CSCI 234

Design of Internet Protocols:
Error Detection and Correction

George Blankenship

Outline

- Effect of Transmission Errors
- Cause of Transmission Errors
- Detection of Transmission Errors
- Correction of Transmission Errors

Effects of Errors

- Data may distorted (bit inversion)
- Data may be deleted (unrecognizable)
- Data may be added (merged messages)
- Data may be reordered (queuing delays)

Place of Errors (Layered Model)

- All errors are bit-based
- Bit insertion and distortion takes place at Physical Layer (PHY) – transmission errors
- Data Link (DL) Layer, provides well-defined service interface to the Network Layer and recovers from transmission errors
- Network Layer and above suffer from implementation (software/hardware) errors

Impact of Transmission Error

- Problem: some data sent from A → B may get 'corrupted'
  - Why is this a big problem?

How will the receiver know they received "bad data"?

- "bad data" is directly proportional to the probability of a transmission error

Error detection and Correction reduce the impact of a transmission error

Handling Transmission Errors

- Example:
  - "hello, world" is the data
  - "hzllo, world" received (detect? correct?)
  - "xello, world" received (detect? correct?)
  - "jello, world" received (detect? correct?)
  - what about similar analysis with "caterpillar"?
- Required: ***error detection***
- Helpful: ***error correction***

Typical Transmission Error Rates

Number of errors caused by data transmission is (typically) **orders of magnitude larger** than number of errors caused by hardware failures within computer systems

- Bit error probability (internal circuits) $< 10^{-15}$
- Average error probability (optical cables) $< 10^{-12}$
- Average error probability (coaxial cables) $< 10^{-6}$
- Average error probability (switched telephone lines) 10^{-4} - 10^{-5}
- Wireless channels are worse

Error Detection and Correction George Blankenship 7

Root Causes of Transmission Errors

- Not completely unpredictable or unaccountable
- Two main causes:
  - **Linear distortion of the original data**
    (caused by attenuation)
  - **Non-linear distortion**
    (caused by echoes, cross-talks, white and impulse noise)
- Errors usually occur in bundles: burst error

Error Detection and Correction George Blankenship 8

Error Measurement

- Several ways error characteristics can be expressed
  - Long-term average Bit Error Rate (BER)
  - Percentage of time BER does not exceed a given threshold value
  - Percentage of error-free seconds
- For the design of error control methods, the BER provides an indication of expected performance and the requirement for error control (detection and correction)

Error Detection and Correction George Blankenship 9

Error Detection & Correction

- Assume error rate of 0.1% (BER)
  - BER predicts that there will be one error for every 100 data items
  - Average sentence of text: 125 characters (125 x 8 bits)
  - One error every sentence of text
- EDC techniques
  - IF receiver detects error and (perhaps) requests sender to re-transmit (detection and retransmission)
  - Receiver detects and corrects error without re-transmission (detection and correction - forward error correction)

Error Control Objective

- Main requirement for error control methods is to increase the reliability of transmissions
- Error control cannot impact the BER
- Error control must impact the effect of the BER
- No error control method can reduce the impact of the BER to 0 or be expected to catch all errors that can possibly occur

The probability of an undetected error is non-zero

"The Weakest Link"

- If channel error rate already lower than that of peripheral equipment, any error control scheme would
  - Degrade performance
  - Decrease protocol reliability
- If the channel error is higher than that of peripheral equipment, any error control for messages between the peripherals would
  - Degrade performance
  - Decrease protocol reliability
- Concentrate on the area with the highest impact on system reliability

Appropriate Error Control

- An effective error control scheme should match the error characteristics of the channels
- Examples:
  - If a channel only produces insertion errors, a protocol protecting against deletions is useless
  - If a channel produces independent, single-bit errors with a relatively low probability, a simple parity scheme can surpass most sophisticated error control methods

Error Detection and Correction George Blankenship 13

Error Model

- P_b (BER): probability a bit is received inverted
- P_1: Probability a block of size F is received correctly

$$P_1 = (1- P_b)^F$$

  - (P_b=10^{-6}) P_1=.999 (1000 bit block), .99 (10000 bit block)
- P_2: Probability that a PDU will be received with undetected error

$$P_2 = 1- P_1$$

  - (P_2=10^{-3}) 1 of 1000 has undetected error

Error Detection and Correction George Blankenship 14

Transmission Error Detection

- Most important transmission errors show up as data:
  - Insertion
  - Distortion
- Methods to verify data consistency
  - Duplication (voting)
  - Seal (signature)
  - Reordering (predictable pattern)

Error Detection and Correction George Blankenship 15

Error Handling Strategies

- Error-correcting
  - Redundant information packaged with data
  - Receiver able to detect that a transmitted unit is corrupted
  - Receiver is able to correct corrupted data
  - Probability of false positives and negatives
- Error-detecting
  - Redundant information packaged with data
  - Receiver is able to detect a corruption has occurred
  - Location of corruption is not possible
  - Receiver may request a retransmission
  - Low probability of false positive or negative

Residual Error Rate

- Not all errors can be detected
- Residual Error Rate (RER) always exists
  - p is a transmission error probability
  - f is a fraction of errors caught by error control
  - RER= $p\ (1 - f)$

Code Types (Error Protection)

- **Code rate**
  - Code rate decreases (redundancy increases) -> efficiency increases
- **Two basic types of codes**
  - **Block codes**
    - all frames (codewords) have same length
    - encoding for each possible data message can be statically defined
  - **Convolution codes**
    - codewords depend on data message and a number of previously encoded messages
    - length of the codewords is usually constant

Code Type Classification

- **Linear Codes**
  - every linear combination of codewords produces another valid codeword
- **Cyclic Codes**
  - every cyclic shift of a valid codeword produces a valid codeword
- **Non-Systematic Codes**
  - Add redundancy and transform the coded message such that no part of original message recognizable from the un-decoded message.
- **Systematic Codes**
  - Message data not disturbed: redundant symbols are added separately to each block.

Error Detection and Correction George Blankenship 19

Data Transmission

- Data link breaks physical layer stream of bits into *frames*

  `...010110100101001101010010...`
- How does receiver detect boundaries?
  - Length count
  - Special characters
  - Bit stuffing
  - Special encoding

Error Detection and Correction George Blankenship 20

Encoded Frames

- Frame has *m* data bits, *r* redundancy bits
  - n = (m+r) bit *codeword*
- Given two codewords, compute distance:
  - `10001001`
  - `10110001`
  - XOR, 3 bits difference
  - *Hamming Distance*
- "So what?"

✓Small packets: low packet error rate, high packetization overhead
✓Large packets: high packet error rate, low overhead
✓Efficiency depends on BER and energy consumption per transmitted bit

Error Detection and Correction George Blankenship 21

Frame Processing

- PHY provides this service to DL
- PHY accepts bit stream and attempts to deliver to destination
  - (Bit stream not guaranteed to be error free)
- DL usual approach is
  - break the bit stream up into discrete frames
  - compute the checksum for each frame
  - When frame arrives at destination, checksum recomputed
  - If checksum is different, DL knows an error has occurred (must be dealt with)

Error Protection Algorithms

- Data redundancy for error control
- Error detection
  - Parity
  - Checksum
  - CRC
- Error detection and correction
  - Hamming

Error Detection: Parity

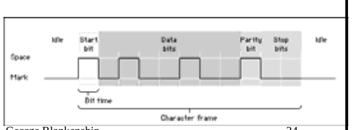
Error detection:
some information is added to message, that allows checking for errors.

PARITY

1 ASCII char = 7 bits;
1 extra bit is added to each character, called parity bit:

- Even parity: value of parity bit is set to make total number of 1's even.
- Odd parity: value of parity bit is set to make total number of 1's odd.

Example:
(ASCII) W: 1010 111
(**odd parity**) W: **0** 1010 111
(**even parity**) W: **1** 1010 111



Parity Example

- The parity of the bitstream 10111101 is even
  - The parity bit will be 0.
- The parity of the bitstream 01110011 is odd
  - The parity bit will be 1.
- The parity of the bitstream 00000000 is even
  - The parity bit will be 0.

Error Detection and Correction George Blankenship 25

Extended Parity check

- Easily extended from:
  - single-error-detecting code to single error-correcting code
- Conditions:
  - Code rate of EC code generally lower than ED code (additional bits)
  - Codes considered to be useful only when communication of control messages is difficult
- Example:
  - long transmission delay
  - absence of a return channel
  - high bit-error rate

Error Detection and Correction George Blankenship 26

Error Detection: Checksums

Let: message = “Hello, world” == 12 bytes == 128 bits (including parity bit)

Hello, world *assuming: parity bit = 0

72	101	108	108	111	44	32	119	111	114	108	100

Sum = 72 + 101 + 108 + ... + 100 = 1128

Checksum [16-bits] = 1128 **Checksum [8-bits]** = 1128 mod 256 = 104

Message:

72	101	108	108	111	44	32	119	111	114	108	100	104

1 checksum/12 bytes ➔ too much overhead
typical use: 1 checksum byte per **128 Bytes** of data

Error Detection and Correction George Blankenship 27

Error Detection: Cyclic Redundancy Check (CRC)

Number: 629 | Divisor: 25 ← <u>Pre-agreed</u> between sender/receive

mod(Number, Divisor): mod(629, 25) = 4

⇩

Transmit: (629,4)

⇩

Receiver: mod(629, 25) == 4 ? ➔ Transmission probably OK

Error Detection: Standard CRC

Implementation of CRC:

Number: Bitstream of 1 Block (e.g. 128 Bytes)

Divisor: common CRC schemes are:

CRC12	1100000001011
CRC16	11000000000000101
CRC-CCITT	10001000000100001
CRC32	100000100110000010001110110110111

M = mod(Number, Divisor): computed very efficiently with simple circuits

FRAME: Number MOD

⇩

Receiver: mod(Number MOD, CRC) = = M ? **Yes ➔ Transmission probably OK**

Hamming Distance

- Difference between two codewords is the number of bits in which they differ
- Hamming Distance of a code
  - minimum difference between two words in a code (XOR operation)
- Hamming Distance determines
  - code's error detection and correction abilities

Error Correction: Hamming

- Add code words to the data
- Code words identify that data has not been modified with a reasonable probability of undetected error
- Code words can be used to identify probable location of modification with an significant probability of an undetected error

Error Detection and Correction George Blankenship 31

Hamming Distance

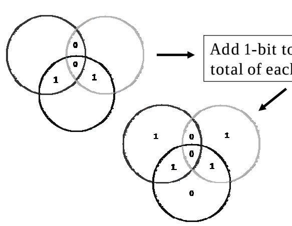
- Hamming distance of *n*
- Any combination of up to *n*-1 bit errors per codeword can be detected
- Any combination of up to (*n*-1)/2 errors per codeword can be corrected

Error Detection and Correction George Blankenship 32

Hamming Code

Hamming code: main idea of a (7, 4) Hamming code

Data: 0011

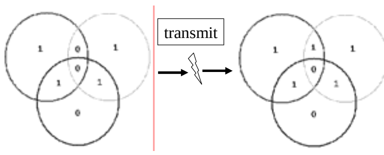


Add 1-bit to each circle, total of each circle ➔ even

Error Detection and Correction George Blankenship 33

Hamming code

Data: 0011 →



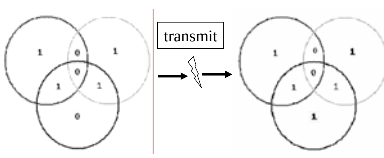
→ receiver

- red-circle: parity ERROR
- green circle: parity ERROR
- blue circle: parity OK

→ Error must be in bit shared by red and green

Hamming code

Data: 0011 →



→ receiver

- red-circle: parity OK
- green circle: parity OK
- blue circle: parity ERROR

→ Error must be in unshared bit of Blue circle

Hamming code…

Data: bit-stream → write data in particular sequence → compute, add hamming code bits to data → transmit → receiver checks Hamming codes, corrects errors → Corrected data

Common Hamming code used: (12, 8) Hamming code

Extending Hamming code for longer bit-stream

1. All bit positions that are powers of two are used as parity bits.
   – Positions 1, 2, 4, 8, 16, 32, 64, etc.

2. All other bit positions are for the data to be encoded.
   – Positions 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc.

3. Each parity bit stores the parity for assigned bits in the code word:
   – The position of the parity bit determines the sequence of bits that it alternately checks and skips

   – Position 1: skip 0 bit, check 1 bit, skip 1 bit, check 1 bit, skip 1 bit, …
   – Position 2: skip 1 bit, check 2 bits, skip 2 bits, check 2 bits, skip 2 bits, …
   – Position 4: skip 3 bits, check 4 bits, skip 4 bits, check 4 bits, skip 4 bits, …
   – Position 8: skip 7 bits, check 8 bits, skip 8 bits, check 8 bits, skip 8 bits, …
   – Position 16: skip 15 bits, check 16 bits, skip 16 bits, check 16 bits, skip 16 bits, …
   – Position 32: skip 31 bits, check 32 bits, skip 32 bits, check 32 bits, skip 32 bits, …

Burst errors

- Problem with Hamming:
  – burst errors: several contiguous data bits in error
- Handling burst errors:
  – Interleaving
  – Reed-Solomon coding

- Examples of usage: storage media (CDROM’s), …

Message Authentication

- Purpose
  – authenticate source
  – authenticate content
- Method
  – encrypt message
  – message seal

Hash Function

- A *hash function H* is a transformation that takes an input *m* and returns a fixed-size string, which is called the hash value *h* (that is, *h* = *H*(*m*)).
- It is desirable for the hash value to be unique for each input, but relative size of *m* and *h* usually prevent uniqueness

Error Detection and Correction George Blankenship 40

Message Hash

- Message hash is a seal of message, change to message would generate new hash value
- Hash is appended to message to allow recipient to validate content
- Hash is encrypted to prevent re-computation of hash after modification of message
- Encryption key is signature of author

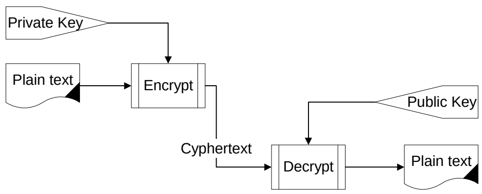
Error Detection and Correction George Blankenship 41

Public Key Encryption

- Central problem of encryption is key management
- It is impossible to distribute a random key for each possible dialog worldwide using symmetric key encryption
- Asymmetric key encryption allows each source to have a random key

Error Detection and Correction George Blankenship 42

Asymmetric Encryption



Digital Signatures

- Message hash combined with public key encryption yields a method for an individual to "sign" a message
  - Protects content from modification
  - Authenticates author
  - Author cannot repudiate message
- Certificate authority controls the release of the public keys