Transmission Errors

Gihan Dias
Sulochana Sooriarachchi
CS2032 – Principles of Computer Communication

Transmission Errors

"I heard from your **brother** in Finland. Plan to meet there tomorrow"

TRANSMISSION ERROR

"I heard from your **brothel** in Finland. Plan to meet there tomorrow"

Introduction

- **Error**: an alteration of one or more bits between transmission and reception
- **Error Detection**: know that an error has occurred
  - Discard the message, notify the sender, sender retransmits
  - Receiver corrects the error (**Error Correction**) and uses the message
- Types of errors:
  - Single bit errors – only one bit has changed
  - Error bursts – two or more bits, the first and last of which are in error

Error Detection Techniques:

Sending additional bits, value of which depends on the data

E.g. Parity check, Cyclic Redundancy Check (CRC)

Parity Check

- Count 1-bits in the data
- Parity bit:
- Odd parity => add a bit such that the total number of 1’s in the block of data becomes odd
- Even parity => add a bit such that the total number of 1’s in the block of data becomes even

This added bit is called the **Parity Bit**

Parity – Pros and Cons

- Can detect single bit errors
- E.g. A bit stream transmitted at a speed of 19.2 Kbps was interrupted by an electrical surge lasted for 0.01 sec. How many bits may get affected?
- Single bit errors are unlikely
- E.g. A block of bits sent over parallel cable and a single path get damaged
- Single bit errors!
- Parity is the basis for more sophisticated error detection techniques

Parity

- For double bit error detection
- Two parity bits; one for odd numbered bits (bit 1, 3, 5, 7…) and the other for even numbered bits (bit 2, 4, 6, 8…)
- Burst Error Detection
- Sending data bits in separate frames with each frame having a parity bit on its own: e.g. Data block sent as a two dimensional array

Cyclic Redundancy Check (CRC)

- Based on Polynomial Division
- Bit strings are interpreted as polynomials

$$b_{n-1}\ b_{n-2}\ b_{n-3} \dots b_2\ b_1\ b_0$$

$$\Downarrow$$

$$b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + b_{n-3}x^{n-3} + \cdots + b_2x^2 + b_1x + b_0$$

- Sender and receiver agree upon a **Generator Polynomial**

Steps

Generator Polynomial $G(x)$

1. Append number of zeros to the bit string, the number that is equal to the order of $G(x)$
2. Get the polynomial $B(x)$ for the resulting bit string in (1)
3. Divide $B(x)$ by $G(x)$ and get the remainder $R(x)$
4. Get $T(x)$ where $T(x) = B(x) - R(x)$
5. Send $T(x)$
6. Say $T'(x)$ is what the receiver gets
7. Receiver must divide $T'(x)$ by $G(x)$

- If remainder is 0 then the message is correctly received

Modulo 2 Arithmetic

- Modulo 2 Additions

  0 + 0 = 0

  1 + 0 = 1

  0 + 1 = 1

  1 + 1 = 0

- Modulo Subtraction

  0 – 0 = 0

  1 – 0 = 1

  0 – 1 = 1

  1 – 1 = 0

- Modulo 2 Division e.g.

									1	0	0	0	1	1	1
1	1	0	0	1	1	1	0	0	0	0	0	1	0	1	0
					1	1	0	0	1						
						0	0	0	1	0					
						0	0	0	0	0					
							0	0	1	0	0				
							0	0	0	0	0				
								0	1	0	0	1			
								0	0	0	0	0			
									1	0	0	1	0		
									1	1	0	0	1		
										1	0	1	1	1	
										1	1	0	0	1	

Example!!!!

- Send the Bit String **1101011**
- Generator Polynomial is $x^4 + x^3 + 1$

CRC Implementation

- Using Circular Shifts
- Needs only to Shift and XOR
- Steps

1. Interpret $G(x)$ as $b_r x^r + b_{r-1}x^{r-1} + \cdots + b_2 x^2 + b_1 x + b_0$
2. Shift register has ***r*** bits: rightmost position corresponds to b_0 and leftmost bit if for $b_{r-1}x^{r-1}$
3. An XOR circuit lies to the right of any position for which the corresponding b_i is 1. The leftmost bit position of the register is feed as one of the operands to XOR
4. Initialize the register to all Zeros
5. Input the bit string as one bit at a time from rightmost bit position of the register

Example!!!!

- Implement the Circular Shifting circuit for Generator Polynomial $x^5 + x^4 + x^2 + 1$
- Send the Bit String **1010001101**

initial	0	0		0
Step 1				
Step 2				
. . .				
Step n				

Error Correction

Detect, locate and correct erroneous bits

Hamming Code

- Hamming distance
  - E.g. v1 = 011011, v2=110001
  - The number of bits in which v1 and v2 do not agree is called the Hamming distance.
  - Then d(v1, v2) = 3
- Block code technique:
  - Map k-bit sequences into n-bit codewords
  - Invalid codeword correspond to minimum hamming distance

Block Code Example!!

Thank you