Name: Joaquim Miguel Conceição Espada

Login: con0004

Date: 4th October of 2016

Theoretical Report

Title: Error Correction

Task: Learn and understand the methods used to correct errors in data transmission

Scheme:

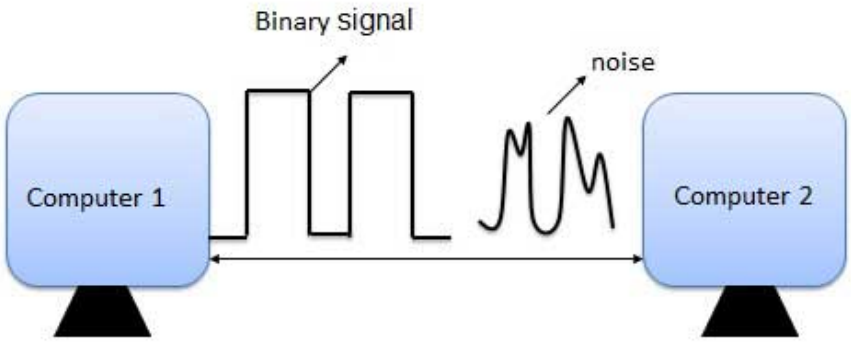

Illustration 1: Data transmission with noise wave

Introduction to problematics:

When data is transmitted it can scrambled by a channel noise and so errors maybe introduced during the transmission from the sender and receiver.

To avoid this situations there are used some techniques and methods that allow correcting errors and the main idea is the following the sender encodes the message in a redundant way using error-correction code (ECC). There are several methods that follow that main idea as Automatic Repeat Request(ARQ), Repetition Code and Hamming Code.

In the final part of class a simple implementation of Hamming Code was tested in Octave.

Elaboration:

Initial Concepts

Minimum Hamming Distance

The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.

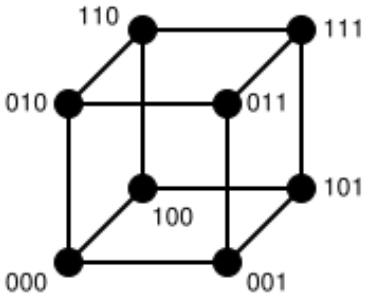

Illustration 3: 3-bit binary cube for finding Hamming distance

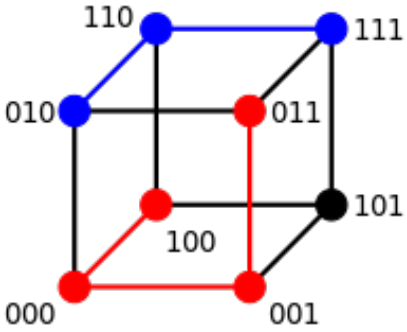

Illustration 2: Two example distances: 100→011 has distance 3 (red path); 010→111 has distance 2 (blue path)

$$Error\ Correction = \frac{d_{min} - 1}{2}$$

- $d_{min} = 1$ → Different in only 1 bit, no error correction neither error detection
- $d_{min} = 2$ → 1 bit error detection and no error correction
- $d_{min} = 3$ → 2 bit error detection and error correction

Signal to Noise Ratio

SNR is a measure that compare the level of background noise with a level of desired noise. It is often expressed in decibels (dB).

It is defined as the ratio of the power of a signal and the power of noise.

$$SNR = \frac{P_{signal}}{P_{noise}}$$

Automatic Repeat reQuest

This method uses acknowledgements (messages sent by the receiver indicating that it has correctly) received a data frame or packet (and timeouts specified periods of time allowed to elapse before an acknowledgment is to be received) to achieve reliable data transmission over an unreliable channel.

If the sender doesn't receive the acknowledgement message before the timeouts, he will re-transmit the data until the acknowledgement message from the receiver is received. The sender will do this process until the sender exceeds the predefined number of re-transmissions.

There are three different protocols in ARQ :

- Stop and wait
- Go Back N
- Selective Repeat

Stop and wait

It is the simplest protocol of automatic repeat-request (ARQ) method. In this technique the sender sends one frame/packet at a time. After sending each frame/packet the sender waits and doesn't send any further frames until he receives the ACK signal from the receiver.

When the receiver doesn't send a frame in an expected interval of time the sender re-transmits the same packet. Theses intervals of time is know as timeouts. This process is repeated until there are no more frames to be sent. Stop-and-wait ARQ is inefficient compared to other ARQs, because the time between packets.

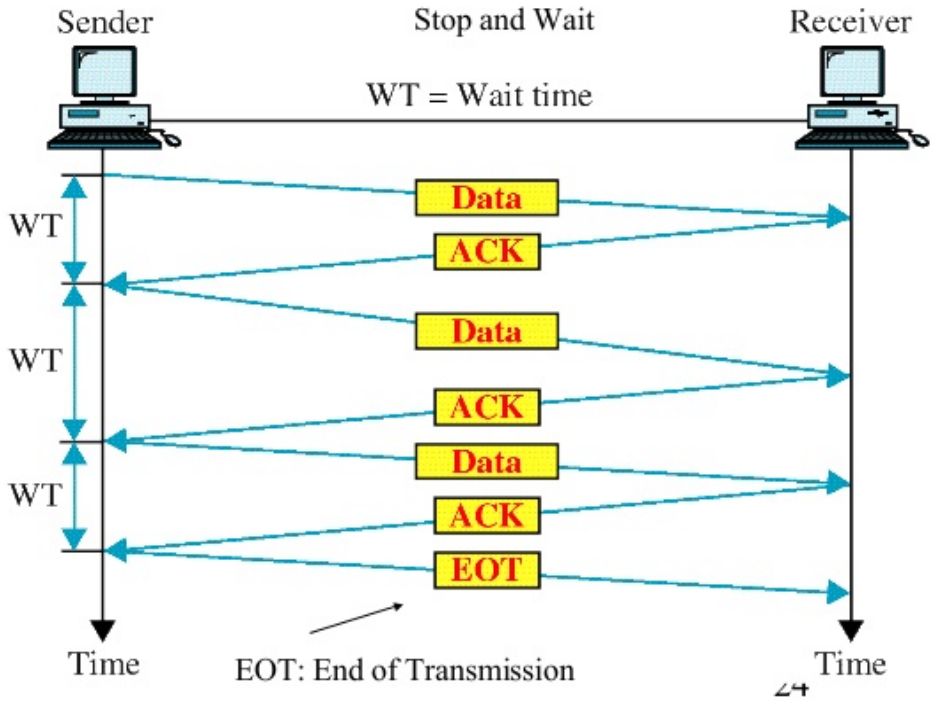

Illustration 4: Stop and Wait ARQ Scheme

Go Back N

In this protocol the sender keeps sending the number of frames/packet specified by the window size even without receiving the ACK packet from the receiver.

The receiver keeps track of the sequence number of the next frame that expects to receive, and sends that number with every ACK that it sends. The receiver discards any frame that doesn't have the exact expected sequence then it resend an ACK with the right expected number of sequence frame.

When the sender has sent all of the frames in its window, it will spot that all of frames since the first lost frame were discarded by the receiver and will go back to the sequence number of the last ACK it received and process and fill its window starting with that frame and continue the process over again.

This protocol is more efficient than Stop and wait because in time that Stop-and-wait consumes waiting for the packet Go Back N is sending packets.

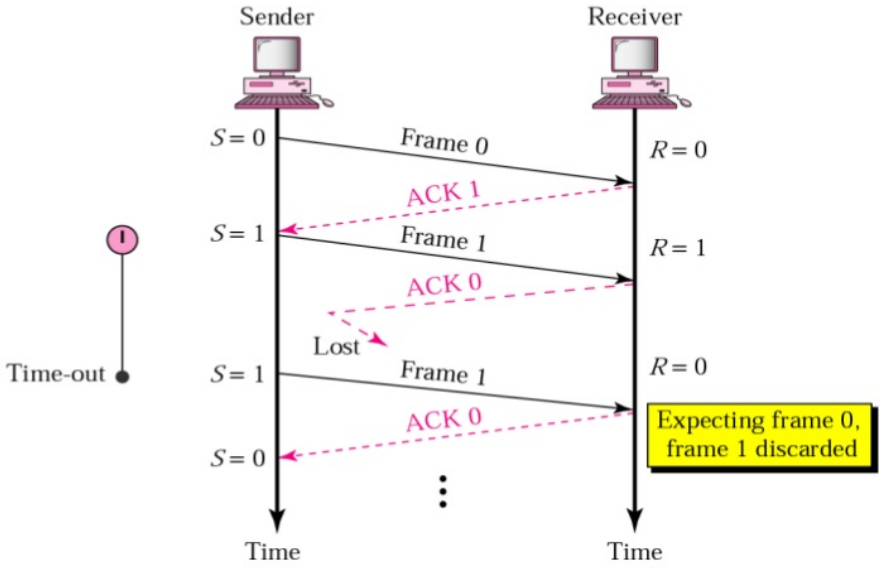

Illustration 5: Lost ACK frame

Selective Repeat

In this protocol the sender continues to send a number of frames/packets designated in the window size even after a frame loss.
The receiver will continue to accept and acknowledge frames even after the first error. The window size must be greater than one.

The receiver keeps track of these sequence number of the first non received frame, and sends that number with every ACK that sends. If a frame from the sender doesn't reach the receiver, the sender continues to send the following frames until the windows is empty, at the same time the receiver continues to fill up is windows, and in each reply we will send the sequence number of the first missing

frame.
Once the sender emptied is windows it will start to resend the frame number given by ACK and then continues where it left off.

Repetition Code

The basis of this algorithm is to repeat the message several times hoping that the channel only corrupts a minority of these repetitions. The receiver will know if an error occurred since the received data isnt a repetition of a single message. Also the receiver can recover the original message by looking at the received message in data stream that occurred the majority of times.
Code parameters

- Minimum Hamming Distance → r
- $Error\,Correction\,Capacity = \frac{r-2}{2}$

Hamming Codes

The Hamming Codes are a binary linear code that encodes *k* bits of data into a *n* bit codeword adding *m* check bits.

Hamming codes can detect up to two-bit errors or correct one-bit errors without detection of uncorrected error. Hamming codes are called perfect codes because they achieve the highest possible rate for codes with their block length and minimum distance of three .

Important Formulas

- $r \geq 2$ → initial condition to use hamming code the minimum hamming distance must be three
- $n = 2^r - 1$ → block length
- $k = 2^r - r - 1$ → Information length
- $R = {}^{k}\!/_{n} = \frac{1-r}{2^r-1}$ →Highest possible for codes with minimum hamming distance of three

Parity Check Matrix (*H*)

The parity-check matrix of a Hamming code is constructed by listing all columns of length r that are non-zero.

Generator Matrix (*G*)

The Generator matrix can be obtained from the Parity Check Matrix by taking the transpose of the left hand side of *H* with the identity k-identity matrix on the left hand side of *G*.

Encoding

When the sender is encoding the message it must be multiplied by *G* the result is the encoded message.

Error Detection

When the receiver receives the message multiplies *H* by the message to obtain the *syndrome* vector *z.* If the *syndrome* vector as in all positions '0', the receiver can conclude that no errors have occurred.

Error Correction

The receiver must multiply *H* by the received message and obtain the *syndrome vector* z. Then the receiver must check the value of the syndrome vector and see the correspondent number column in matrix *H.* The last step is to flip or negate is value.

Decoding

Once the message is received with no error or error was corrected the least *k* significant bits of the code word are the encoded message.

A few Hamming Code Combinations		
k	***r***	***n***
Information part	**Parity Bits**	**Block length**
1	2	3
4	3	7
11	4	15
26	5	31
57	6	63
120	7	127
247	8	255

Hamming Code (7,4)

The Hamming Code (7,4) is an implementation of the Hamming Code that encodes four bits of data into seven by adding three by adding three parity bits.

The code generator matrix and the parity-check matrix are:

$$G = \begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}$$

Drawing 2: Generator Matrix

$$H = \begin{array}{cccc|ccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}$$

Drawing 1: Parity Check Matrix in Standart form

Message = 0 0 0 1
Encoding of the message = $0001 \times G = 0001111$

Hamming Code (7,4) – Checking and correcting errors

m = 0 0 0 1 1 1 1 → received message with no error

m' = 1 0 0 1 1 1 1 → received message with one error

No error scenario

$$S = \begin{array}{cccc|ccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} * \begin{array}{c} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{array} = \begin{array}{c} 0 \\ 0 \\ 0 \end{array}$$

Error scenario

$$S = \begin{array}{cccc|ccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array} * \begin{array}{c} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{array} = \begin{array}{c} 1 \\ 0 \\ 0 \end{array}$$

An error has been detected because the result isn't a null vector. When can see that the value of the *syndrome vector* equals the first column of the parity matrix indicating that the error is in the first bit of the the m'. Now we need to flip or negate the bit and error is corrected.

Octave Implementation of the non-error scenario

```
octave:2> G=[1 0 0 0 0 1 1 ; 0 1 0 0 1 0 1 ; 0 0 1 0 1 1 0 ; 0 0 0 1 1 1 1] # Declaration of the
Generating matix
G =

  1  0  0  0  0  1  1
  0  1  0  0  1  0  1
```

```
  0   0   1   0   1   1   0
  0   0   0   1   1   1   1


octave:3> C = [0 0 0 1] * G #encoded message
C =

  0   0   0   1   1   1   1


octave:4> H = [1 0 1 0 1 0 1 ; 0 1 1 0 0 1 1 ; 0 0 0 1 1 1 1] # Declaration of the parity check matrix

H =

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

octave:5> C' #transpose matrix of C
ans =

  0
  0
  0
  1
  1
  1
  1

octave:7> S =mod(H*C',2) #Obtaining syndrome vector
S =

  0
  0
  0
```

Conclusion:

After the detailed analysis of the following techniques: ARQ and it's variations, Repetition Code and Hamming I concluded that the simplest was the Repetition Code and Hamming Code with the Hamming distance of three was the best one.
Regarding the ARQ protocols the more efficient of all three is the Selective Repeat because the sender continues to send the following frames until the windows is empty, at the same time the receiver continues to fill up is windows, and in each reply we will send the sequence number of the first missing frame.
In Stop and wait protocol there are lot of lost time because the sender needs to receive an ACK/NACK to send the next frame or retransmit the lost one.
The Go back to N the sender send continuously the packets to the receiver while the receiver keeps

track of the sequence number of the next frame that expects to receive. When the sender windows is empty and were lost frames the sender will start retransmitting since the sequence number of the first lost frame.

References:

- Notes from Ing. Pavel Nevlud classes
- https://en.wikipedia.org/wiki/Automatic_repeat_request
- https://en.wikipedia.org/wiki/Stop-and-wait_ARQ
- https://en.wikipedia.org/wiki/Selective_Repeat_ARQ
- https://en.wikipedia.org/wiki/Repetition_code
- https://en.wikipedia.org/wiki/Hamming_distance#Error_detection_and_error_correction
- https://en.wikipedia.org/wiki/Hamming_code