ITU
Faculty of Computer and Informatics
Prof. Dr. F. Sema OKTUĞ
Ress. Asst. G. Selda UYANIK

24.10.2013

BLG 433E: COMPUTER COMMUNICATIONS

PROJECT – I

In this project, you will simulate hamming code single-error correcting (SEC) capability for two different types of errors: (burst and normal) in different environments and study their performances.

Please go through how you can code 8 bit data to 12 bit codewords using Hamming code for SEC capability.

Coding Example:

Original data (m=8 bits) : 10100010
Codeword after Hamming Code (SEC), even parity : **0**1**1**10**1**0**1**0010 → frame sent by sender

Note: You will send frames one by one regardless of error type: will apply coding on read 8-bit data and send 12-bit data to environment.

➢ Introductory Information on How to Implement Environmental Errors:

Environmental errors may not always take place uniformly randomly and smoothly, rather, errors may come in the form of bursts. Below is an abstract description for your implementation:

burstSize = 1 → Uniform Type Error:

For each bit of frame sent by sender, draw a random number *rnd* between [0 - 1]:

$$\begin{cases} rnd < errorRate & invert\ bit \\ otherwise & no\ change \end{cases}$$

burstSize = b → Burst Type Error:

For each frame sent by sender, draw a random number *rnd* between [0 - 1]:

rnd < *errorRate* → Select random bit position for start of the burst (constraint: remaining bits should be at least of size b). Invert *b* bits afterwards.

rnd ≥ *errorRate* → No change on frame

Note that this is a simple implementation of burst type errors; indeed, a burst error frame may not have all burst bits (b bits) erroneous, indeed what ensured is actually at least the first and last bits of b sized frame have error.

➢ **Program:** Write a C++ program to implement the described behaviors below:

- *inputFile* (.txt file), *errorRate* and *burstSize* are command line parameters
- A traffic generator -sender- : Reads original bit frames from each line of file: *inputFile*

  (m = 8 bits: each line in the input file corresponds to a legal message of 8 bits. There are 100 lines.)

- For modeling environmental errors, see above description.
- Program simulates transmission of the data read from file (*inputFile*) with an environment having given *errorRate* and *burstSize* parameters. Therefore, what you need to implement in your program is basically: sender/receiver behaviors for coding/decoding (error detecting & correcting), environment behavior (error implementations on data sent by sender) and performance analysis functionalities.
- Performance metric to be analyzed:

error detection : The ratio of the detected errors over all bit errors.

error correction : The ratio of the corrected errors over all bit errors.

error miss-correction rate: The ratio of the miss-corrected errors over all bit errors.

(i.e, cases where too many bit errors has turned sent codeword into another legal codeword and sent data is recovered as another legal message instead of actual data message sent)

(Note: You may calculate these rates with a post processing function that takes inputs: actual data bits sent by sender, received data bits by receiver (may be erroneous) and data bits decoded by receiver -what receiver decodes as data after error detection & correction-)

➢ Runs:

- Run your program with below parameters and keep corresponding *performance metrics.*

errorRate = 0.06, 0.08, 0.12, 0.18, 0.26

and for each error rate:

burstSize = 1, 2, 3, 4

➢ Report:

- Answer below questions:
  - Discuss what can be an advantage and a drawback for having burst errors?
  - Discuss how we can use this code wisely for the case of burst errors.
  - What is the *code rate* for described hamming error detection mechanism?
  - What is *hamming distance of the complete code* (legal codeword set) you will use in this project?
  - In general, why you need distance 2d+1 code to correct d errors?
- Include *error detection/correction/undetection rate* vs *errorRate* graphics having 4 lines plotted corresponding to different *burstSizes* (1,2,3,4) and discuss their behavior.
- Program Related Explanations:
  - Give *brief* but *sufficient* information of structures and parameters you design and use in your program.
  - Include compiling, running options and a sample run.
  - Include a section briefly describing sender/receiver behaviors and error correction algorithm specially modification/s for both error types.

Due to: 7 November 2013, Thursday, 23:00

Submission NOTES:

- Submissions are through Ninova system and has a **strict deadline**, no assignments submitted after deadline is accepted.
- Source codes and the report file you prepared should be uploaded as one compressed file named after your student number as "*student_number*.zip"
- You should implement your program in C++ programming language.
- Be sure that your program works and produces reasonable and expected results.
- This is *not* a group assignment and getting involved in any kind of cheating induces negative grades.