Implementation of longitudinal redundancy check and cyclic redundancy check algorithms using NetSim

RAKESH S

M.tech, 2nd year, CS&E dept.
SSIT, Tumkur (D), Karnataka (S), India

rakeshs.snb@gmail.com

SOWMYA M N,

Lecturer of CS&E dept,
SSIT, Tumkur,
Karnataka, India.

sowmyamn22@rediffmail.com

Dr.M.SIDDAPPA,

Professor & HOD of CS&E dept,
SSIT, Tumkur,
Karnataka, India.

siddappa.p@gmail.com

***Abstract*-- Network Simulation is a comprehensive tool for studying computer networks. NetSim features state of the art network simulation technology, and comes with additional offerings that enhance and quicken learning. NetSim is an educational network simulator and provides educators with a comprehensive and effective means of teaching computer networks. The software provides for network simulation across various protocols like Ethernet, Wireless LAN. TCP / IP, ATM and devices like routers, ATM switches. Apart from simulation NetSim also features C / C++ /JAVA programming exercises, animated basics and real time packet capture. NetSim provides network performance metrics at various abstraction level such as Network, sub-network, Node and a detailed packet trace. NetSim features a Development environment with a source code editor and a compiler. Model libraries with source code are provided for user modification. Importantly, options are not limited to the listing as it is possible to develop any type of protocol or device model with NetSim's protocol editing facilities. Error detection [2] is a technique that enables reliable delivery of digital data over unreliable communication channels. Many communication channels are subject to channel noise, and thus errors may be introduced during transmission from the source to a receiver. Error detection techniques allow detecting such errors, while error correction enables reconstruction of the original data. Error detection is the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver.**

Key words:

AWT: Abstract Window Toolkit
CRC: Cyclic Redundancy Check
LRC: Longitudinal Redundancy Check
NetSim: Network Simulation

I. INTRODUCTION

A. Network Simulation

NetSim is Network simulation software which takes in user inputs and provides output metrics. NetSim is a popular tool developed by TETCOS, in association with Indian Institute of Science, Bangalore. NetSim has also been featured with Computer Networks and Internets V edition by Dr. Douglas Comer, published by Prentice Hall. NetSim provides network performance metrics at various abstraction levels such as Network, sub-network, Node and a detailed packet trace.

NetSim features a Development environment with a source code editor and a compiler. Model libraries with source code are provided for user modification. Importantly, options are not limited to the listing as it is possible to develop any type of protocol or device model with NetSim's protocol editing facilities.

B. Java Swing class hierarchy

Swing, which is an extension library to the AWT, includes new and improved components that enhance the look and functionality of GUIs. Swing can be used to build Standalone swing GUI Apps as well as Servlets and Applets. It employs model/view design architecture. Swing being 100% Java code makes it more portable and more flexible than AWT. Swing is built on top of AWT. Swing is written entirely in Java, using AWT's lightweight component support. In particular, unlike AWT, the architecture of Swing components makes it easy to customize both their appearance and behavior. Components from AWT and Swing can be mixed, allowing you to add Swing support to existing AWT-based programs. For example, swing components such as JSlider, JButton and JCheckbox could be used in the same program with standard AWT labels, text fields and scrollbars. You could subclass the existing Swing UI, model, or change listener classes without having to reinvent the entire implementation. Swing also has the ability to replace these objects on the fly.

The class JComponent, descended directly from Container, is the root class for most of. Swing components i.e. used to build a GUI. The list below shows some of the commonly used Swing components that are used in Java swing programs. To learn and understand these swing programs, AWT programming knowledge is not required.

LRC [6] is a form of redundancy check that is applied independently to each of a parallel group of bit streams. The data must be divided into transmission blocks, to which the additional check data is added.

CRC [1] [6] is a single-burst-error-detecting cyclic code and non-secure hash function designed to detect accidental changes to digital data in computer networks. It is characterized by specification of a so-called *generator polynomial [4]*, which is used as the divisor in a polynomial long division over a finite field, taking the input data as the dividend, and where the remainder becomes the result.

II. PROCEDURE

A. Steps for doing LRC:

- A block of bits is organized in a table (rows and columns).
- For example instead of sending 32 bits, we organize them in a table made of 4 rows and 8 columns.
- We then calculate the Parity bit for each column and create a new row of 8 bits which are the parity bits for the whole block.
- Note that the first parity bit in the 5th row is calculated based on all the first bits.
- The second parity bit is calculated based on all the second bits and so on..
- We then attach the 8 parity bits to the original data and send them to the receiver.

B. Steps for doing CRC:

Steps performed by Sender:

- Get the raw frame.
- Left shift the raw frame by n bits and divide it by divisor.
- The remainder is the CRC bit.
- Append the CRC bit to the frame and transmit.

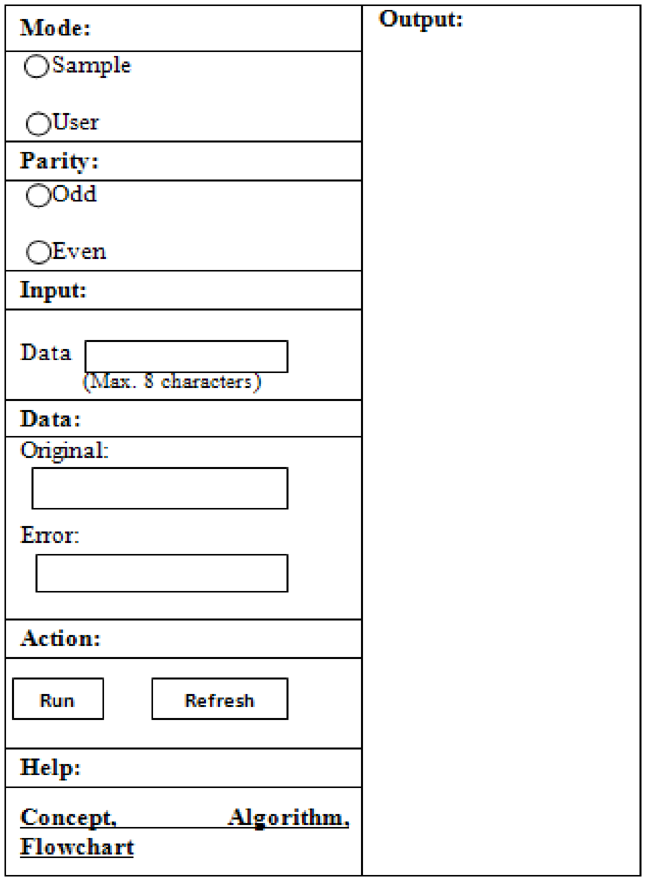
Steps performed by Receiver:

- Receive the frame.
- Divide it by divisor.
- Check the reminder.

III. IMPLEMENTATION

A. Concept of LRC:

- LRC is used for detecting the error in the transmitted data bits.
- The term Redundancy here means the extra information appended to the transmitted data bits. This is mainly appended at the end of each data frame transmitted. These redundancy bits are used to check the accuracy of the received data frame [6].
- The input data is converted to Bit Format and the bits are divided into columns of 8 bits and rows are required (a data frame of 32-bits is converted into a table of 8 columns and 4 rows).



(Note: The front end design of LRC is changed due to space)

- Calculate the parity bits of each of the 8 columns and create a new row of 8 bits, which will have the parity bits for the whole block [3].

- The first parity bit values of the first column, second parity bit value based on second column values and so on…..
- After calculation of the 8 parity values it is attached to the original data and sends across to the receiver.

B. Front end design of LRC:

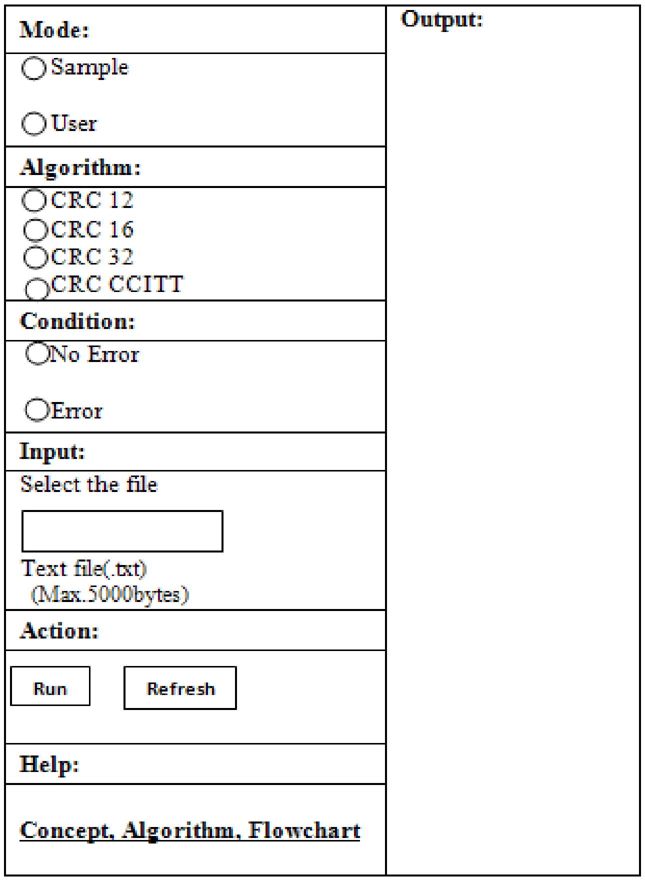
- For designing the front end of the LRC we use **JPanel**.
- We use **JLabel** component for creating the labels like Mode, Parity, Input, Data, Action, Help, Sample, User, Odd, Even, Data, Max. 8 characters, Original, Error and Concept, Algorithm, Flowchart.
- In Mode section **JRadioButton** component is used for selecting Sample mode or User mode and for selecting Odd or Even in Parity section.
- In Input section **JTextField** component is used to enter the digits and it is fixed to 8 characters and in Data section it is used to display original data and if any error occur during running it is also displayed.
- In Action section **JButton** component is used for selecting RUN or REFRESH action.
- In Help section a hyper link is creating, when click on this link it will display the information like Concept, Algorithm and Flowchart of the CRC.

C. Concept of CRC:

- CRC is a type of function that takes as input a data stream of any length and produces an output [7].
- Bit strings are treated as representation of polynomial with co-efficient of ‘0’ and ‘1’.
- The sender and receiver must agree upon the Generator polynomial in advance.
- The size of data must be greater than the size of the Generator polynomial to compute the checksum [7].
- The computed checksum is appended to the transmitting frame.
- If the receiver gets the frame it tries dividing it by generator polynomial. If there is a remainder there has been a transmission error else no error [8].
- The Generator polynomial for CRC-12 is given below,
- X12+X11+X3+X2+X1+1, the binary equivalent of CRC-12 is 1100000001111.

D. Front end design of CRC:

- For designing the front end of the CRC we use **JPanel**.
- We use **JLabel** component for creating the labels like Mode, Algorithm, Condition, Input, Action, Help, Output, Sample, User, CRC-12, CRC-16, CRC-32, CRC-CCITT, Select the file, Text file(.txt), Max. 5000 bytes and Concept, Algorithm, Flowchart.
- In Mode section **JRadioButton** component is used for selecting Sample mode or User mode and for selecting CRC-12, CRC-16, CRC-31 and CRC-CCITT mode in Algorithm section and for selecting No Error and Error mode in Condition section.
- In Input section **JTextField** component is used for selecting input file.
- In Input section **JButton** component is used for browsing the input file in the system and in Action section for selecting RUN or REFRESH action.
- In Help section an Hyper link is create, when click on this link it will display the information like Concept, Algorithm and Flowchart of the CRC.



(Note: The front end design of CRC is changed due to space)

IV. PSEUDO CODE

A. Pseudo code of LRC:

- Read the content of the input file, i.e. the input data, type of parity and the error data from the Input.txt and store it in variable like the above method.
- After finding out the length of the data converted into bits form a table of 8 columns and depending on the data length the number of rows.

- If the passed flag (type of parity) is 2 then it is even parity else it is odd parity.
- Calculate the parity bit of each of the 8 columns and create a new row of 8 bits which with have parity bits for the whole block.
- The first parity bit of the first column is calculated based on all the first bit values of the first column, second parity bit value based on second column value and so on…..
- After the parity bit is formed, i.e. the last row has been formed which represents the parity bit of the original data and error data, they are written into the output file in each line.

B. Pseudo code of CRC:

- Convert the data into binary format.
- If it is sender side append 12 0’s with the binary data else add original CRC with the binary data.
- If the MSB is 0 then XOR binary data with the string 000000000000 else XOR with the string 1100000001111.
- Find the HEXA decimal format of the XOR results.
- If it is 0 both sender and receiver matches then there are no error else there is an error.

V. CONCLUSION

This paper gives the details about error detection mechanisms such as LRC and CRC and how these mechanisms can work on NetSim using front end designs of LRC and CRC.

VI. ACKNOWLEDGMENT

I am very thank full to my HOD Dr. M. Siddappa, SSIT, and my internal guide Sowmya M N, SSIT and also my external guide Mahendran, TETCOS, Bengaluru, for their guidance in my project work and also their guidance for preparing this paper.

REFERENCE

[1]. Ritter, Terry. "The Great CRC Mystery". *Dr. Dobb's Journal* **11** (2): 26–34, 76–83. http://www.ciphersbyritter.com/ARTS/CRCMYST.HTM. Retrieved 21 May 2009.

[2]. Peterson, W. W. and Brown, D. T.. "Cyclic Codes for Error Detection". *Proceedings of the IRE* **49**: 228.

[3]. N. Cam-Winget, Nancy; R. Housley, Russ; D. Wagner, David; J. Walker, Jesse (May 2003). "Security Flaws in 802.11 Data Link Protocols". *Communications of the ACM* **46** (5): 35–39.

[4]. Stigge, Martin; Plötz, Henryk; Müller, Wolf; Redlich, Jens-Peter (May 2006). *Reversing CRC – Theory and Practice*. Berlin: Humboldt University Berlin. pp. 24. http://sar.informatik.hu-berlin.de/research/publications/SAR-PR-2006-05/SAR-PR-2006-05_.pdf. Retrieved 21 July 2009.

[5]. Anachriz (30 April 1999). "CRC and LRC how to Reverse it". http://www.woodmann.com/fravia/crctut1.htm. Retrieved 21 January 2010.

[6]. Williams, Ross N. (24 September 1996). "A Painless Guide to LRC and CRC Error Detection Algorithms V3.00". http://www.repairfaq.org/filipg/LINK/F_crc_v3.html. Retrieved 5 June 2010.

[7]. Koopman, Philip; Chakravarty, Tridib (2004). "Cyclic Redundancy Code (CRC) and Longitudinal Redundancy Code(LRC) Polynomial Selection For Embedded Networks". http://www.ece.cmu.edu/~koopman/roses/dsn04/koopman04_crc_poly_embedded.pdf.

[8]. Greg Cook (26 March 2010). "Catalogue of parametrised LRC and CRC algorithms". http://regregex.bbcmicro.net/crc-catalogue.htm. Retrieved 5 June 2010.