Error Detection and Correction Techniques

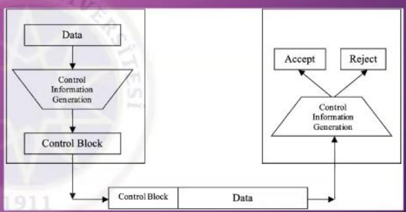
- Data Link Layer (in OSI model)
- Error reasons
  - Attenuation
  - Delay Distortion
    - Video + Voice
    - Problem in time sensitive conditions
  - Noise in the communication environment
    - Thermal noise
      - Random electron motion
    - Intermodulation noise
    - CrossTalk
    - Impulse Noise

Error Types

	Data Sent	Data Received
• Single bit error	1 Ø Ø Ø Ø 1 1 1	1 Ø 1 Ø Ø 1 1 1
• Multi bit error	1 Ø Ø Ø Ø 1 1 1	1 Ø 1 Ø Ø Ø 1 1
• Error bursts	1 Ø Ø Ø Ø 1 1 1	1 Ø Ø 1 1 Ø Ø 1

Error Detection

- Both sides have original data?
- Sending data twice?
- Control block?
  - 4 different types
    - VRC (Vertical Redundency Code)
    - LRC (Longitudial Redundency Code)
    - CRC (Cyclic Redundency Check)
    - Checksum



VRC (Vertical Redundency Code)

- Parity check
- Simple error coding technique
- The number of errors should be **odd**
- XOR operation

	VRC	Data						
Data Sent	1	Ø	1	Ø	Ø	1	1	Ø
Data Received 1	1	Ø	1	Ø	Ø	1	1	Ø
Data Received 2	1	Ø	1	1	Ø	1	1	Ø
Data Received 3	1	Ø	1	1	Ø	1	Ø	Ø

LRC (Longitudial Redundency Code)

- LRC is 2D-VRC

	Byte 1	Byte 2	Byte 3	Byte 4	LRC
	1	Ø	1	1	1
	Ø	Ø	1	1	Ø
	Ø	1	Ø	1	Ø
	1	1	Ø	1	1
	1	Ø	1	Ø	Ø
	Ø	1	1	Ø	Ø
	1	Ø	Ø	Ø	1
VRC	Ø	1	Ø	Ø	1

1ØØ11Ø1Ø	ØØ11Ø1Ø1	11ØØ11ØØ	1111ØØØØ	1ØØ1ØØ11

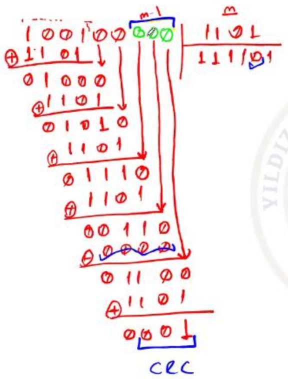
1ØØ11Ø1Ø	Ø111Ø111	11ØØ11ØØ	1Ø11ØØ1Ø	1ØØ1ØØ11

CRC (Cyclic Redundency Check)

- The data to be sent is divided into a predetermined prime polynomial.
- The remainder value is added to the data to be sent as an error control code.
- The remainder zero in receiver side means that error-free transmission.
- Common polynomials used for CRC: 13-bits, 17-bits, 33-bits
  - The number of undetectable errors is almost zero
- Commonly used polynomials in CRC technique:
  - CRC-12 $x^{12}+x^{11}+x^3+x+1$
  - CRC-16 $x^{16}+x^{15}+x^2+1$
  - CRC-ITU $x^{16}+x^{12}+x^5+1$
  - CRC-32 $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$

Example: Data Sent: 100100, polynom: $x^3 + x^2 + 1$, CRC = ?

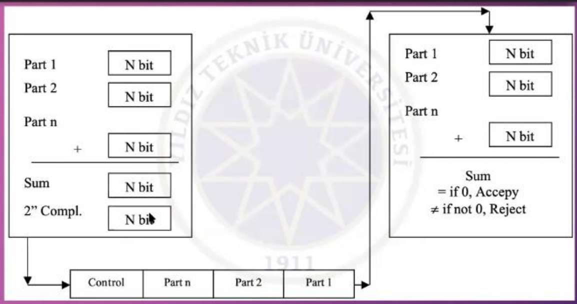
1 1 0 1



Checksum

- The sender divides the data into N-bits parts (usually 16 bits are used).
- The parts are collected using the first complementary arithmetic.
  - In this way, a total value of only N bits is obtained.
- Calculate two's complement using summed value
  - The calculated value is added to the end of the information to be sent.
- The checksum detects all of the odd errors and most of the even numbers.
  - However, if one or more bits in a part are 0 when they are 1, but there is a 0 when 1 in another part, the error will not be understood because there will be no difference in this column sum.

Checksum - Con't



Hamming Code

BLM3051 Data Communication Week 5

- If we sent m bit data, the error occurs in 1,2,...,m bit
- Adding error-free state, the data length will be m+1
- Control block length must be $log_2(m+1) \leq r$
- $m+r$ bit must be sent error-free
- So, control block length must be $log_2(m+r+1) \leq r$
- (1, 2, 4, 8, 16. bits)

B_{11}	B_{10}	B_9	B_8	B_7	B_6	B_5	B_4	B_3	B_2	B_1
D_7	D_6	D_5	R_4	D_4	D_3	D_2	R_3	D_1	R_2	R_1

Furkan Çakmak

Hamming Code

BLM3051 Data Communication Week 5

	R_4	R_3	R_2	R_1	Info
0	0	0	0	0	Error-free
1	0	0	0	1	1. bit error
2	0	0	1	0	2. bit error
3	0	0	1	1	3. bit error
4	0	1	0	0	4. bit error
5	0	1	0	1	5. bit error
6	0	1	1	0	6. bit error
7	0	1	1	1	7. bit error
8	1	0	0	0	8. bit error
9	1	0	0	1	9. bit error
10	1	0	1	0	10. bit error
11	1	0	1	1	11. bit error

- If we sent m bit data, the e
- Adding error-free state, the
- Control block length must b
- $m+r$ bit must be sent error
- So, control block length mu
- (1, 2, 4, 8, 16. bits)

B_{11}	B_{10}	B_9	B_8	B_7	B_6	B_5	B_4	B_3	B_2	B_1
D_7	D_6	D_5	R_4	D_4	D_3	D_2	R_3	D_1	R_2	R_1

Furkan Çakmak

Hamming Code - Con't

BLM3051 Data Communication Week 5

- $R_1 = B_1 \oplus B_3 \oplus B_5 \oplus B_7 \oplus B_9 \oplus B_{11}$
- $R_2 = B_2 \oplus B_3 \oplus B_6 \oplus B_7 \oplus B_{10} \oplus B_{11}$
- $R_3 = B_4 \oplus B_5 \oplus B_6 \oplus B_7$
- $R_4 = B_8 \oplus B_9 \oplus B_{10} \oplus B_{11}$

	R_4	R_3	R_2	R_1	Info
0	0	0	0	0	Error-free
1	0	0	0	1	1. bit error
2	0	0	1	0	2. bit error
3	0	0	1	1	3. bit error
4	0	1	0	0	4. bit error
5	0	1	0	1	5. bit error
6	0	1	1	0	6. bit error
7	0	1	1	1	7. bit error
8	1	0	0	0	8. bit error
9	1	0	0	1	9. bit error
10	1	0	1	0	10. bit error
11	1	0	1	1	11. bit error

B_{11}	B_{10}	B_9	B_8	B_7	B_6	B_5	B_4	B_3	B_2	B_1
D_7	D_6	D_5	R_4	D_4	D_3	D_2	R_3	D_1	R_2	R_1

Furkan Çakmak

Hamming Code - Con’t

- $R_1 = B_1 \oplus B_3 \oplus B_5 \oplus B_7 \oplus B_9 \oplus B_{11}$
- $R_2 = B_2 \oplus B_3 \oplus B_6 \oplus B_7 \oplus B_{10} \oplus B_{11}$
- $R_3 = B_4 \oplus B_5 \oplus B_6 \oplus B_7$
- $R_4 = B_8 \oplus B_9 \oplus B_{10} \oplus B_{11}$

B_{11}	B_{10}	B_9	B_8	B_7	B_6	B_5	B_4	B_3	B_2	B_1
1	0	0		1	1	0		1		

- $R_1 = B_3 \oplus B_5 \oplus B_7 \oplus B_9 \oplus B_{11}$ $= 1 \oplus 0 \oplus 1 \oplus 0 \oplus 1$ $= 1$
- $R_2 = B_3 \oplus B_6 \oplus B_7 \oplus B_{10} \oplus B_{11}$ $= 1 \oplus 1 \oplus 1 \oplus 0 \oplus 1$ $= 0$
- $R_3 = B_5 \oplus B_6 \oplus B_7$ $= 0 \oplus 1 \oplus 1$ $= 0$
- $R_4 = B_9 \oplus B_{10} \oplus B_{11}$ $= 0 \oplus 0 \oplus 1$ $= 1$

	R_4	R_3	R_2	R_1	Info
0	0	0	0	0	Error-free
1	0	0	0	1	1. bit error
2	0	0	1	0	2. bit error
3	0	0	1	1	3. bit error
4	0	1	0	0	4. bit error
5	0	1	0	1	5. bit error
6	0	1	1	0	6. bit error
7	0	1	1	1	7. bit error
8	1	0	0	0	8. bit error
9	1	0	0	1	9. bit error
10	1	0	1	0	10. bit error
11	1	0	1	1	11. bit error

Hamming Code

- If we sent m bit data, the error occurs in 1,2,…,m bit