

COVER PAGE

DYNAMIC ALLOCATION OF BUFFER SPACE IN A BRIDGE USING TREND BASED ANALYSIS

Kaushik Muhuri **, k_muhuri@yahoo.com , (033)3212724

** INDIAN INSTITUTE OF INFORMATION TECHNOLOGY-CALCUTTA
SALT LAKE CITY,CALCUTTA,INDIA-700 091

Anirban Banerjee * , anir_iit@yahoo.co.uk , (033)3212724

* INDIAN INSTITUTE OF INFORMATION TECHNOLOGY-CALCUTTA
SALT LAKE CITY,CALCUTTA,INDIA-700 091

DYNAMIC ALLOCATION OF BUFFER SPACE IN A BRIDGE USING TREND BASED ANALYSIS

Kaushik Muhuri , Anirban Banerjee***

**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY-CALCUTTA
SALT LAKE CITY, CALCUTTA, INDIA-700 091**

Abstract

We are going to investigate the throughput performance of networks interconnected via a bridge under an ordinary traffic environment while employing an efficient buffer management scheme. Our intention is to develop an optimal algorithm that will efficiently allocate buffer space to packets traversing the bridge, in a dynamic manner . The algorithm is aimed at maximization of buffer space utilization. The simulated networks are connected through a bridge which is not involved in correcting errors in the packets or more generally do not modify frames in any way.

1. Introduction

Local Area Networks (LAN's) as we know have evolved greatly in recent years, to accommodate new and emerging applications such as real – time voice and video traffic, high speed data from graphics systems and many other demanding applications . Such a system must be able to satisfy delay constraints imposed on it , which become very tight when real – time traffic is involved . Local Area Network (LAN) serves the interconnection needs of nodes distributed within a diameter of a few kilometers. LAN's might ultimately be used as integral blocks of a Wide Area Network (WAN) .

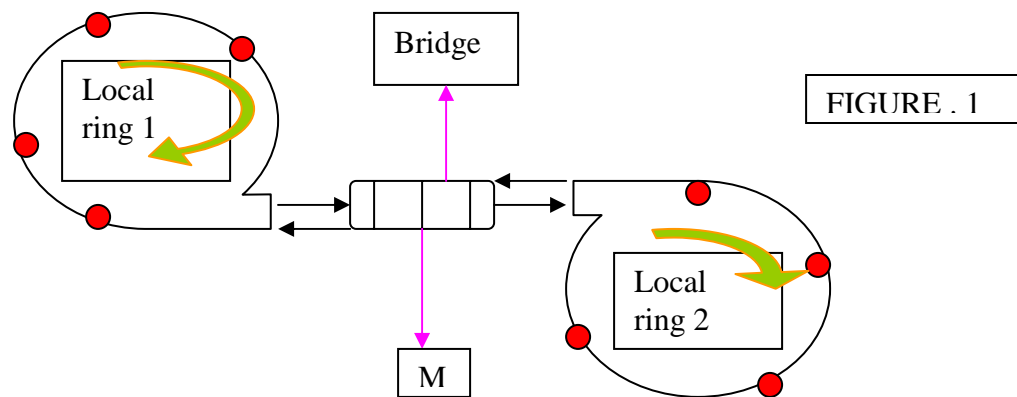
Whenever a LAN can no longer support any more users or the delay characteristics become too tight for comfort it becomes necessary to try and connect independent autonomous sub networks. A bridge or for that

matter a gateway can serve this purpose. A bridge provides a connection at the Data Link Layer. Ring implementation requires active repeaters and substantial logic working at channel speeds (e.g. address recognition , flag setting) at each station . It is imperative for the bridge functions to be simple in order to meet the needs for interconnecting networks at reasonable costs. This excludes them from performing any complex flow or error control. In case of congestion, bridges simply begin to discard frames that they cannot handle . Here the discarded frames will not be retrieved by using any sort of end – to – end protocol between the communicating stations.

In this paper we have developed an efficient buffer management scheme for a bridge providing a link between two token ring networks . The packets are stored in the bridge and are forwarded on the other ring when a token is available . The buffer management scheme is basically aimed at improvement of buffer utilization whereas keeping the throughput close to the case where in no buffer management scheme is used . The packet format used here is as stated in the IEEE 802.5 format . The algorithm works closely on a basic principle related closely to trend analysis . It senses the network load within an interval of time and predicts the next possible buffer requirements using analysis based on a finite number of requests that have already been handled by the system . The actual requirement of buffer space is used as a fed back in the next prediction phase . The duration or granularity of the prediction process depends on the requirements of the user , we have fixed the duration of the prediction scheme to the last ten requests .

2. Ring Networks Connected By a Bridge

The network topology with two token passing ring networks interconnected through a bridge is shown in FIGURE . 1 . The node stations are represented by dots . The arrow within each ring represents the direction of information flow for each ring . ‘ M ‘ is the common memory pool where inter ring packets are stored .



BASIC TOKEN RING STRUCTURE

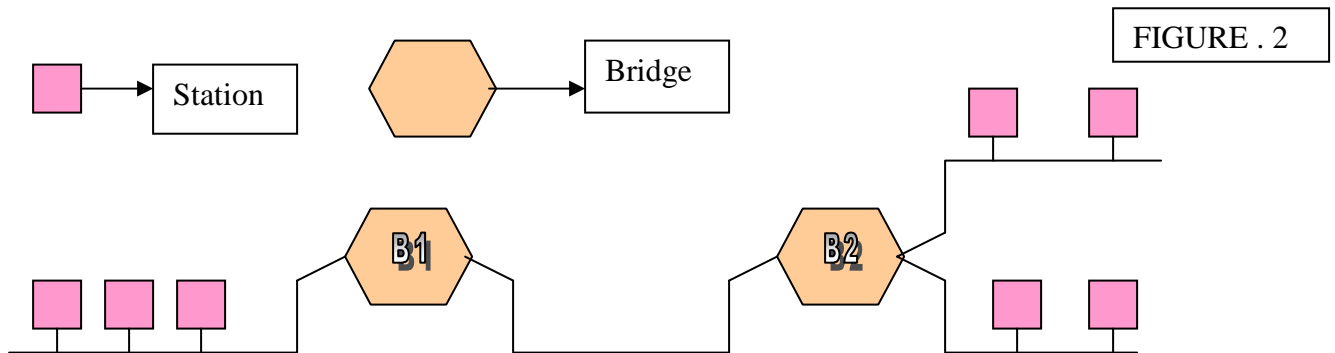
The token ring standard used here is as described in the IEEE 802.5 standard.

3. Token Rings

The token ring standard used here is as described in the IEEE 802.5 standard. It defines a priority mechanism by which eight levels of priorities can be provided. We have chosen the non – priority mode of operation , i.e. we assume that all stations and the bridge operate at the same ring access priority level and are only allowed to transmit a single frame per token .

4. Bridge

A bridge provides a basic store and forward function . Frames are buffered on the bridge until they can be forwarded on the other ring where the destination station is located. Bridges do not examine the network layer header and thus can handle IP, IPX and OSI packets equally well.



BASIC BRIDGE STRUCTURE

The buffer pool is organized as a single ended queue structure. We assume that the processors associated with the bridge are fast enough to perform flushing and other buffer maintaining activities.

5. Stations

It is assumed that the stations are involved in ordinary traffic generation only. The messages generated by the application layer are passed on to the Logical Link Control (LLC) sub layer . Either a complete message is passed or it is passed by fragmenting it into parts when the message length exceeds the maximum frame length specified by the LLC.

The layers above the LLC are responsible for the segmentation and reassembly of messages

The LLC users are supplied with data units in the form of Information frames (I – frames) by the LLC layer . If the LLC cannot transmit data frames at least at the rate at which it receives , it would apply backpressure on the upper layers . A data unit supplied by a higher layer entity is not accepted by the LLC when one or more of the following events occur-

- 1) certain number of I – frames waiting for transmission
- 2) certain number of data units not yet processed

However in our case , the packets do not wait for the acknowledgement after their transmission . This mechanism is quite common in case of real – time traffic .

6. Network Performance Using Buffer Management Algorithms

The bridge is usually implemented on a system which belongs to both the rings . The bridge buffer shares it's memory space with the other processes running on the node – station on which it is implemented . The traffic load passing through the bridge fluctuates with time and thus allocating fixed amounts of buffer space is quite wasteful and inefficient . Here dynamic allocation of memory space is made depending on present traffic load and the past statistics that has been registered by the bridge . The buffer management strategy is to monitor the network traffic characteristics within a particular interval of time and allocate buffer space accordingly . This interval is referred to as the observation period .

The algorithm developed attempts to register the packet load that traversed through the bridge during the the last 3 observation periods and tries to fit a polynomial curve of a degree chosen according to the user , in order to register the pattern of requests for buffer space that have been registered during the last few observation periods .

7. Algorithm

The pseudo code is as below :

```
num_of_points=0 ;
while (1) {
    if ( num_of_points == 3 ) then
    {
        move_elements_one_step_left ( p_load ) ;
        move_elements_one_step_left ( overflow ) ;
        num_of_points = 3 ;
    }
    read p_load [ num_of_points ] = sum ( packets_size ) ;
    if ( num_of_point == 0 ) then
    {
        forecast = p_load[num_of_points] ;
        if ( ub_bytes > 50 ) then
        {
            forecast += 0.5 * ub_bytes ;
            overflow [0] = 1 ;
        }
        else overflow [0] = 0 ;
    }
    else
```

```

{
  forecast = curve_of_best_fit ( p_load , 2 ) ;

  if ( ub_bytes > 50 )
  {
    if ( overflow [ num_of_points - 1 ] == 1 ) then

      {
        forecast += 1.5 * ub_bytes ;

        overflow [ num_of_points ] = 1 ;

      }

    else
    {
      forecast += 0.5 * ub_bytes ;

      overflow [ num_of_points ] = 1 ;
    }

    } overflow [ num_of_points ] = 0 ;
  }

  t = forecast - 0.5 * buff ;

  if ( buff > 50 ) && ( t > 0 ) then forecast = t ;

  a = forecast - buff ;

  if ( buff < 300 ) then buff = 300 ;

  if ( mem - a > 0 ) then

    {
      mem = mem - a ;

      buff = round ( forecast ) ;
    }
  ++ num_of_points ;
}

```

ABBREVIATIONS USED

P_load : packet load

Ub_bytes : unbuffered bytes in the last prediction interval

Buff : free buffer area in bytes

Mem : size of common memory pool

Forecast : next predicted memory request

Num_of_points : number of points in which the polynomial curve tries to fit itself

Overflow [] : an array that stores information about whether buffer space allocated in the last few prediction intervals was not enough

Packets_size : the actual size of the packets constituting the packet load

It is absolutely clear from the above steps that the buffer space allocated for the packets traversing through the bridge is predicted on the basis of past performance of the system, this data pool is actually built by the arrays **p_load** & **overflow**. If a considerable number of bytes overflow in the last prediction interval then the forecast is increased by 0.5 to 1.5 times the forecasted allotment of memory space. On the other hand if it is decreased by half the amount of unused buffer if a considerable amount of the buffer remains unutilized. The minimum buffer size is maintained at 300 bytes. After the forecasting phase is over the memory pool is checked, whether excess memory is available or not. If forecast is less than the buffer space then the excess memory is returned to the memory pool.

The traffic statistics used to verify the algorithm is as given in CHART 1.

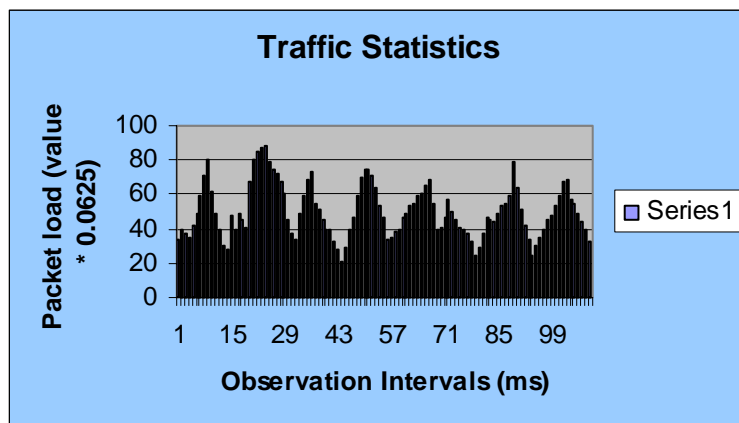


CHART 1.

The detrended normal probability plot of the traffic statistics used to verify the algorithm is given below in CHART 2.

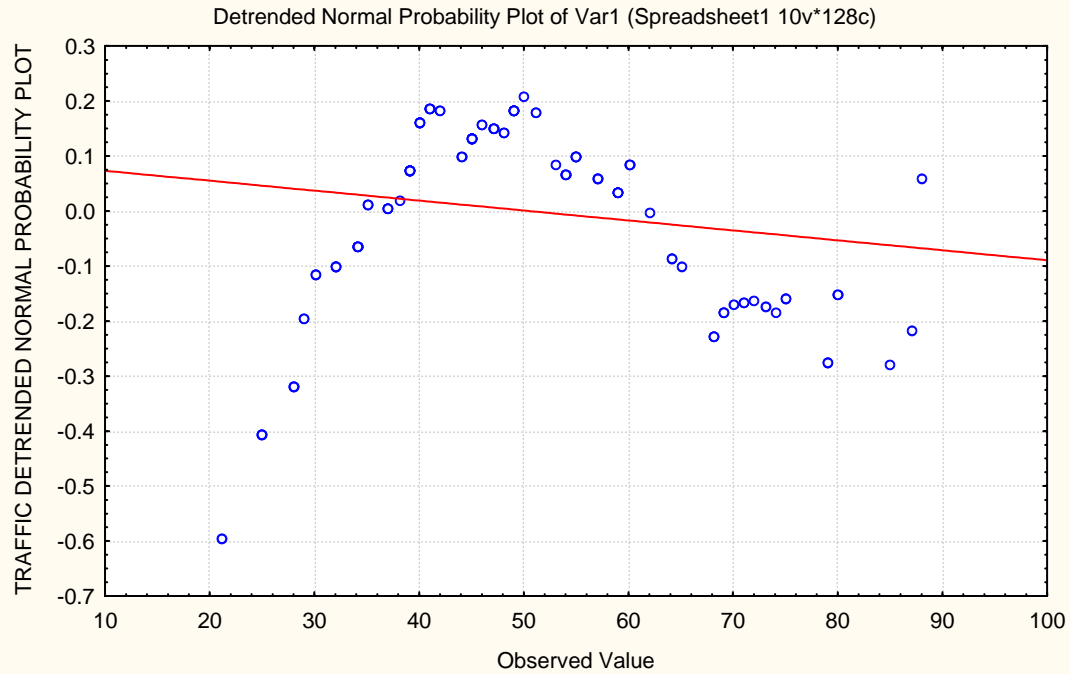


CHART 2.

The statistics observed on varying the number of sample points to be fed to the prediction module are as presented in CHART 3. ,

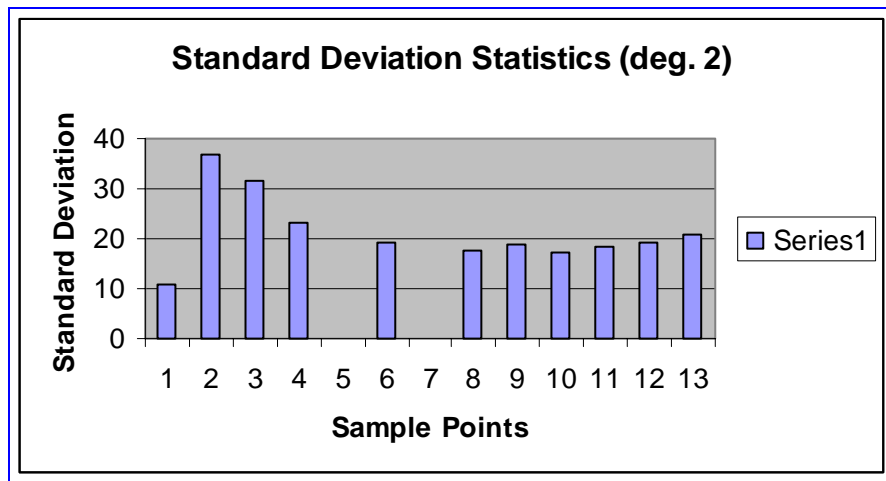


CHART 3.

It was observed that increasing the degree of the polynomial curve used in the prediction algorithm does not lead to a significant difference in the accuracy of prediction , the statistics for standard deviation of a polynomial curve of degree 3 is as given in CHART 4.

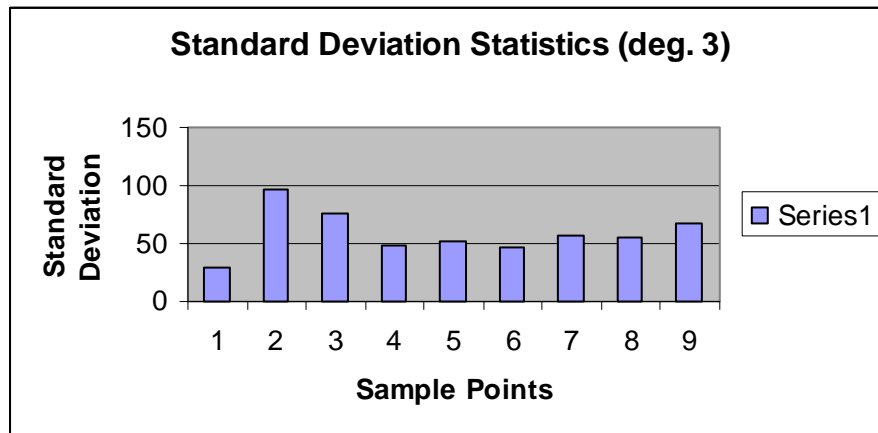


CHART 4.

However one drawback of using a curve of lower degree , e.g. 2 is that it cannot adjust to severe fluctuations , about more than 50% of the previous value , with low number of sample points such as 4 or 5 , very accurately (REF. CHART 5.).

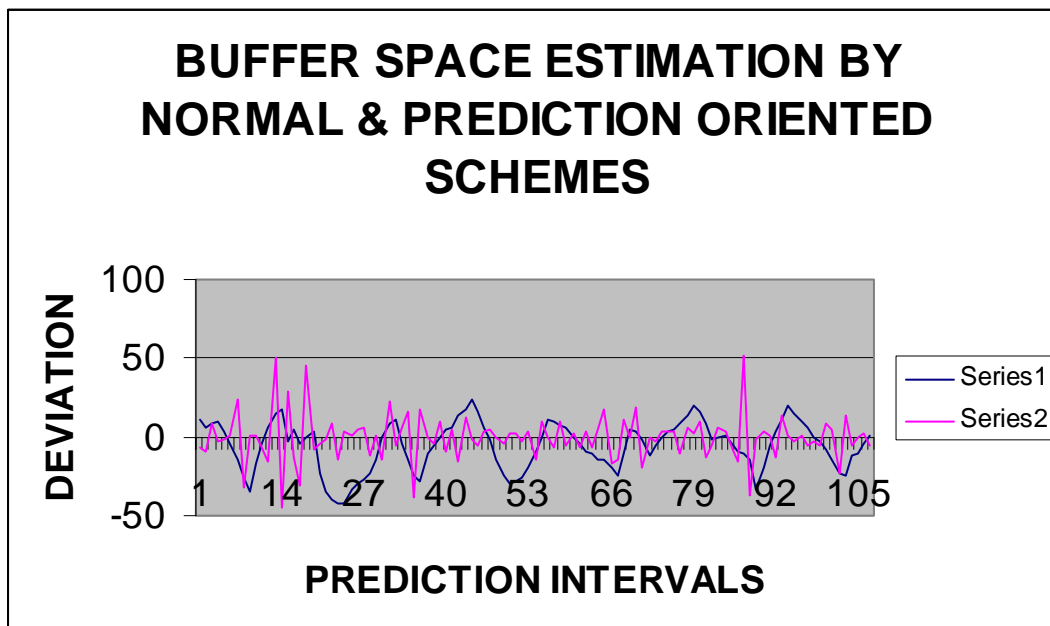


CHART 5.

NORMAL SCHEME

PREDICTION SCHEME (deg. 2 , sample points 3)

The standard deviation and other statistics of the normal scheme in comparison to the prediction module is given below :

		NORMAL	PREDICTIVE MODULE
MEAN	-	- 5 . 22727	- 0 . 47664
MEDIAN	-	-3 . 0	0 . 0
STANDARD DEVIATION	-	15 . 41158	14 . 82371

68. Conclusion

We thus observe that using the algorithm described in the above article the buffer space allocation to processes requiring memory space in the bridge joining two token ring networks is 4% better , while considering the standard deviation & 90.881% considering mean (curve of degree 2 , 3 sample points) , this figure becomes significant at normal traffic lode of about 1000 octets of user information within the packets competing for buffer space in the memory of the bridge . Another significant observation is that the fluctuations in allocation of buffer space to satisfy the packet load is much larger at some specific points of time while in the predictive mechanism the fluctuations are much more smoother which would allow the bridge to handle so called sudden traffic bursts better , the bridge employing the predictive mechanism is not easily overwhelmed by sharp increase or decrease in traffic flow , unlike the normal allocation scheme.

REFERENCES

- 1) Andrew S Tanenbaum “ Computer Networks ” , Prentice Hall , India , 2000
- 2) William Stallings “ Computer and Data Communications ” , Prentice Hall , India , 2000
- 3) Douglas Comer “ TCP/IP Networking ” , Vol-1,3 , Prentice Hall , India , 2000
- 4) Cormen , Rivest , Lieserson “ Introduction to Algorithms ” , Addison Welsely , India , 2000