IMPLEMENTATION OF FRAMING SEQUENCE ALGORITHMS IN NETSIM

ADARSHA SAGAR H.V.
MTech (4th SEM) CS&E
SSIT, Tumkur,
Karnataka, India.
asagar717@gmail.com

Prof.RAMESH D
Professor of CS&E
SSIT, Tumkur,
Karnataka, India.
rameshd_ssit@yahoo.com

Dr.M.Siddappa,
Professor & HOD of CS&E,
SSIT, Tumkur,
Karnataka, India.
siddappa.p@gmail.com

***Abstract*— NetSim is a network simulation tool used by the academic community for teaching,, The *Data Link Layer* deals with the algorithms for achieving reliable, efficient communication between two adjacent machines above the physical layer. *Framing* is a Protocol Data Unit (PDU) for data link layer is often called a frame with frame structure that contains Header, Data Field and Trailer.**

***Keywords :NetSim*,networksimulation,Datalinklayer,*Framing*.**

1. INTRODUCTION

The Data Link Layer deals with the algorithms for achieving reliable, efficient communication between two adjacent (i.e. physically connected by a communication channel like a wire) machines just above the physical layer. Data transfer data rate and error correction are the major concerns of the data link layer. Circuit errors, finite data rate and propagation delay have important implications for the efficiency of the data transfer. The protocols used for communications must take all these factors into consideration.

Functions of the data link layer:

1. Providing a well-defined service interface to the network layer
2. Determining how the bits of the physical layer are Grouped into frames
3. Dealing with transmission errors
4. Regulating the flow of frames so that slow

Receivers are not swamped by fast senders.

The data link layer can be designed to offer various services. Three possibilities that are commonly provided are:

1. **Unacknowledged connectionless service.**
2. **Acknowledged connectionless service.**
3. **Acknowledged connection-oriented service.**

Unacknowledged connectionless service consists of having the source machine send independent frames to the destination machine without having the destination machine acknowledge them. No connection is established beforehand or released afterward. Good channels with low error rates, for real-time traffic, such as speech

Acknowledged connectionless service When this service is offered, there are still no connections used, but each frame sent is individually acknowledged. This way, the sender knows whether or not a frame has arrived safely. Good for unreliable channels, such as wireless

The usual approach is for the data link layer to break the bit stream up into discrete frames and compute the checksum for each frame. When the frames arrive at the destination, the checksum is re-computed. There are four methods of breaking up the bit stream.

Connection-oriented service. With this service, the source and destination machines establish a connection before any data are transferred. Each frame sent over the connection is numbered, and the data link layer guarantees that each frame sent is received. Furthermore, it guarantees that each frame is received exactly once and that all frames are received in the right order.

When connection-oriented service is used, transfers have three distinct phases.

1. In the first phase the connection is established by having both sides initialize variable and counter need to keep track of which frames have been received and which ones have not.
2. In the second phase, one or more frames are actually transmitted.
3. In the third phase, the connection is released, freeing up the variables, buffers, and other resources used to maintain the connection

1. **Character count.**
2. **Starting and ending character stuffing.**
3. **Starting and ending flags, with bit stuffing.**
4. **Physical layer coding violations**

.

The first framing method, **Character count**, uses a field in the header to specify the number of characters in the frame.

When the data link layer at the destination sees the character count, it knows how many characters follow. **Problem:** count can possible be misrepresented by a transmission error. This method is rarely used anymore.

The second framing method, **Starting and ending character stuffing**, gets around the problem of resynchronization after an error by having each frame start with the ASCII character sequence DLE STX and end with the sequence DLE ETX. (DLE is Data Link Escape, STX is Start of Text, and ETX is End of Text). **Problem:** a serious problem occurs with this method when binary data, such as object programs or floating-point numbers, are being transmitted it is possible that the DLE, STX, and ETX characters can occur, which will interfere with the framing. One way to solve this problem is to have the sender's data link layer insert and DLE character just before each "accidental" DLE and the data link layer on the other machine removes them before it gives the data to the network layer, this is called **Character Stuffing.**

The third method, starting and ending flags with bit stuffing allows data frames to contain and arbitrary number of bits and allows character codes with an arbitrary number of bits per character. Each frame begins and ends with a special bit pattern, 01111110, called a **flag** byte.

Whenever the sender's data link layer encounters five consecutive ones in the data, it automatically stuffs a 0 bit into the outgoing bit stream, which is called **bit stuffing.** The receiving machine destuffs the 0 bit.

The fourth method, **Physical coding violations**, is only applicable to networks in which the encoding on the physical medium contains some redundancy. For example, some LANs encode 1 bit of data by using 2 physical bits**.**

2.CONCEPT OF BIT STUFFING

When data is transmitted, the transmitter should distinguish between the start and the end of the data. This is done to ensure that the receiving node reads the data completely. This is accomplished by appending a unique pattern of bit stream before and after the data.

Care should be taken to ensure that this unique pattern does not occur in the data itself. If the pattern appears in the data then the data is broken .The process of appending a unique pattern and breaking the pattern is known as **Framing.**

Bit stuffing is one of the framing methods in which each frame begins and ends with a special bit pattern '01111110', called a flag byte.

Whenever the sender's data link layer encounters six consecutives 1s in the data, it automatically stuffs a 0 bit into the outgoing bit stream to break the pattern

3.CONCEPT OF CHARACTER STUFFING

When data is transmitted, the transmitter should distinguish between the start and the end of the data. This is done to ensure that the receiving node reads the data completely. This is accomplished by appending a unique pattern of bit stream before and after the data.

Care should be taken to ensure that this unique pattern does not occur in the data itself. If the pattern appears in the data then the data is broken .The process of appending a unique pattern and breaking the pattern is known as **Framing.**

Character Stuffing is one of the framing methods in which each frame starts with DLE STX (Starting Delimiter) character sequence: ends with DLE ETX (Ending Delimiter) character sequence.

Characters for DLE STX or DLE ETX may occur in the data .To solve this problem the transmitting node insult a DLE character just before each DLE character occurring in the data

4. ALGORITHMS

A. Algorithm for Bit Stuffing

1. Read the input file "input.txt" that contains the input data.

2. Concatenate the Bit Strings contained in the **Destination address**, **Source address**, **Data, Checksum** into a **single** temporary Bit string called Frame String.

Check the **Frame String** bit by bit. if the Frame string contains consecutive five 1s then inserts 0 to break the bit pattern.

1. Repeat step 3 for the whole Bit string.
2. Add bit pattern "01111110" to the stuffed Frame string and store it as HDLC frame string.
3. De-Concatenate the Frame String into Destination address, source address, Data, Checksum Bit strings.
4. Write the output data in the output file "output.txt"

.

B. Algorithm for Character Stuffing

1. Read the input file input.txt that contains the input data.
2. Concatenate the strings contained in Destination address, Source address, Data, Checksum Into a single temporary string called frame string.
3. Check the Frame String by each character if the Frame String by each character .if the frame String contains starting delimiter, then inserts an extra starting delimiter (before current Character) to break the character sequence pattern.
4. Repeat step 3 for the whole string.
5. Add the delimiters at both ends of the stuffed Frame String and store it as HDLC Frame String.
6. De- concatenate the frame string into the Destination address, Source address, Data, Checksum Bit strings

5. ADVANTAGES AND DISADVANTAGES

Advantages and Disadvantages of Bit Stuffing:

The main **advantage** of Bit stuffing such as for bringing bit streams that do not necessarily have the same or rationally related bit rates up to a common rate, or to fill buffers or frames.

1. The location of the stuffing bits is communicated to the receiving end of the data link, where these extra bits are removed to return the bit streams to their original bit rates or form.

2. Bit stuffing may be used to synchronize several channels before multiplexing or to rate-match two single channels to each other.

3. Another use of bit stuffing is for run length method coding: to limit the number of consecutive bits of the same value in the data to be transmitted.

4. Bit stuffing does not ensure that the payload is intact (*i.e.* not corrupted by transmission errors); it is merely a way of attempting to ensure that the transmission starts and ends at the correct place.

The main **disadvantage** of this form of bit-stuffing is that the code rate is unpredictable; it depends on the data being transmitted.

Advantages and Disadvantages of Character Stuffing:

Character stuffing is a process that transforms a sequence of data bytes that may contain 'illegal' or 'reserved' values into a potentially longer sequence that contains no occurrences of those values.

The packet is framed by 2 characters at either end: <DLE><STX> ... (packet goes here) ... <DLE><ETX> (Data Link Escape, Start and End of Text). If <DLE> appears in the data stream itself, another <DLE> is inserted to make a <DLE><DLE> pair, so that it is not mistaken for an end of packet signal.

The opposite occurs at the receiving end. <DLE> followed by anything other than <STX>, <ETX> or <DLE> must be an error. This is often done by software in the device handler. This is disadvantage of Character stuffing.

6.CONCLUSIONS.

Bit stuffing does not ensure that the payload is intact (*i.e.* not corrupted by transmission errors); it is merely a way of attempting to ensure that the transmission starts and ends at the correct places. Error detection and correction techniques are used to check the frame for corruption after its delivery and, if necessary, the frame will be resent transmitted Packet. The purpose of byte stuffing is to convert data packets into a form suitable for transmission over a serial medium like a telephone line.

Boundaries between packets are marked by a special reserved byte value and byte stuffing ensures, at the cost of a potential Increase in packet size, that this reserved value does not inadvertently appear in the body of any transmitted packet.

7. REFERENCES

[1] W. Stallings, *Data and computer communications*. Prentice Hall,2004.

[2] W. Simpson, "The point-to-point protocol (PPP)," RFC 1661, 1994.

[3] S. Cheshire and M. Baker, "Consistent overhead byte stuffing," *IEEE Transactions on Networking*, vol. 7, pp. 159–172, Apr. 1999.

[4] J. Romkey, "Nonstandard for transmission of IP datagrams over serial lines: SLIP," RFC 1055, 1988.

[5] W. Simpson PPP in HDLC-like framing" RFC 1662, 1994.

[6] D. Knuth, *the Art of Computer Programming - Semi numerical Algorithms*. Addison Wesley, 1997, vol. 2

[7] C.Duan, A.Tirumala, and S. P. Khatri, "Analysis and avoidance of cross-talk in on-chip buses," in *Proceedings IEEE Annual Symposium on High Performance Interconnects (HOTI'01)*, Stanford, CA, USA, August 22–24, 2001, pp. 133–138.

[8] C.Duan and S. P. Khatri, "Exploiting crosstalk to speed up on-chip buses," in *Proceedings Conference and Exhibition on Design, Automation, and Test in Europe (DATE'04)*, Paris, France, February 16–20, 2004, vol. 2, pp. 778–783.