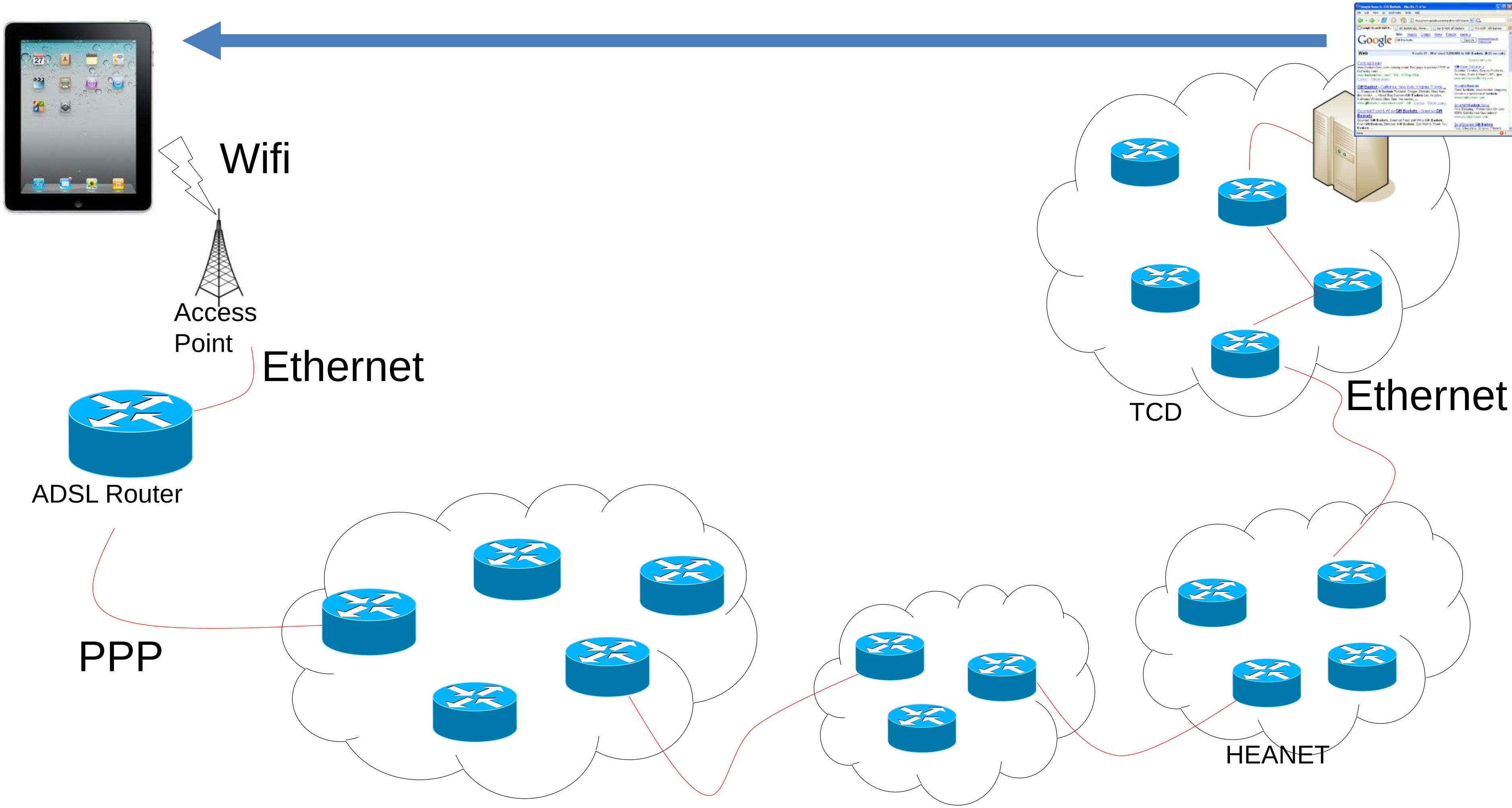
- Each frame is numbered
- ACK includes number of next frame expected

Window Size for Go-Back-N

- Depends on size of max. frame number
  - Frame # needs to be included in every frame
  - e.g. 4 bits – 2^4 = 16 frame numbers

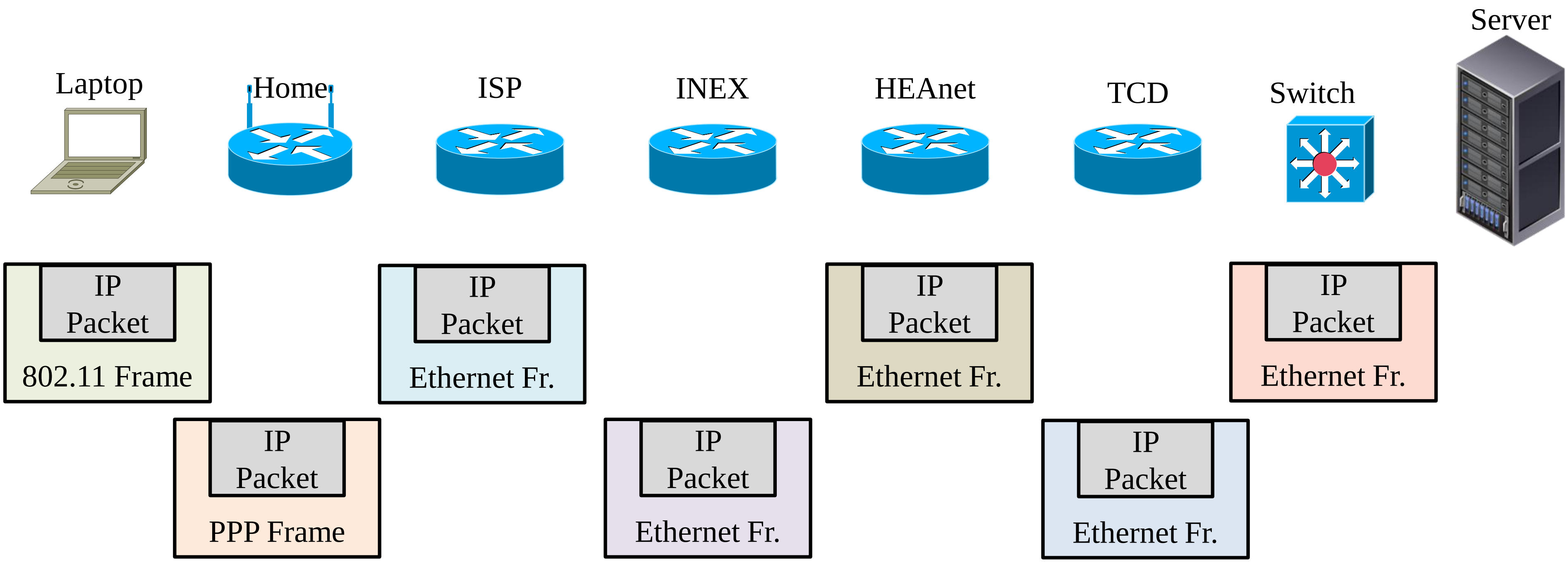
- Trade-off between window size and frame size

Goal

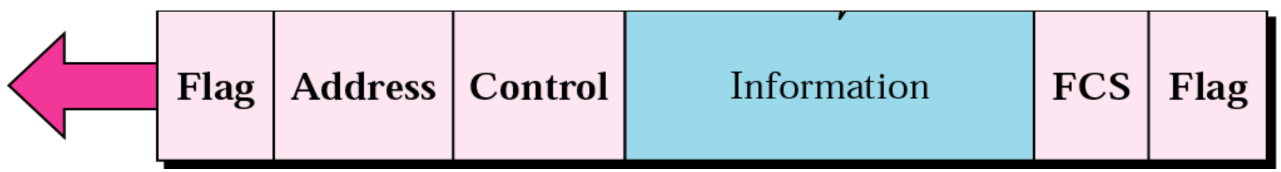


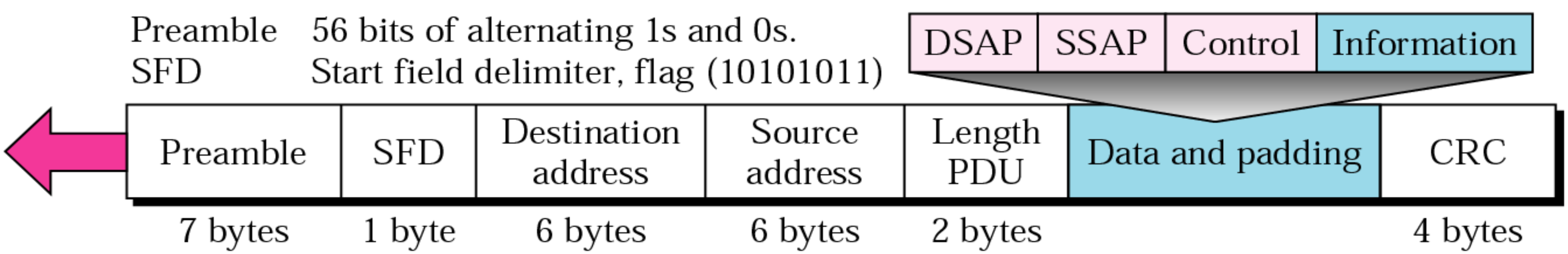
Wifi
Access Point
Ethernet
ADSL Router
PPP
Internet Service Provider
Irish Internet Exchange
HEANET
TCD
Ethernet
Google

Encapsulation



Ethernet & HDLC Header





* Figure is courtesy of B. Forouzan

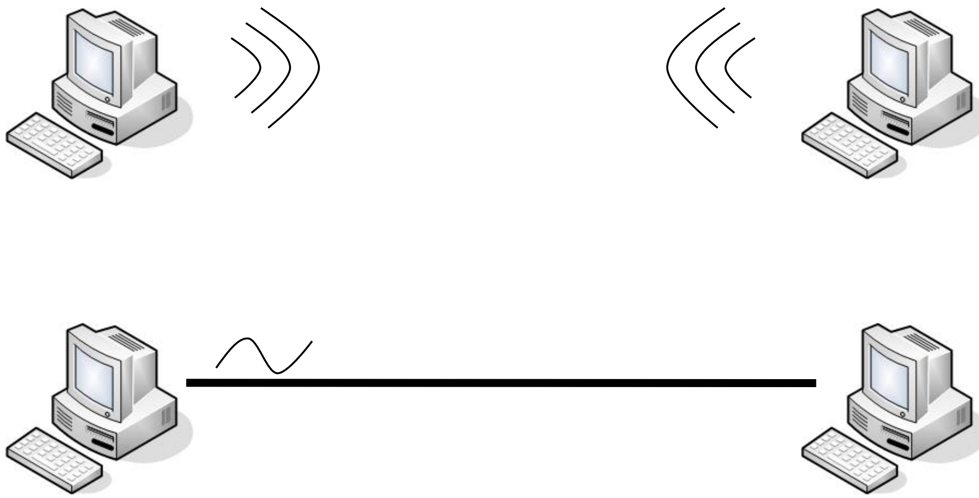
CS2031 Telecommunications II

Error Detection and Correction

Errors in Transmissions

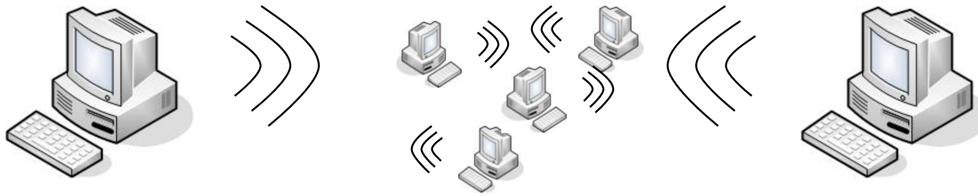
- Causes for Errors
- Types of Errors
- Detection of Errors
- Correction of Errors

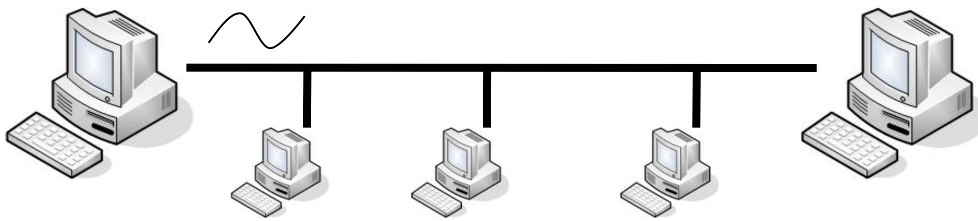
Terminal to Terminal Comms



- Either over dedicated or shared medium

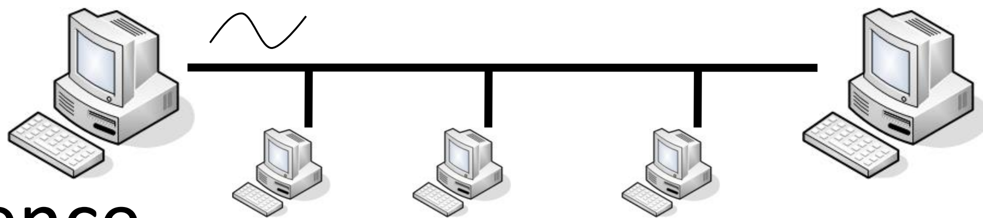
Causes for Errors





- Interference
  - Collision with communication from other nodes
  - Electrical interference from third parties
  - Thermal interference

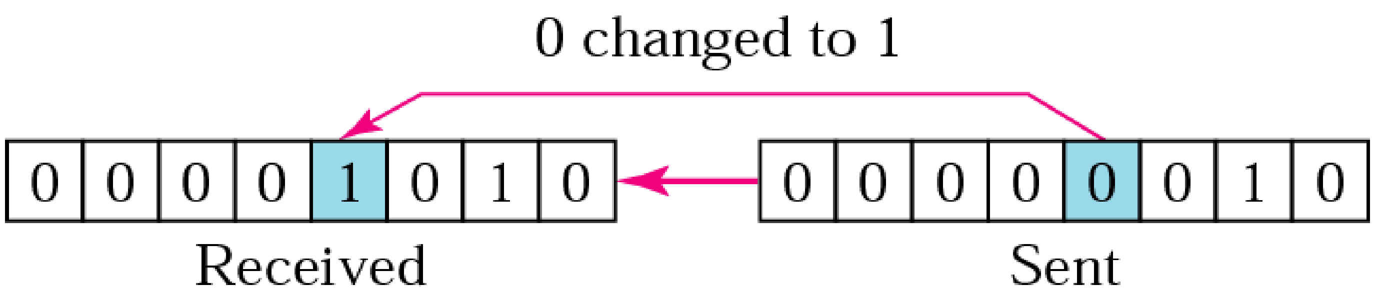
Causes for Errors



- Interference
  - Collision with communication from other nodes
  - Electrical interference from third parties
  - Thermal interference

Types of Errors: Single-Bit Error

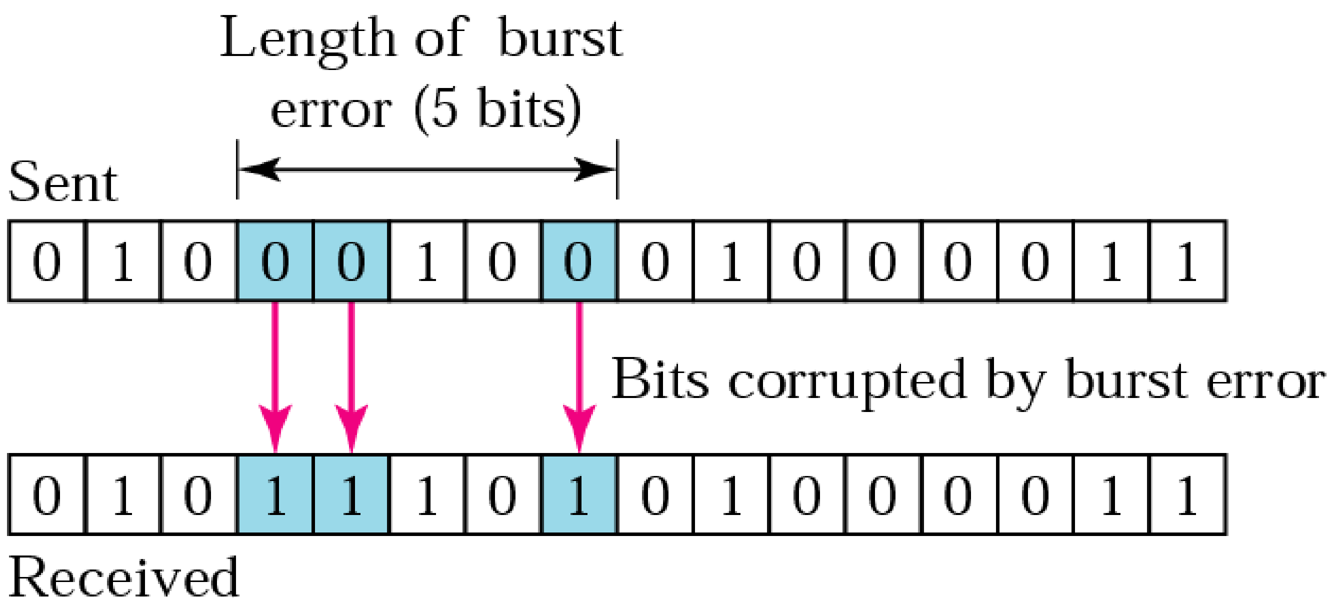
In a single-bit error, only one bit in the data unit has changed.



* Figure is courtesy of B. Forouzan

Types of Errors: Burst Error

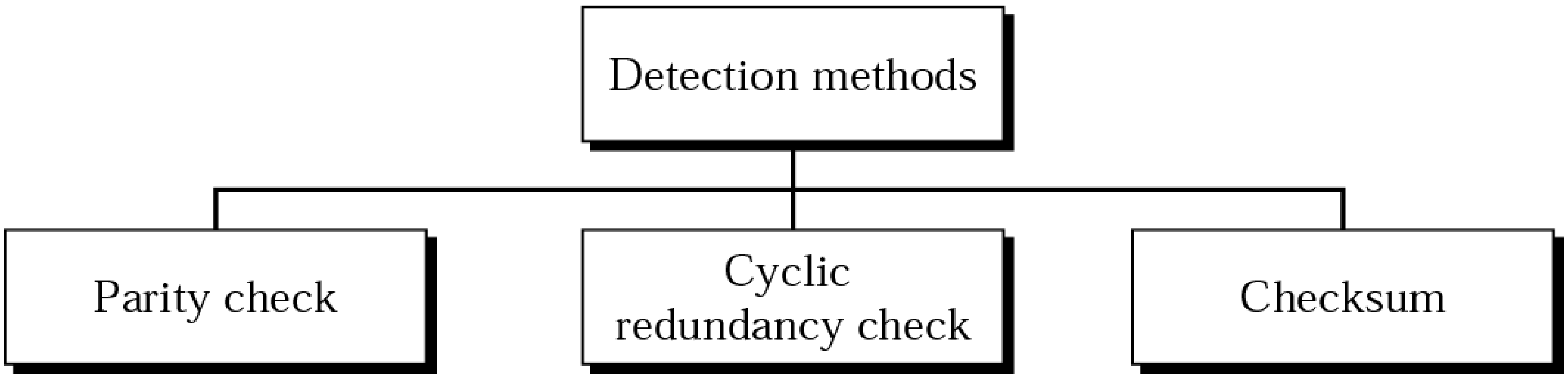
A burst error means that 2 or more bits in the data unit have changed



* Figure is courtesy of B. Forouzan

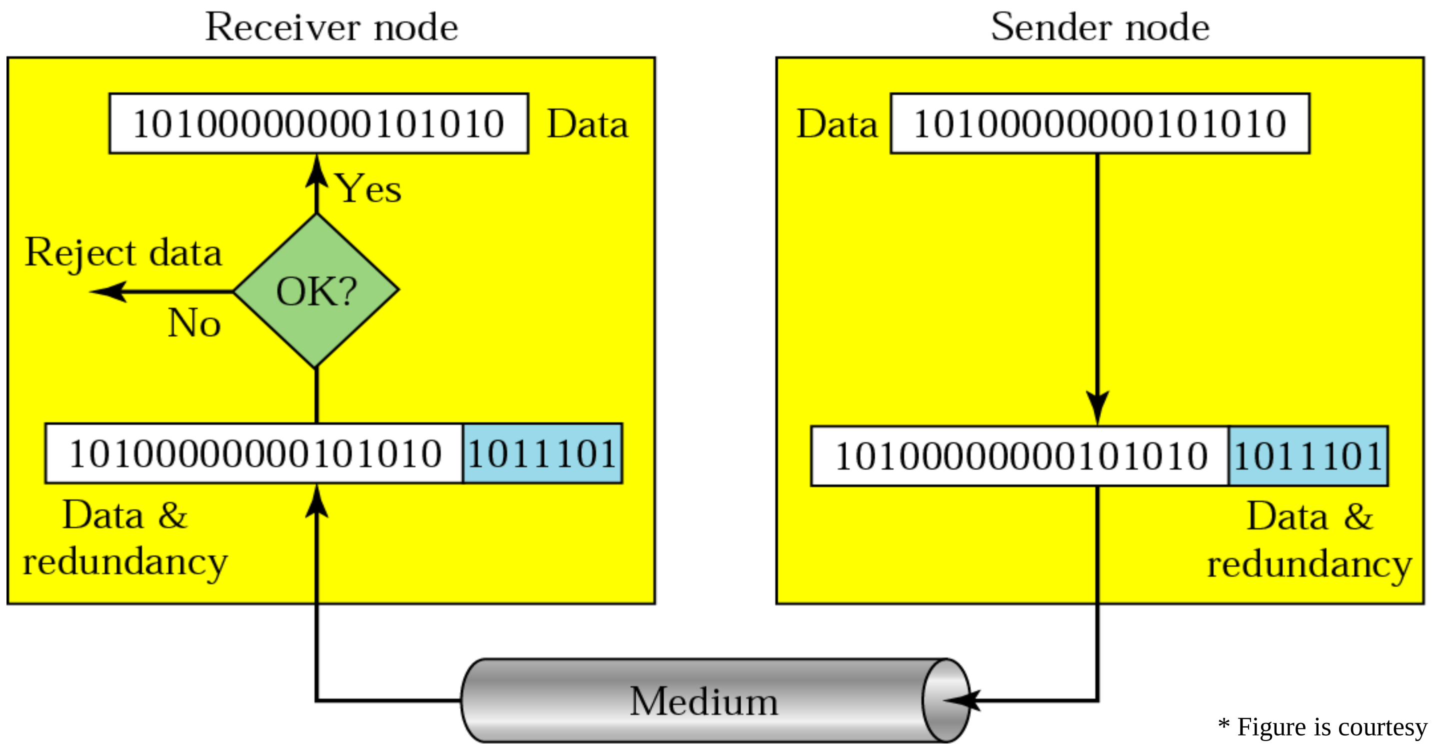
Detection of Errors

- Redundancy



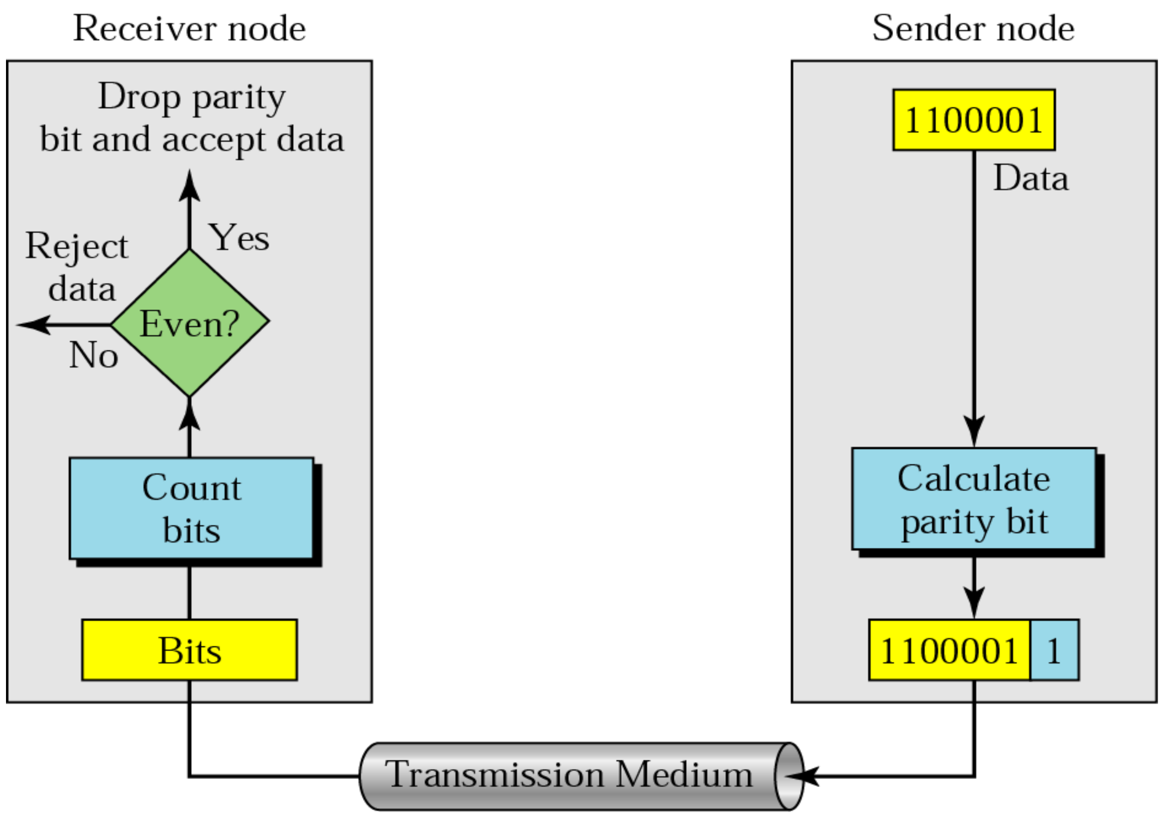
Redundancy

Error detection uses the concept of redundancy, which means adding extra bits for detecting errors at the destination



* Figure is courtesy of B. Forouzan

Even-Parity Concept



A parity bit is added to every data unit so that the total number of 1s is even (or odd for odd-parity).

* Figure is courtesy of B. Forouzan

Even-Parity: Example - Sender

- Assume you want to send the following:

1110111 1101111 1110010 1101100 1100100

- The following bits are actually sent:

1110111**0** 1101111**0** 1110010**0** 1101100**0** 1100100**1**

Even-Parity: Example - Receiver

11101110 11011110 11100100 11011000 11001001

6 6 4 4 4

- The receiver counts the 1s in each character and comes up with even numbers (6, 6, 4, 4, 4). The data are accepted.

111$\boxed{1}$110 11011110 11101$\boxed{1}$00 11011000 11001001

7 6 5 4 4

- The receiver counts the 1s in each character and comes up with even and odd numbers (7, 6, 5, 4, 4). The receiver knows that the data are corrupted, discards them, and asks for retransmission.

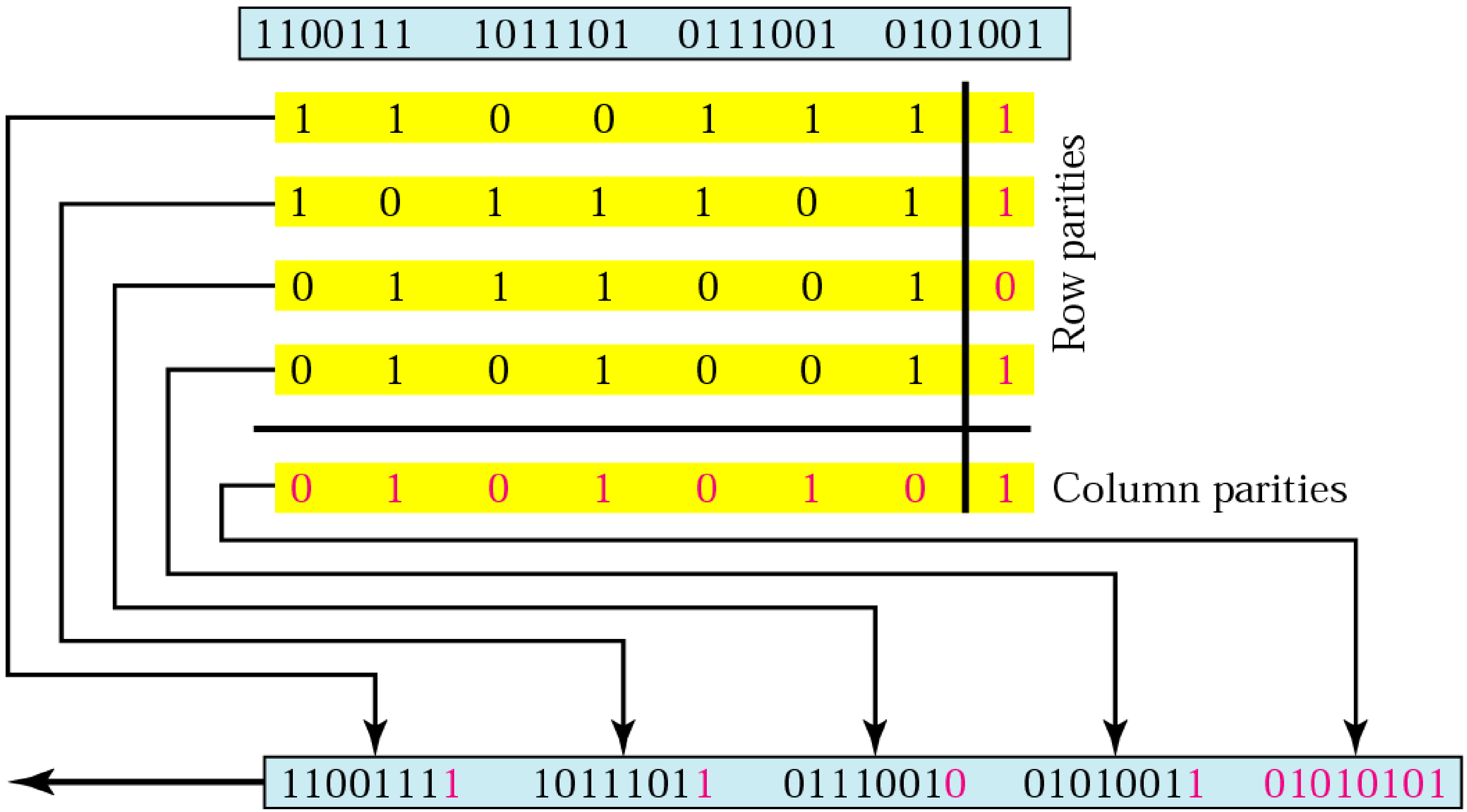
Simple Parity Check

- Can detect all single-bit errors

- Can detect burst errors only if the total number of errors in each data unit is odd

Two-Dimensional Parity Check

In two-dimensional parity check, a block of bits is divided into rows and a redundant row of bits is added to the whole block.



* Figure is courtesy of B. Forouzan

Example: 2D-Parity Check

Suppose the following block is sent:

10101001   00111001   11011101   11100111   10101010

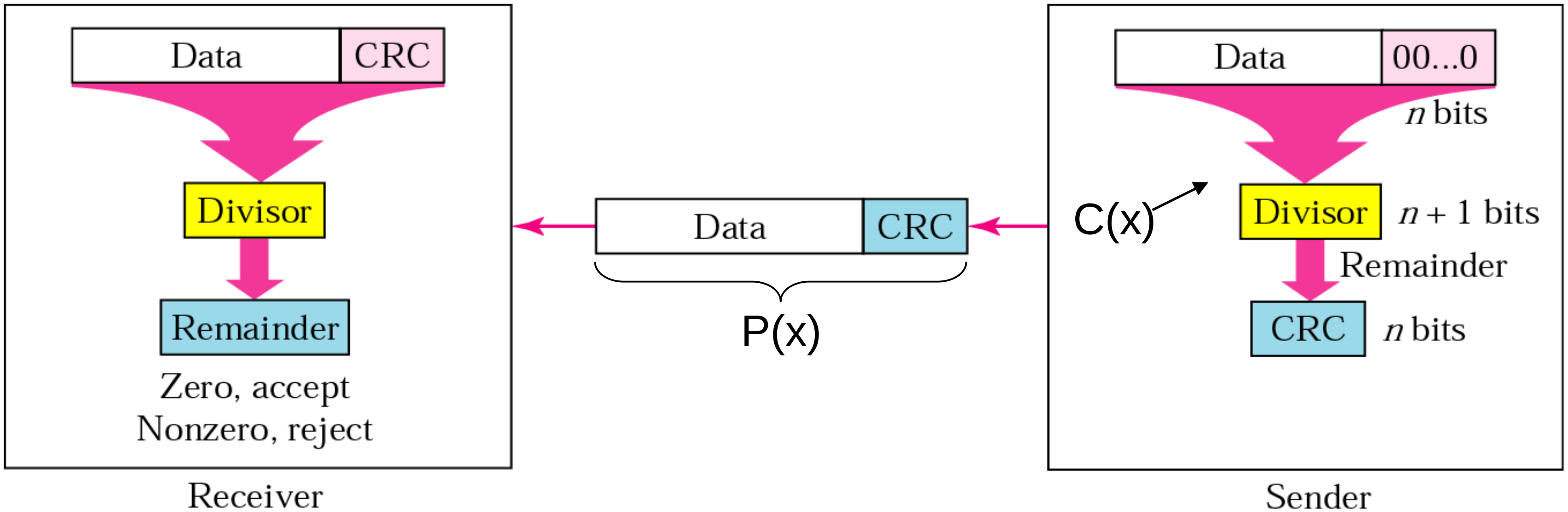
However, it is hit by a burst noise of length 8, and some bits are corrupted.

1010**0011**   **1000**1001   11011101   11100111   10101010

When the receiver checks the parity bits, some of the bits do not follow the even-parity rule and the whole block is discarded.

10100011   10001001   11011101   11100111   **1**0**101**0**1**0

Cyclic Redundancy Check (CRC)



- P(x) divided by C(x) = 0
- (P(x)+remainder) divided by C(x) should be != 0

* Figure is courtesy of B. Forouzan

Division – Decimal&Binary

39 / 20 = 1 + 19

100111 / 10100 = 1 + 10011

32 4 2 1 16 4 16 2 1

CRC Calculation

- CRC Calculation → Polynomial Division

  not Binary Division!!!

$$x^3 + 4x^2 + 3x + 12 \quad / \quad x^2 + 3 \quad = \quad x + 4$$

$$
\begin{array}{l}
x^3 + 4x^2 + 3x + 12 \\
x^3 \qquad\quad + 3x \\
\qquad 4x^2 \qquad + 12 \\
\qquad\qquad\qquad \text{-----} \\
\qquad\qquad\qquad\quad 0
\end{array}
$$

CRC Calculation

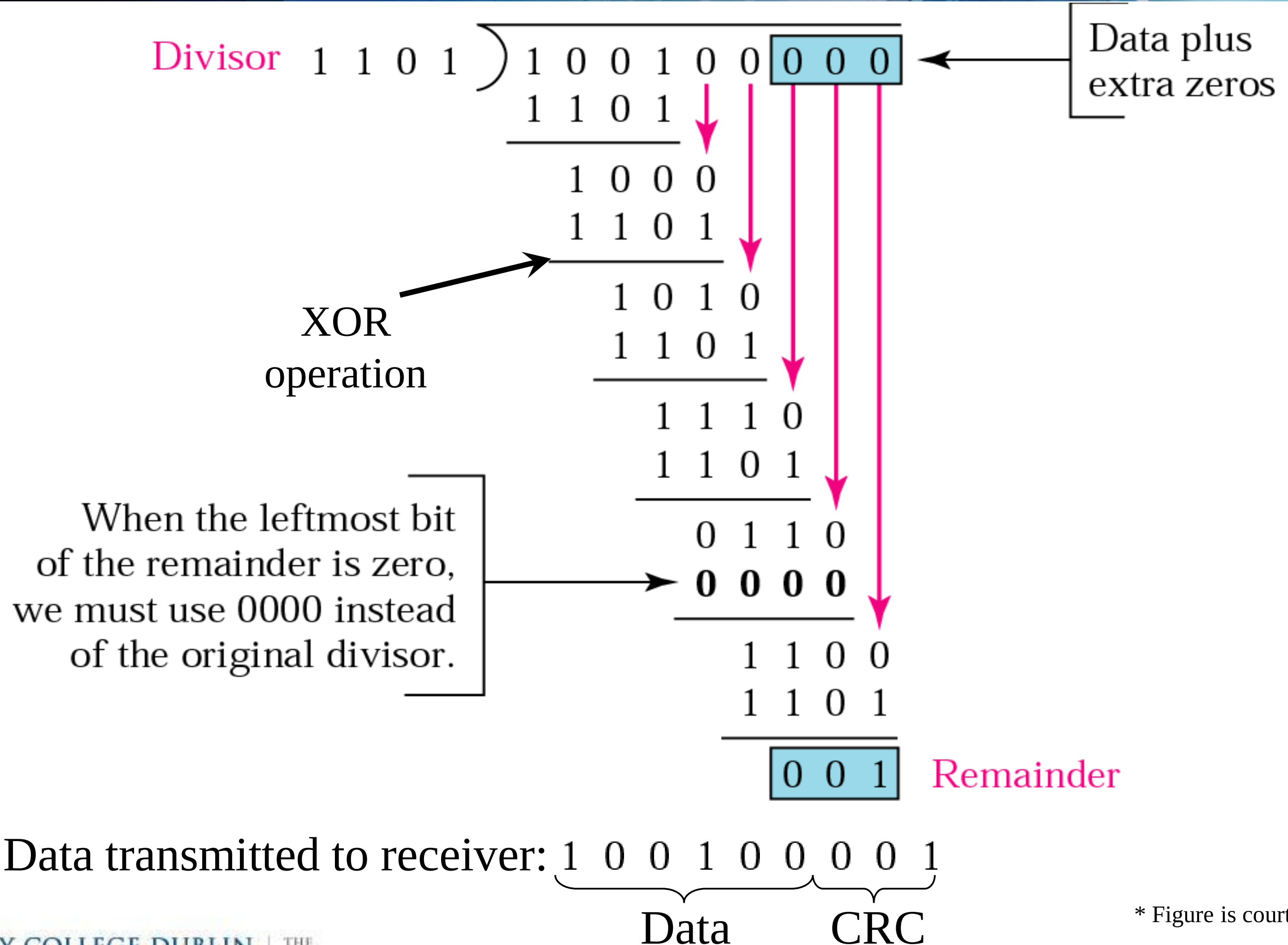
- CRC Calculation→ Polynomial Division

  not Binary Division!!!
- CRC: Coefficient r={0,1}

100010000000000001011

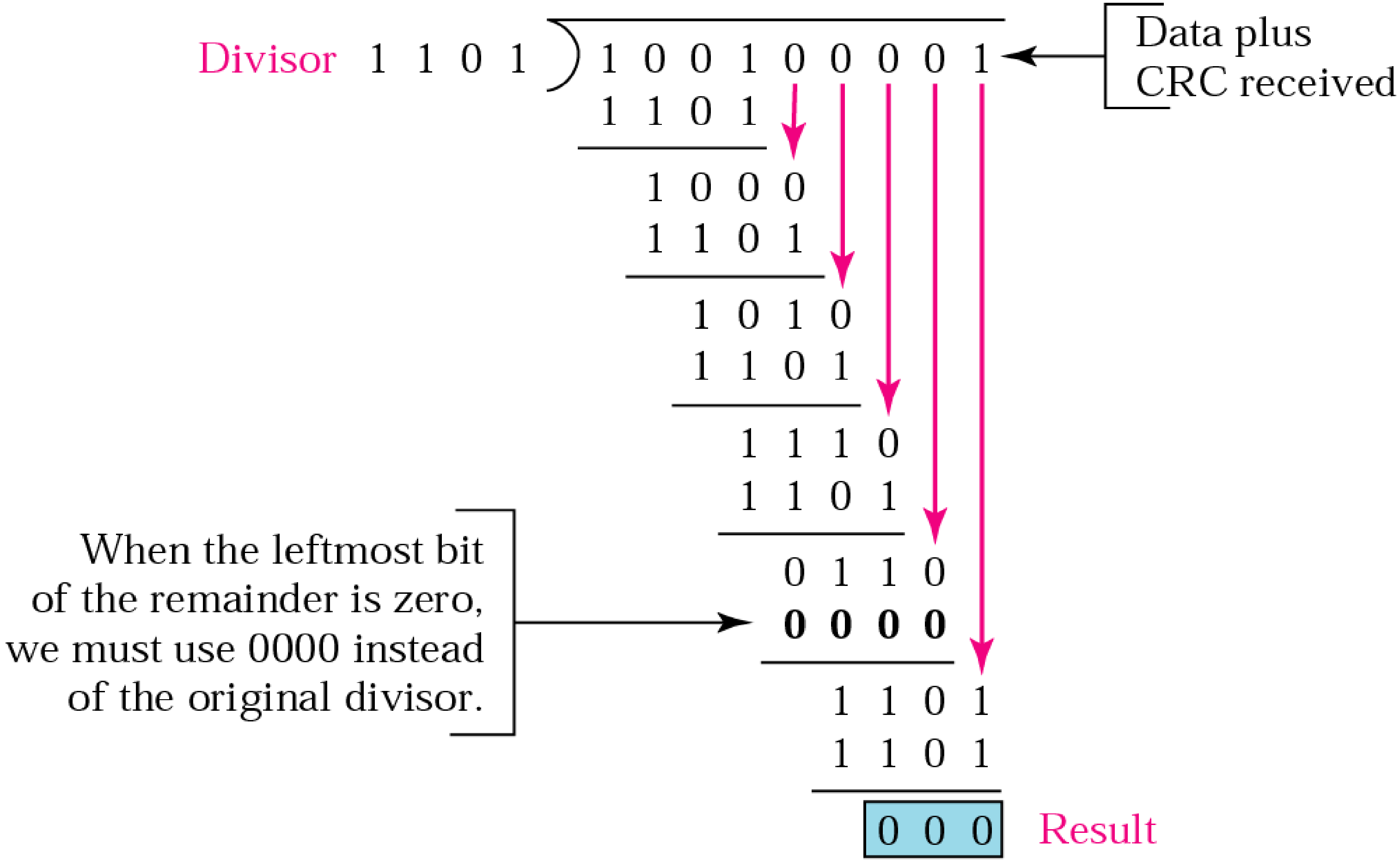
$x^{20} + x^{15} + x^4 + x + 1 \quad / \quad x^{16} + x^{12} + x^5 + 1$

CRC: Sender



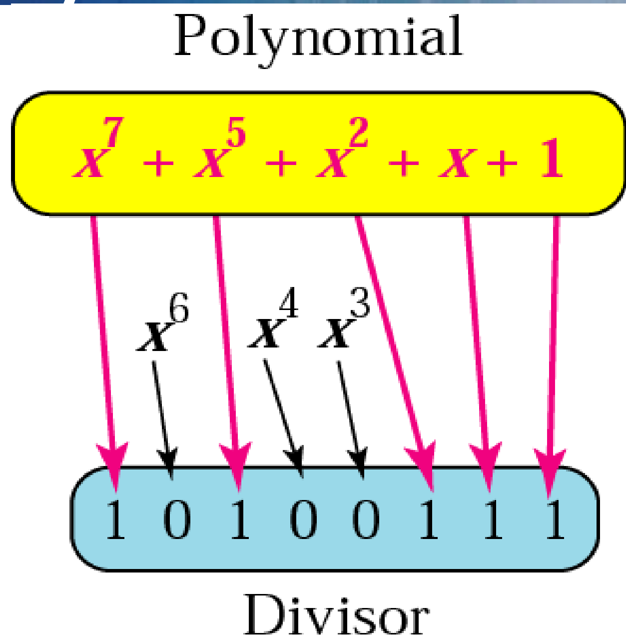
* Figure is courtesy of B. Forouzan

CRC: Receiver



* Figure is courtesy of B. Forouzan

Polynomial Notation



- Rules for selecting divisor:
  - It should not be divisible by x
  - It should be divisible by x+1

* Figure is courtesy of B. Forouzan

Polynomials

- We cannot choose x (binary 10) or $x^2 + x$ (binary 110) as polynomial because both are divisible by x.
- However, we can choose $x + 1$ (binary 11) because it is not divisible by x, but is divisible by $x + 1$. We can also choose $x^2 + 1$ (binary 101) because it is divisible by $x + 1$ (binary division).

* Figure is courtesy of B. Forouzan

Standard Polynomials

Name	Polynomial	Application
CRC-8	$x^8 + x^2 + x + 1$	ATM header
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^2 + 1$	ATM AAL
CRC-16	$x^{16} + x^{12} + x^5 + 1$	HDLC
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	LANs

* Figure is courtesy of B. Forouzan

CRC Performance

- Can detect all burst errors that effect an odd number of bits
- Can detect all burst errors of the length less than or equal to the degree of the polynomial
- Can detect with a very high probability burst errors of a length greater than the degree of the polynomial.

CRC-12 Performance

The CRC-12

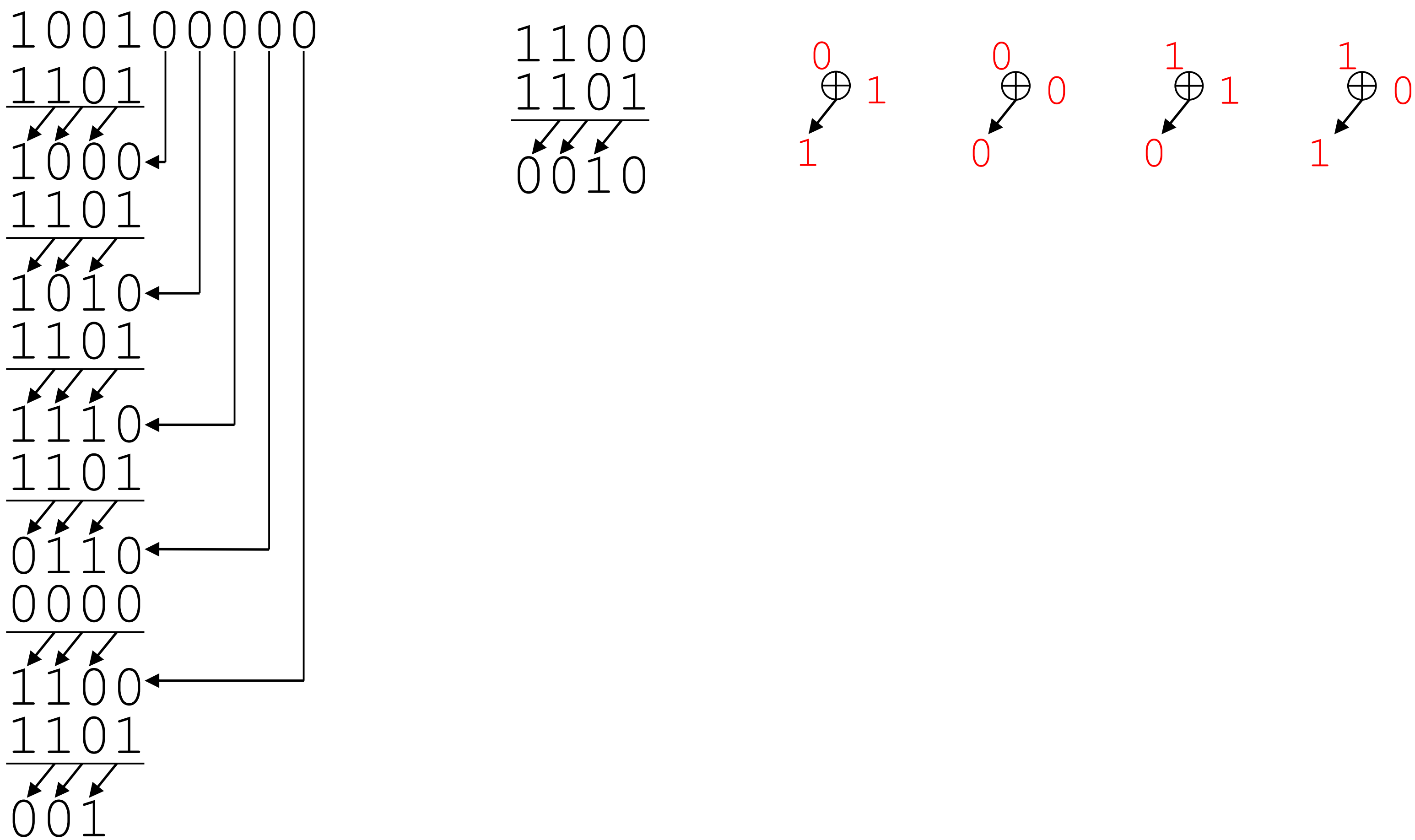
$$x^{12} + x^{11} + x^3 + x + 1$$

which has a degree of 12, will detect all burst errors affecting an odd number of bits, will detect all burst errors with a length less than or equal to 12, and will detect, 99.97 percent of the time, burst errors with a length of 12 or more.

CRC Calculation

```
1001000000
1101
----
1000
1101
----
1010
1101
----
1110
1101
----
0110
0000
----
1100
1101
----
001
```

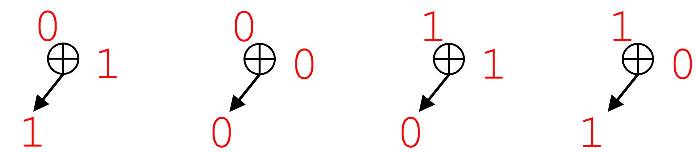
CRC Calculation



CRC Calculation

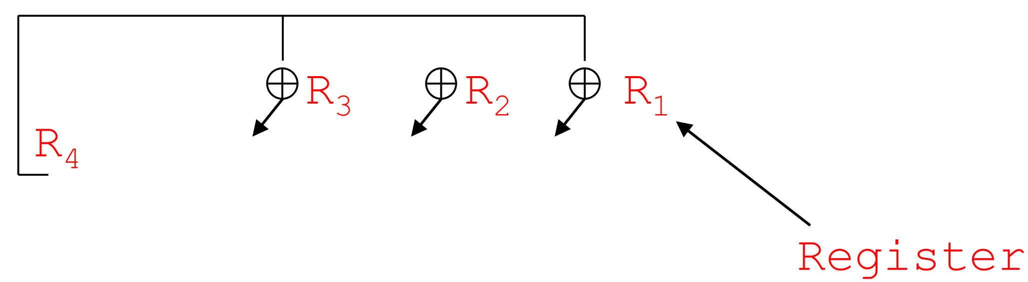
```
100100000
1101
---------
1000
1101
----
1010
1101
----
1110
1101
----
0110
0000
----
1100
1101
----
 001
```

```
1100
1101
----
0010
```

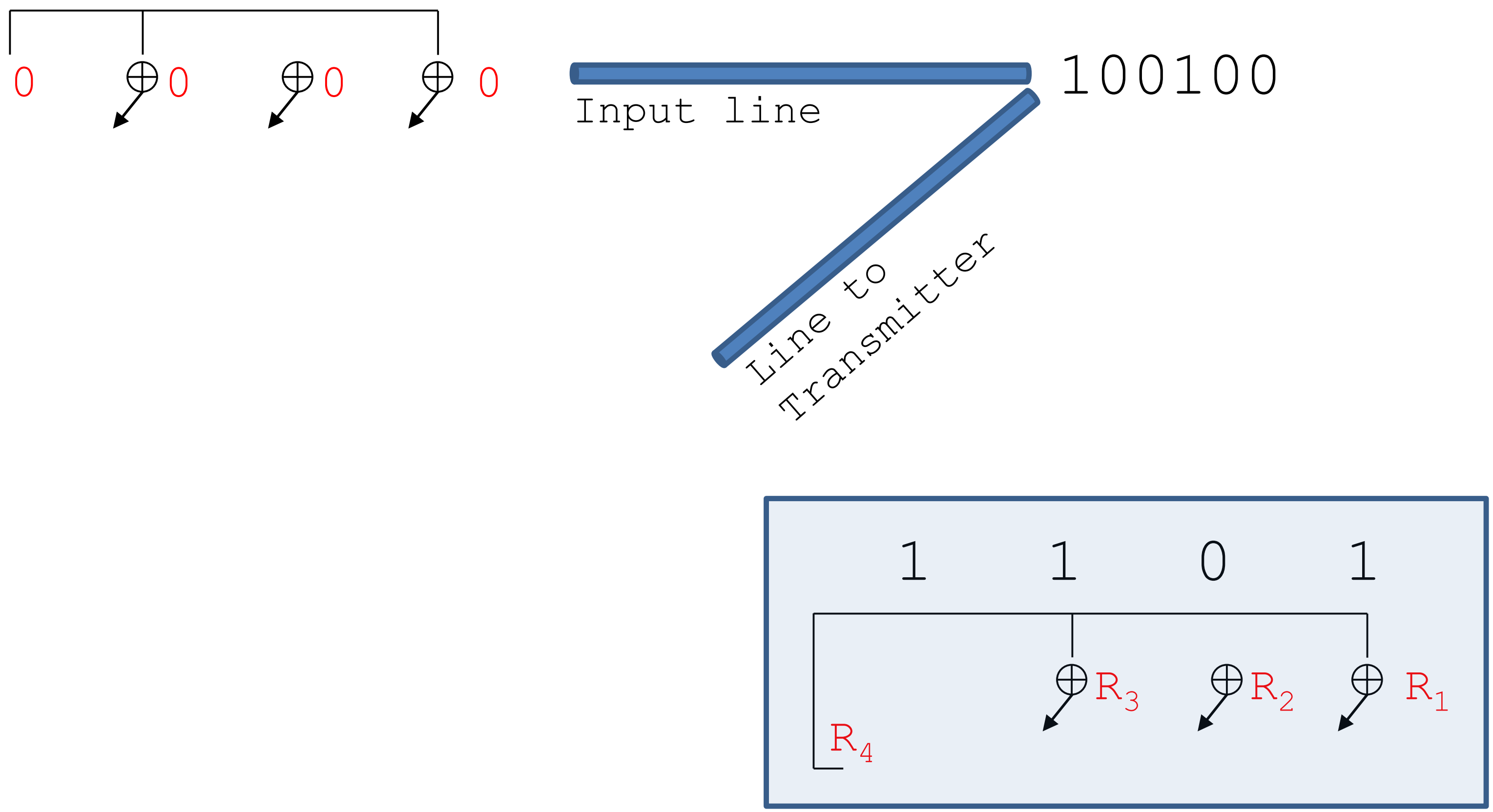


Representation of divisor:

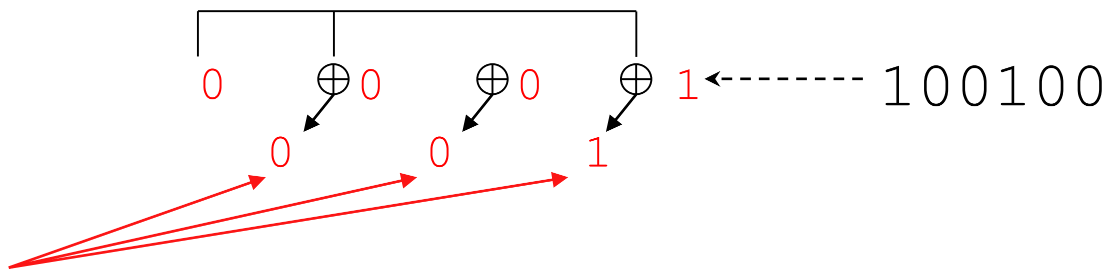
1 1 0 1



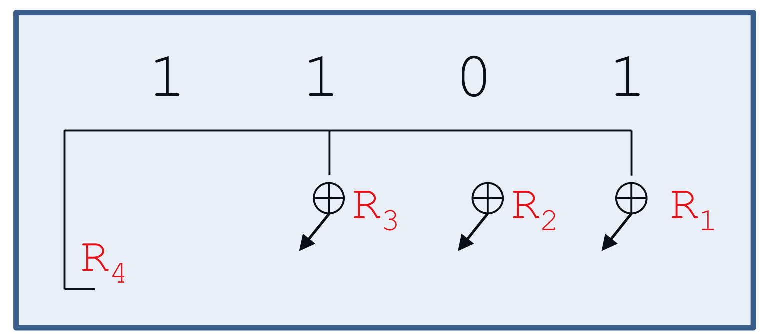
CRC Calculation



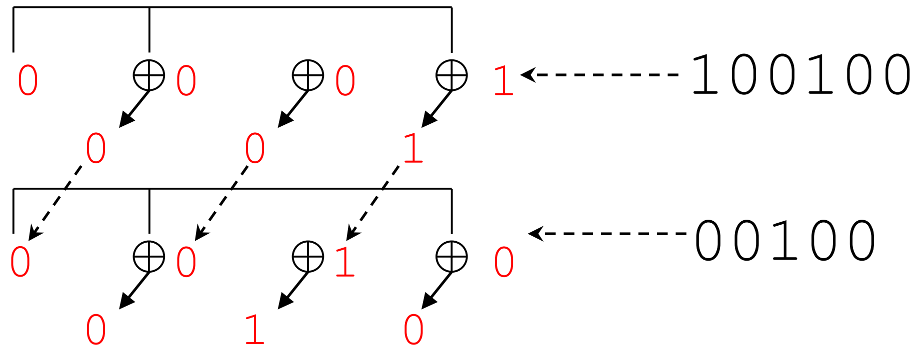
CRC Calculation



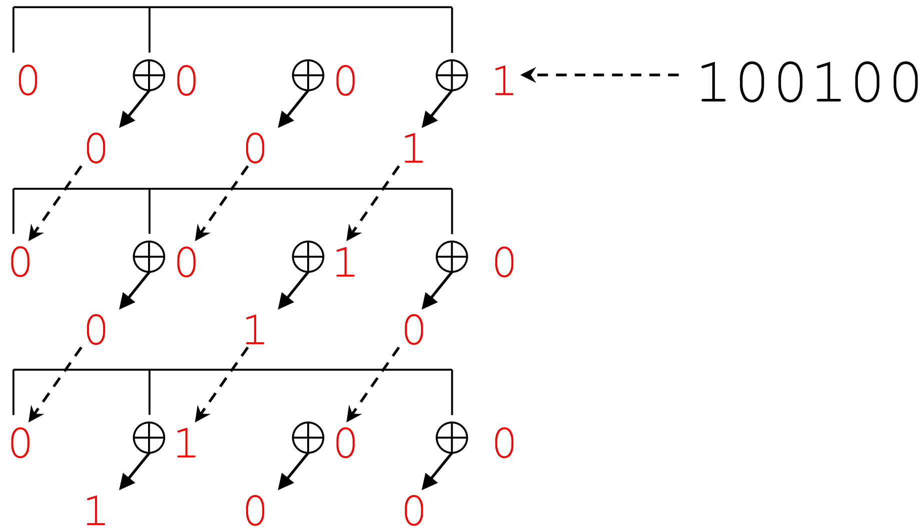
New content for registers R_4, R_3, R_2



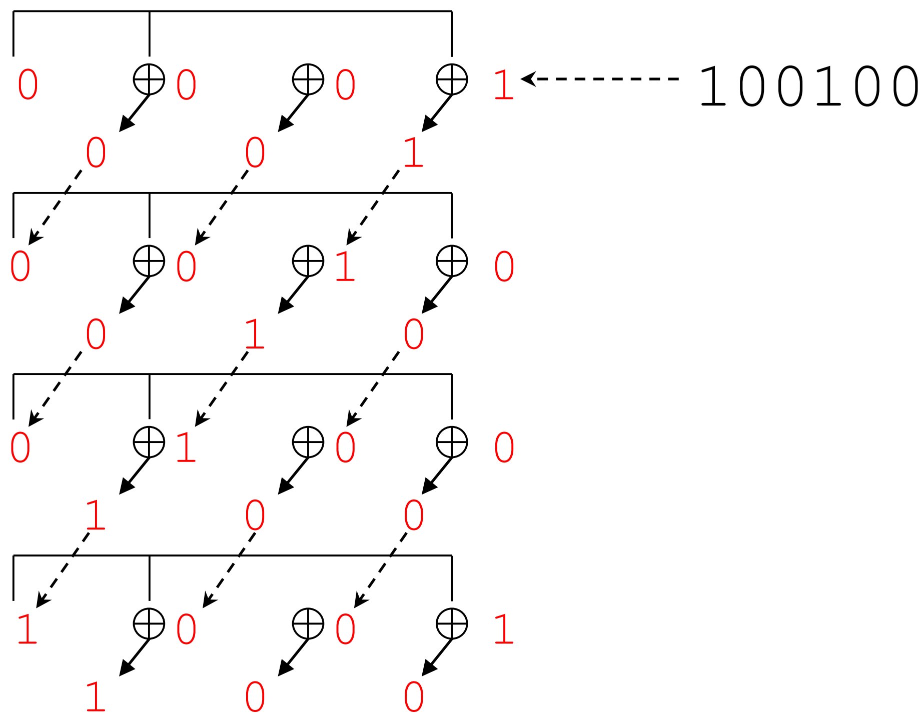
CRC Calculation



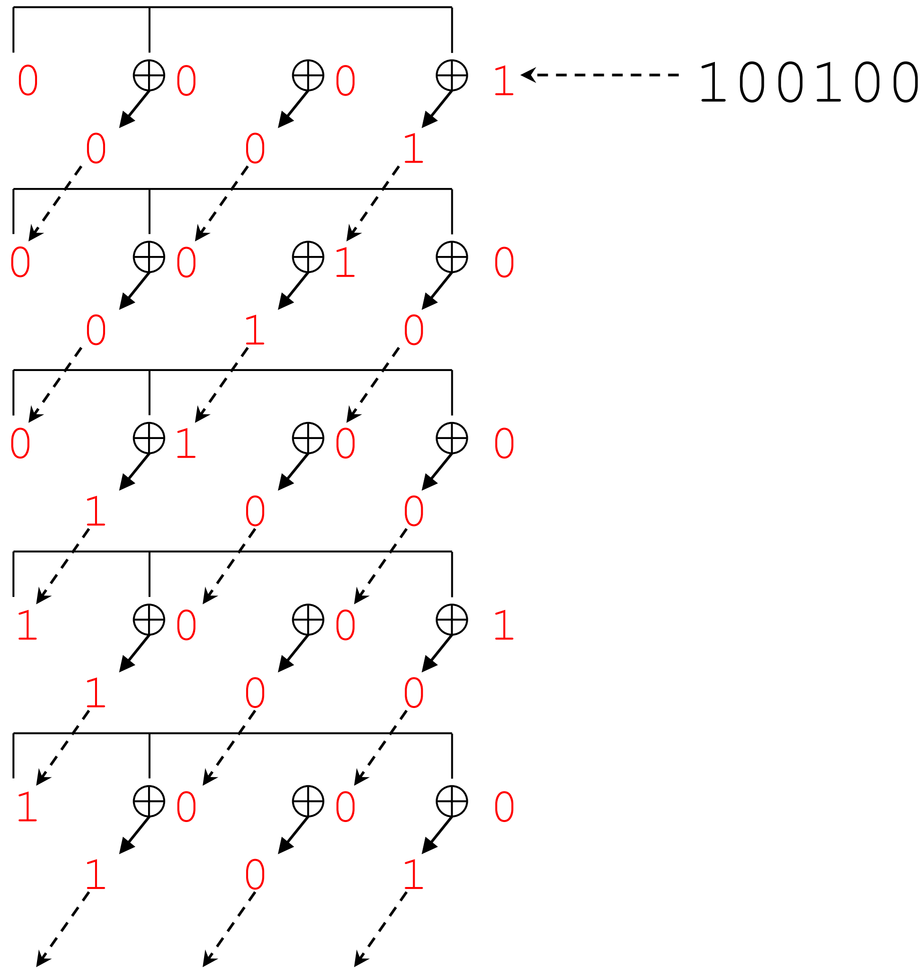
CRC Calculation



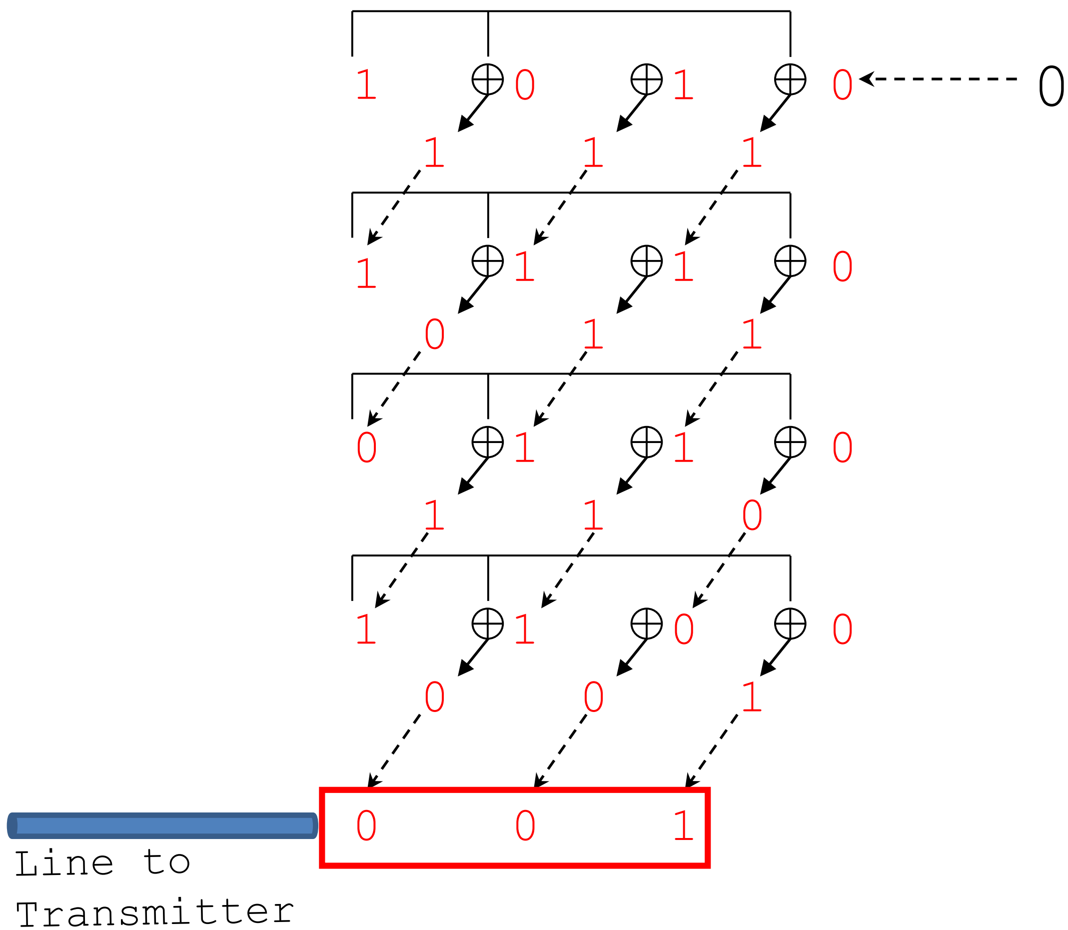
CRC Calculation



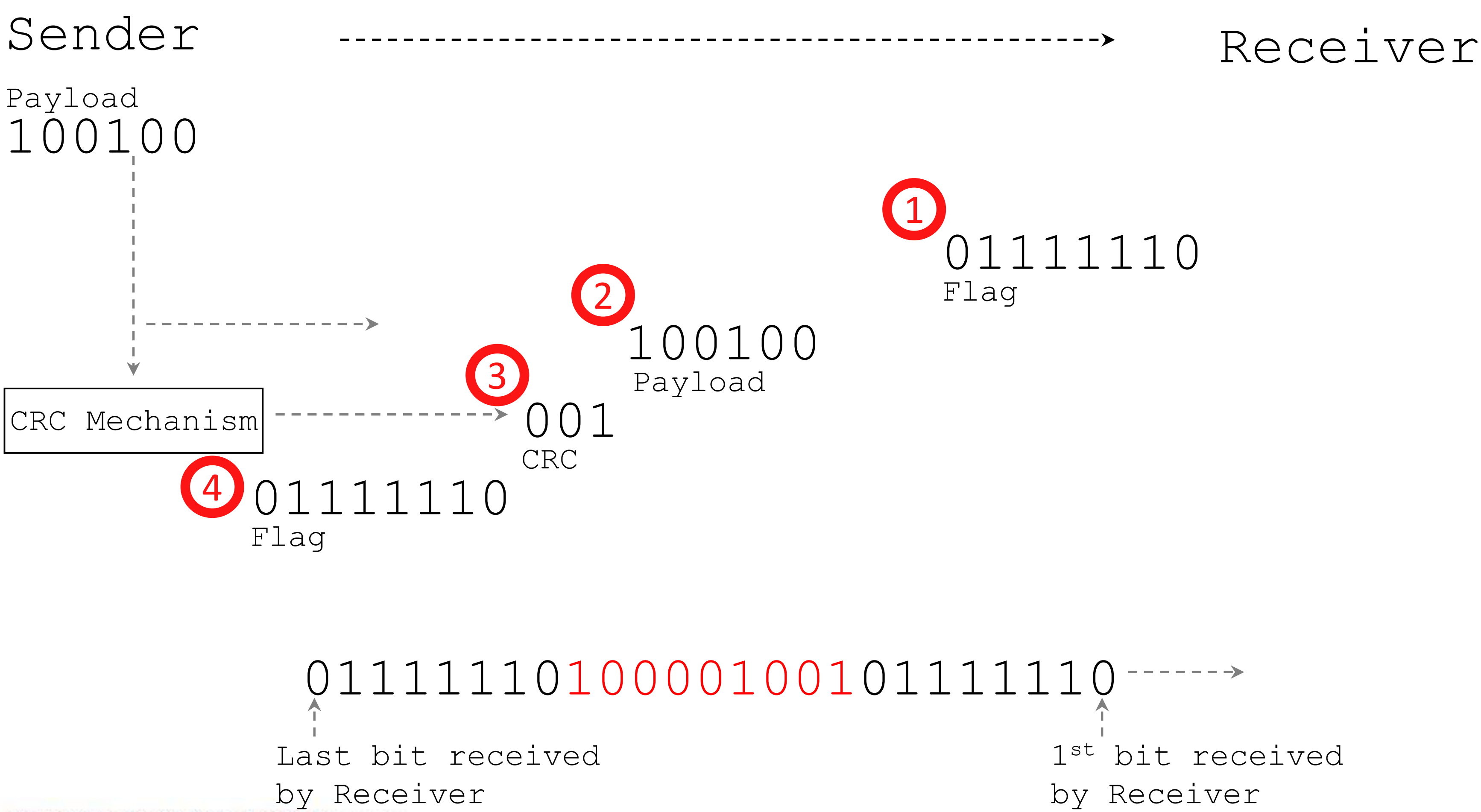
CRC Calculation



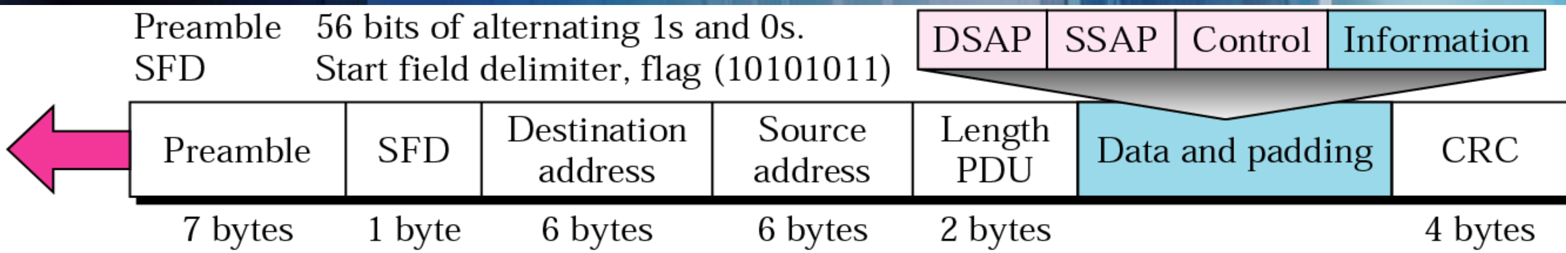
CRC Calculation



CRC Calculation



802.3 MAC Format



- 64-bit frame preamble (10101010) used to synchronize reception
  - 7 bit preamble (10101010) + 1 start flag (10101011)

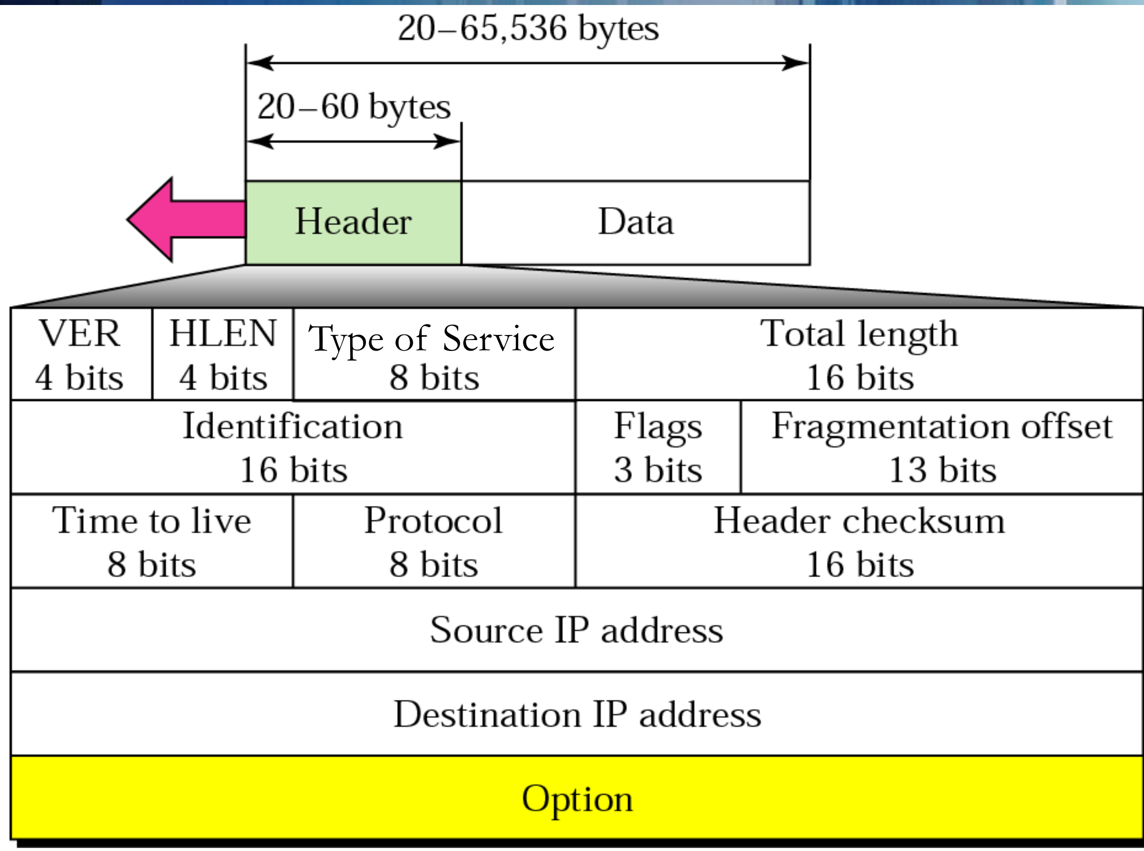
- Maximum frame length: 1536 bytes
  - ⇨ max 1500 bytes payload
- Minimum frame length: 64 bytes
  - ⇨ min 46 bytes payload

* Figure is courtesy of B. Forouzan

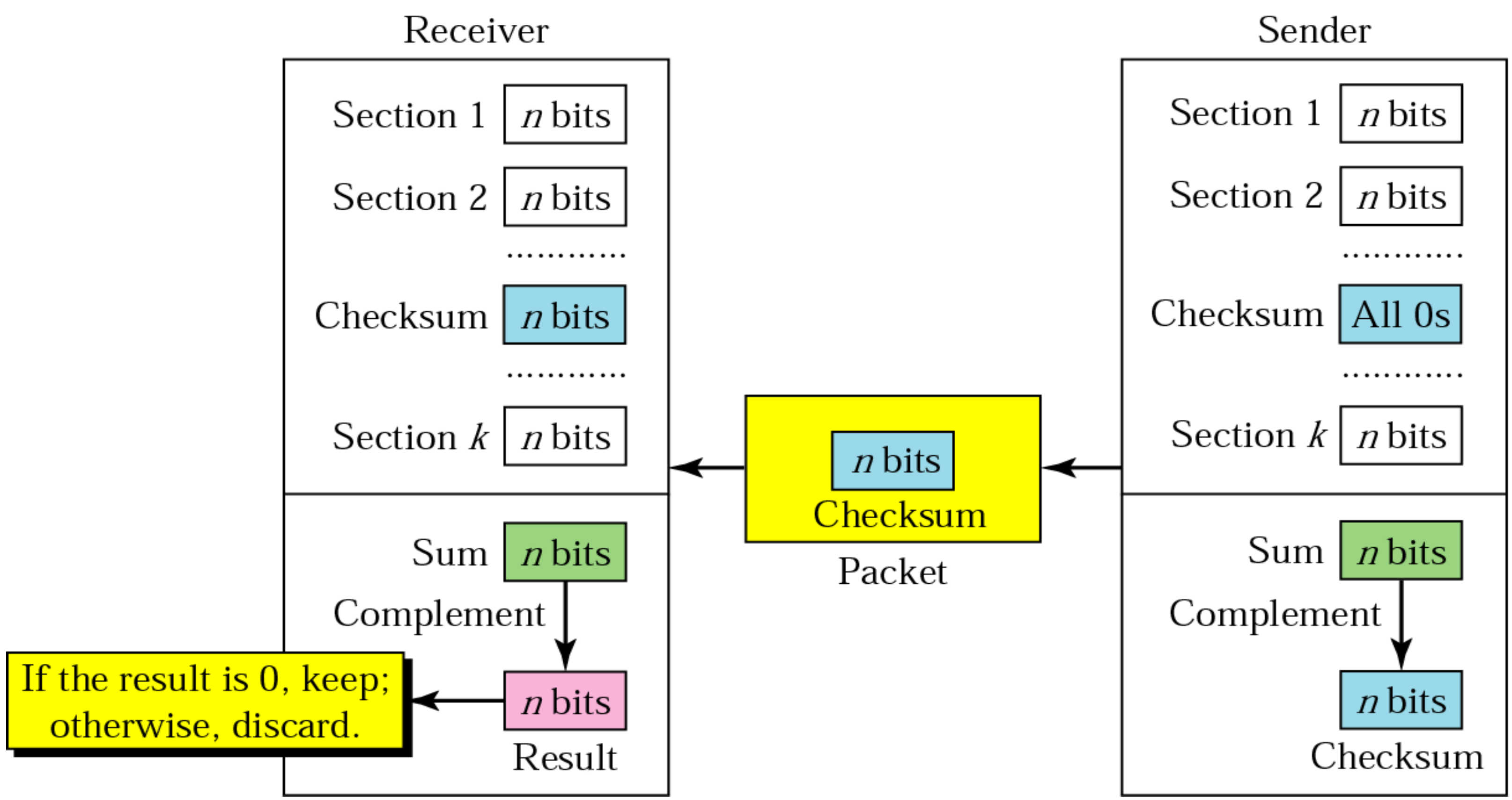
Example from a Linux box

```
wlan0     Link encap:Ethernet  HWaddr 00:0b:81:89:56:ca
          inet addr:192.168.192.12  Bcast:192.168.192.255  Mask:255.255.255.0
          inet6 addr: fe80::20b:81ff:fe89:56ca/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:292 errors:0 dropped:374 overruns:0 frame:0
          TX packets:199 errors:0 dropped:2 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:47787 (46.6 KiB)  TX bytes:26749 (26.1 KiB)
```

IP Datagram


20–65,536 bytes
20–60 bytes
Header
Data
VER
4 bits
HLEN
4 bits
Type of Service
8 bits
Total length
16 bits
Identification
16 bits
Flags
3 bits
Fragmentation offset
13 bits
Time to live
8 bits
Protocol
8 bits
Header checksum
16 bits
Source IP address
Destination IP address
Option

Checksum



* Figure is courtesy of B. Forouzan

Checksum II

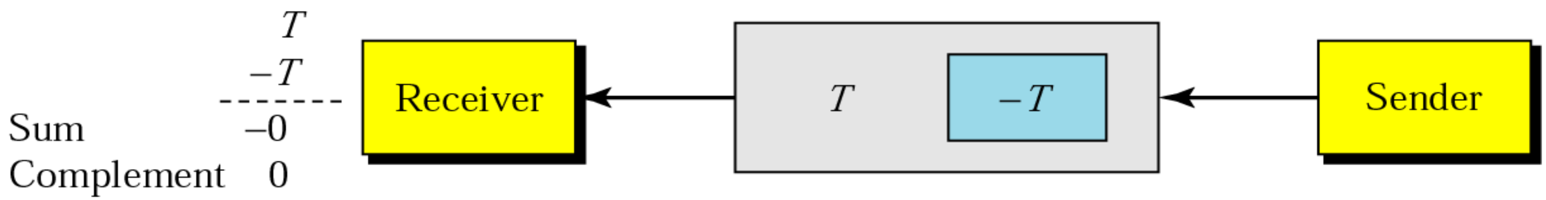
Sender:

The unit is divided into k sections, each of n bits.

All sections are added using one's complement to get the sum.

The sum is complemented and becomes the checksum.

The checksum is sent with the data.



Receiver:

The unit is divided into k sections, each of n bits.

All sections are added using one's complement to get the sum.

The sum is complemented.

If the result is zero, the data are accepted: otherwise, rejected.

* Figure is courtesy of B. Forouzan

Example: Checksum

Sender:

	10101001
	00111001

Sum	11100010
Checksum	**00011101**

The data that is send:

10101001 00111001 **00011101**

Example: Checksum

Sender:

	10101001
	00111001

Sum	11100010
Checksum	**00011101**

The data that is send:

10101001 00111001 **00011101**

Receiver:

	10101001
	00111001
	00011101
Sum	11111111
Complement	**00000000**

Complement: **00000000**

means that the frame is OK.

Example: Checksum

Sender:

	10101001
	00111001

Sum	11100010
Checksum	**00011101**

The data that is send:

10101001 00111001 **00011101**

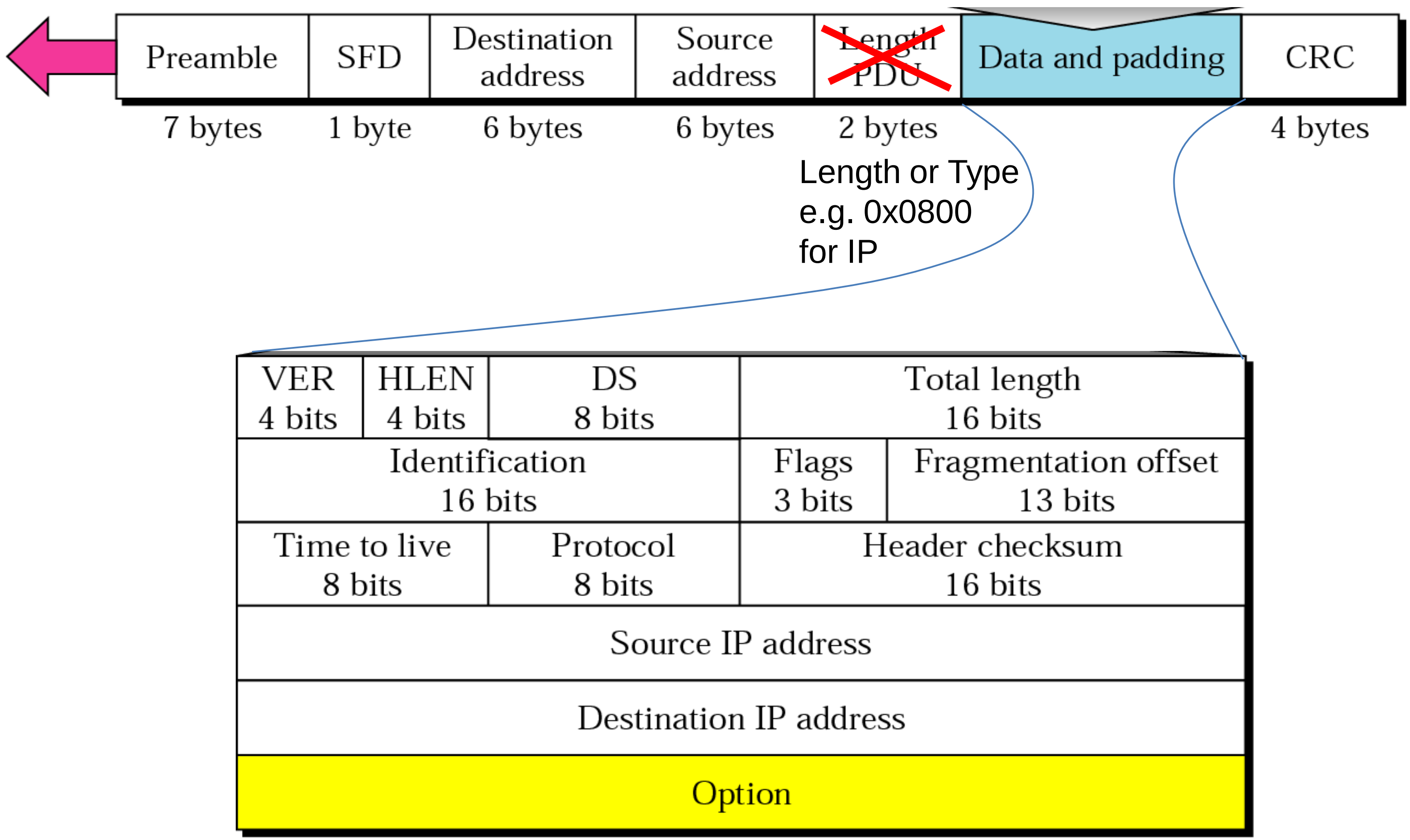
Receiver:

		10101111
		11111001
		00011101
Partial Sum	**1**	11000101
Carry		**1**
Sum		11000110
Complement		**00111001**

Complement: **00111001**

means that the frame is corrupted.

Ethernet & IP



* Figure is courtesy of B. Forouzan

Sample TCP / IP Packet

```
0000  00 07 e9 7c 22 fc 00 11 93 85 e0 c4 08 00 45 00   ...|".........E.
0010  00 2c db 26 40 00 3f 06 0e 77 86 e2 20 37 86 e2   .,.&@.?..w.. 7..
0020  24 33 01 bd 12 3f 3d fa 0f b6 a8 6f 87 c0 50 18   $3...?=....o..P.
0030  bc 40 8a 7c 00 00 85 00 00 00 00 00               .@.|........
```

Ethernet Header:
src addr: 00 07 e9 7c 22 fc
dest addr: 00 11 93 85 e0 c4

IP Header:
src addr: 134.226.36.55
dest addr: 134.226.36.51

TCP Header:
src port: 445
dest port: 4671

NetBios Information

IP Header Checksum

The IP header is generally 20 byte and can be divided into units of 2 bytes/16 bits to calculate the checksum

Summary: Detection of Errors

- Parity Check
- Cyclic Redundancy Check (CRC)
- Checksum

Correction of Errors

- Error Correction through Retransmission
  - Parity, CRC, Checksum determine validity
  - If not valid, discard and wait for sender to retransmit

- Forward Error Correction
  - Determine the corrupted bit or bits at the receiver

Hamming Code

11	10	9	8	7	6	5	4	3	2	1
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

- Redundancy bits distributed throughout data bits
- Individual redundancy bits work as parity bits for specific data bits
  - e.g. r_1 is the parity bit for all odd numbers

    3 = binary 0011   7= binary 0111

    5 = binary 0101   9= binary 1001

* Figure is courtesy of B. Forouzan

Redundancy Bits Calculation

r_1 will take care of these bits.

11		9		7		5		3		1
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

r_2 will take care of these bits.

11	10			7	6			3	2	
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

r_4 will take care of these bits.

				7	6	5	4			
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

r_8 will take care of these bits.

11	10	9	8							
d	d	d	r_8	d	d	d	r_4	d	r_2	r_1

* Figure is courtesy of B. Forouzan

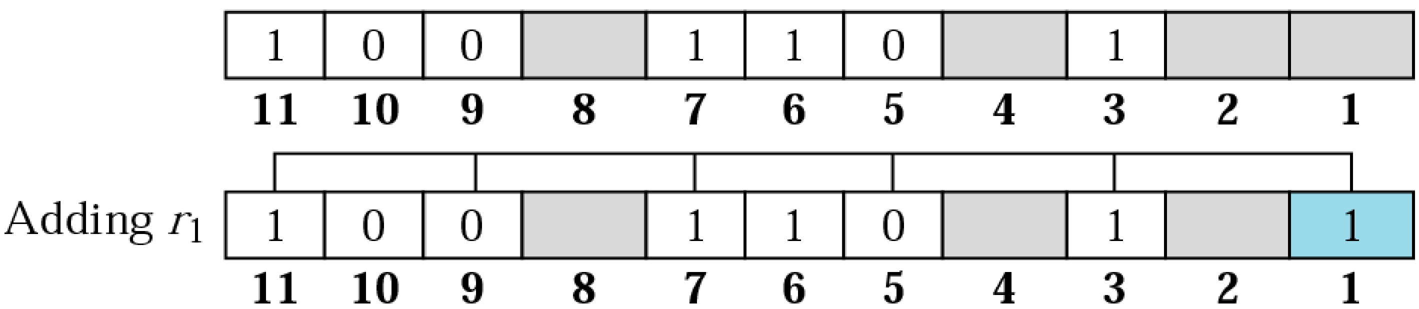
Redundancy Bit Calculation

1	0	0		1	1	0		1		
11	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**

Data:
1 0 0 1 1 0 1

* Figure is courtesy of B. Forouzan

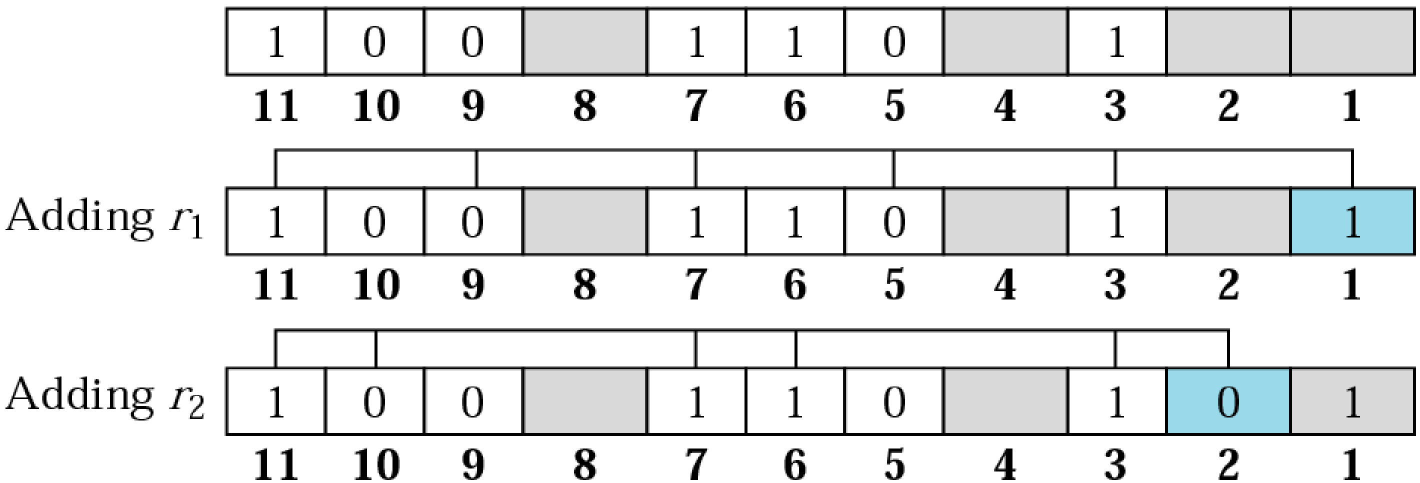
Redundancy Bit Calculation



Data:
1 0 0 1 1 0 1

* Figure is courtesy of B. Forouzan

Redundancy Bit Calculation

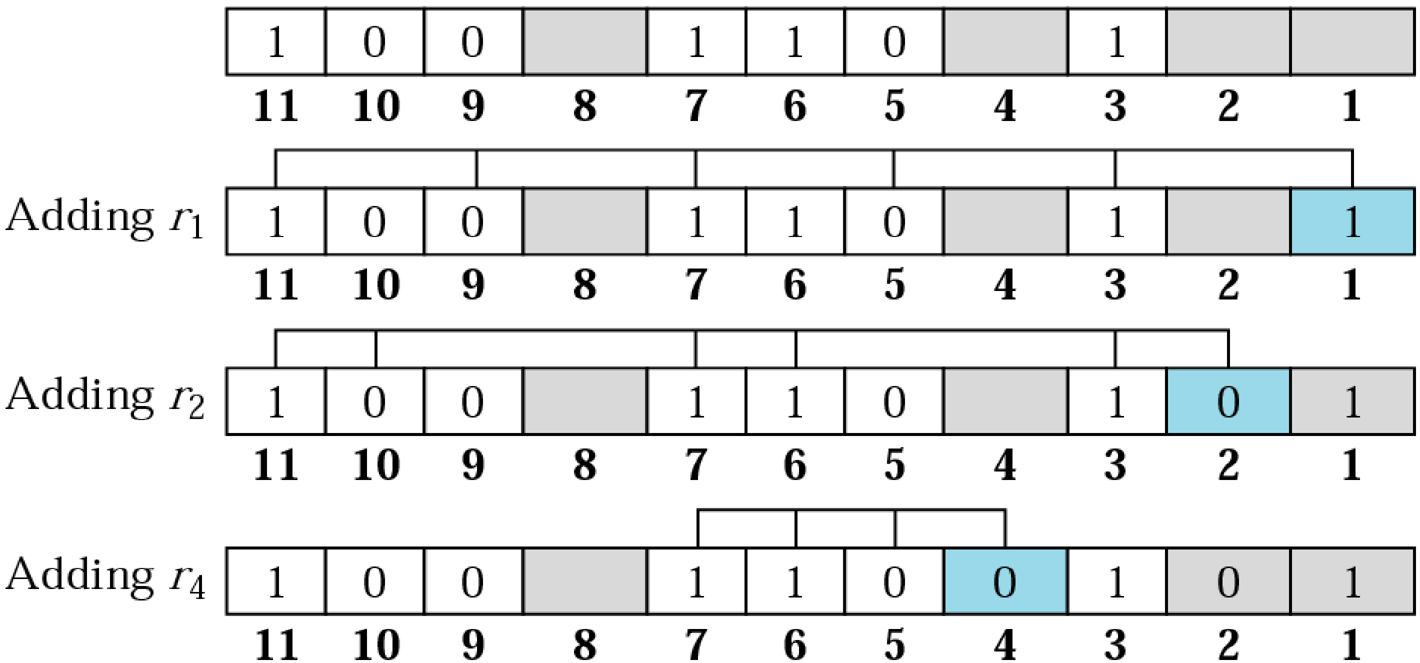


Data:
1 0 0 1 1 0 1

* Figure is courtesy of B. Forouzan

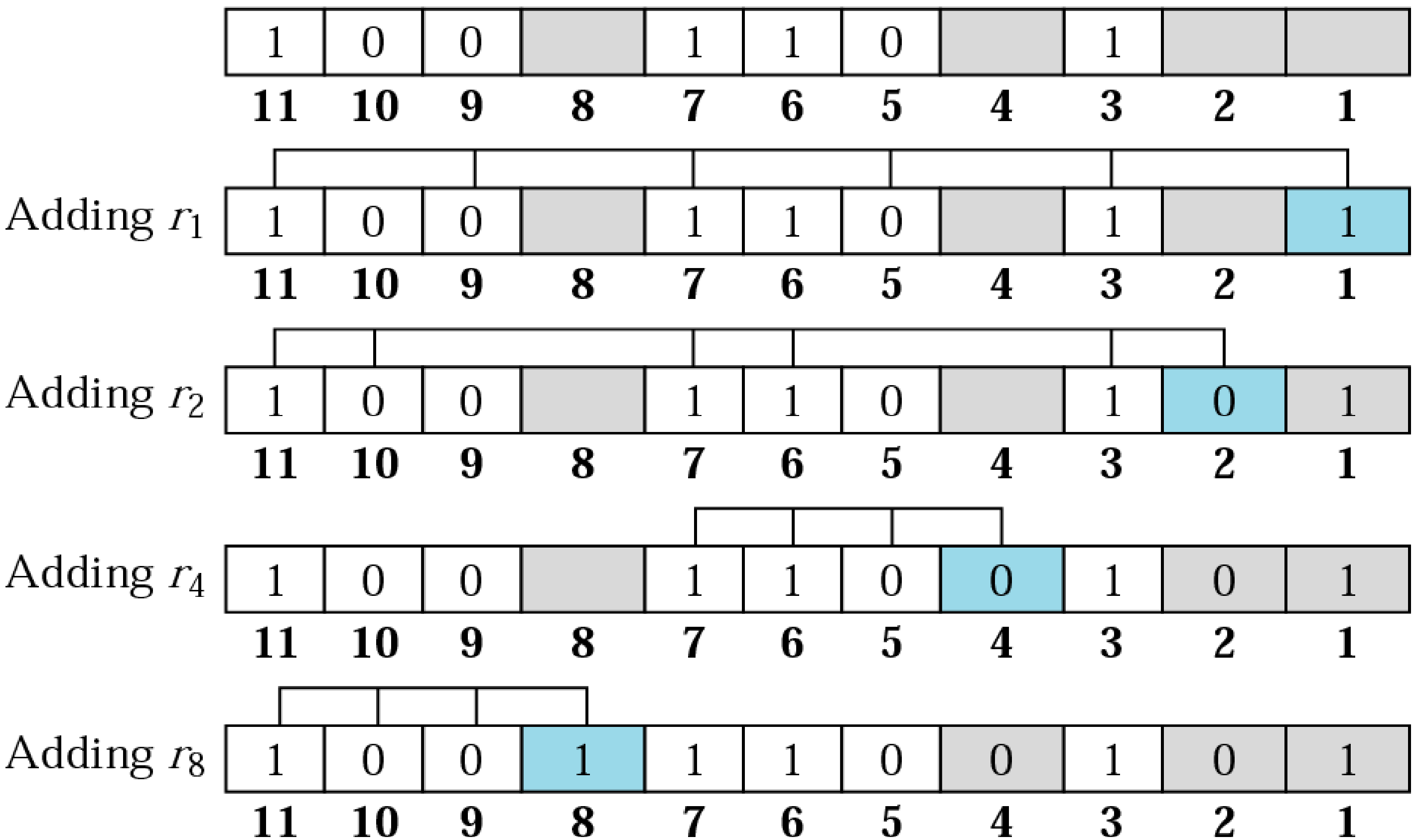
Redundancy Bit Calculation

Data:
1 0 0 1 1 0 1



* Figure is courtesy of B. Forouzan

Redundancy Bit Calculation

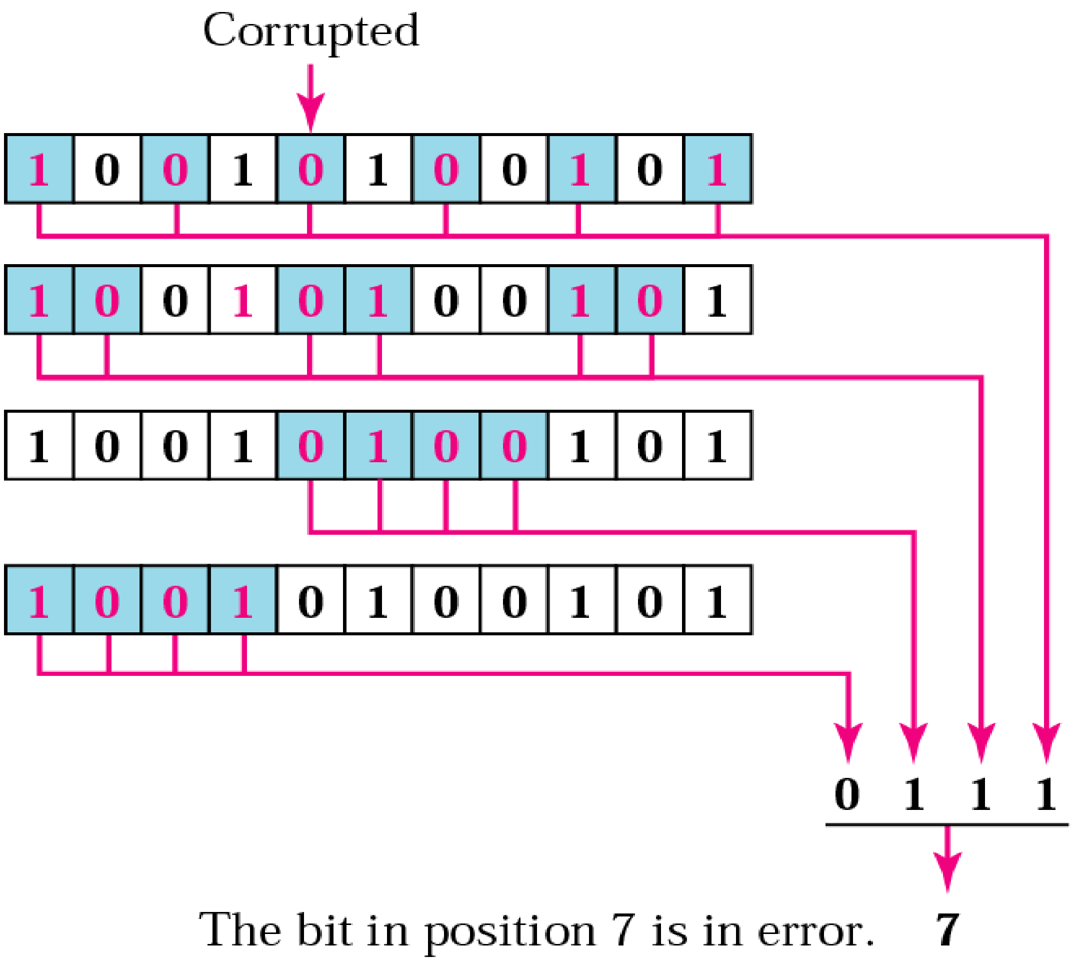


Data:
1 0 0 1 1 0 1

Code:
1 0 0 1 1 1 0 0 1 0 1

* Figure is courtesy of B. Forouzan

Error Detection using Hamming Code



* Figure is courtesy of B. Forouzan

Data and Redundancy Bits

Number of data bits m	Number of redundancy bits r	Total bits m + r
1	2	3
2	3	5
3	3	6
4	3	7
5	4	9
6	4	10
7	4	11

* Figure is courtesy of B. Forouzan

Summary

- Types of Errors
  - Single-Bit & Burst Errors
- Detection of Errors
  - Parity Check / 2D Parity Check
  - CRC (Sequence of data)
  - Checksum (Chunks of data)
- Correction of Errors
  - Error Correction by Retransmission
  - Forward Error Correction – Hamming Code


That's all folks
Telecomms II