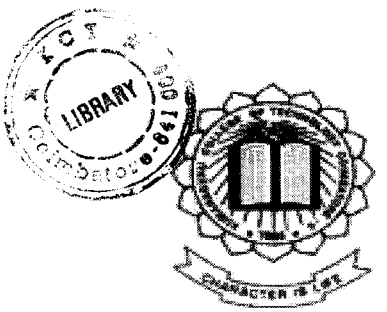
HIGH PERFORMANCE DATA CARRIER



A PROJECT REPORT

P1502

Submitted by

NAME	REG.NO
SANTHOSHI .J	**71201104047**
SATHYA.K.G	**71201104052**

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY:: CHENNAI 600025

APRIL 2005

ANNA UNIVERSITY: CHENNAI 600025

BONAFIDE CERTIFICATE

Certified that this project report **"HIGH PERFORMANCE DATA CARRIER"** is the bonafide work of **SANTHOSHI.J (71201104047), and SATHYA.K.G (71201104052),** who carried out the project work under my supervision.

SIGNATURE

Dr. S. Thangasamy

HEAD OF THE DEPARTMENT

Department of
Computer Science and Engg,
Kumaraguru College of Technology,
Chinnavedampatti P.O.,
Coimbatore - 641006

SIGNATURE

Mrs.V.Vanitha

SUPERVISOR

Senior Lecturer

Department of
Computer Science and Engg,
Kumaraguru College of Technology,
Chinnavedampatti P.O.,
Coimbatore - 641006

Submitted for the Viva-voce Examination held on 20.4.2005

Internal Examiner

External Examiner

ANNA UNIVERSITY: CHENNAI 600025

EVALUATION CERTIFICATE

College : **KUMARAGURU COLLEGE OF TECHNOLOGY**

Branch : **COMPUTER SCIENCE AND ENGINEERING**

Semester: **EIGHT (08)**

S.No	Name of the Student	Title of the Project	Name of Supervisor
1	SANTHOSHI.J	High Performance Data Carrier	Mrs.V.Vanitha, M.E., Senior Lecturer
2	SATHYA.K.G	High Performance Data Carrier	Mrs.V.Vanitha, M.E., Senior Lecturer

The report of the project work submitted by the above students in partial fulfillment for the award of BACHELOR OF ENGINEERING degree in COMPUTER SCIENCE AND ENGINEERING of Anna University were evaluated and confirmed to be the report of the work done by the above students and then evaluated.

(INTERNAL EXAMINER) **(EXTERNAL EXAMINER)**

DECLARATION

We hereby declare that the project entitled **"HIGH PERFORMANCE DATA CARRIER"**, is a record of original work done by us and to the best of our knowledge, a similar work has not been submitted to Anna University or any other institution, for fulfillment of the requirement of the course study.

This report is submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Computer Science and Engineering of Anna University, Chennai.

Place: Coimbatore
Date: 13.4.05

Santhoshi. J
(SANTHOSHI.J)

Sathya.K.G
(SATHYA.K.G)

ABSTRACT

In a network, when the number of packets transmitted are within its carrying capacity then the number of packets delivered is proportional to the number sent. But the routers are no longer able to cope and begin to lose packets, as traffic increases too far. This situation is called congestion. To alleviate congestion, numerous congestion control algorithms have been proposed. **Active Queue Management Algorithms** are some of the prominent Congestion control algorithms. RED, BLUE and ECN are Active Queue Management mechanisms.

Random Early Detection (RED) algorithm detects incipient congestion *early* by computing the average queue size. It does not keep the state of each flow. That is, it puts the data from the all the flows into one single queue, and focuses on their overall performance. RED uses the length of the queue as its congestion indicator, which provides very little information on the severity of congestion. In view of this shortcoming of the RED, nominal changes are made to it to enhance its performance and utilize the maximum bandwidth of the link. With the changes implemented in the original RED, Altered RED comes into existence.

In order to evaluate the performance of the Altered RED, comparisons are made with the Active Queue Management Algorithms BLUE and ECN. The BLUE and ECN algorithm has been proved to perform significantly better than the Original RED in terms of packet loss rates. The implementation of Altered RED achieves reduced packet loss and hence works in equivalence to the performance of BLUE and ECN.

ACKNOWLEDGEMENT

In the venture of synthesizing the draft of this project, we extend our sincere gratitude to our beloved parents who showered the necessary wisdom, support and encouragement to do this course **B.E.,(COMPUTER SCIENCE AND ENGINEERING)** in this prestigious institution.

We express our profound and sincere thanks to principal **Dr.K.K.Padmanabhan, Ph.D.,** for the excellent facilities made available to accomplish the project work.

We deem it a privilege to record our sincere thanks to **Prof. S.Thangasamy Ph.D.,** Head of the Department of Computer Science and Engineering for his valuable suggestions and motivations during the entire period of this course.

We would like to express our heartfelt gratitude to our guide and class advisor **Mrs.V.Vanitha, M.E.** Senior Lecturer, Department of Computer Science and Engineering for her valuable guidance, suggestions, consistent encouragement, inspiration, tolerance, guidance and support offered in successful completion of the project.

Last but not the least, we thank all the teaching and non-teaching staff members of the college for their kind co-operation throughout this project work.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Introduction

1.INTRODUCTION

In the current day Internet with high-speed data transfer soon becoming the mode, congestion of these networks is an unavoidable and significant problem. Due to congestion, high packet loss rates occur. When a packet is lost before it reaches its destination, all the resources it has consumed in transit are wasted. The explosive growth of the Internet makes it essential to devise and deploy effective congestion control at the transport layer. With most of the Internet traffic transported by the TCP (Transmission Control Protocol), TCP congestion control plays an important role in avoiding the congestion collapse of the Internet.

TCP relies on packet drops as the indication of congestion. The TCP source detects dropped packets either from the receipt of three duplicate acknowledgments or after the time-out of a retransmit timer. It responds to a dropped packet by reducing congestion window. They also respond to ICMP source quench message.

Slow start recovery and *Multiplicative decrease* are two common congestion control techniques of TCP. In slow start mechanism, the size of the congestion window is increased by one with every incoming acknowlegemnt at the source.In the Multiplicative decrease, there is a reduction in the congestion window by half for every packet loss.

1.1 Existing System and its Limitations

Random **E**arly **D**etection is an Active Queue Management Algorithm that has been deployed in current networks for Congestion avoidance and reducing packet loss. It detects incipient congestion by computing the average queue size. When the average queue size exceeds the preset threshold, the algorithm starts randomly dropping or marking the packets based on a calculated probability. The TCP source is notified of congestion as a result of the packet drop.

The **RED** Congestion control mechanisms monitors the average queue size for each output queue and using randomization, choose connections to notify of that congestion. Transient congestion is accommodated by a temporary increase in the queue. The probability that a connection is notified of congestion is proportional to that connections share of the throughput through the router.

The RED algorithm:

The average queue size is calculated using a low-pass filter with an Exponential Weighted Moving Average. The average queue size is compared to two thresholds, a Minimum threshold and a Maximum threshold. When the average queue size is less than the minimum threshold no packets are dropped. All the arriving packets are dropped when the average queue size is greater than the maximum threshold. When it is between the minimum and maximum threshold, each arriving packet is

dropped with a marking probability p_a where p_a is a function of the average queue size.

for each packet arrival
    calculate the average queue size *avg*
    if min_{th} <= *avg* < max_{th}
        calculate the probability p_a
        with probability p_a:
mark the arriving packet
    else if max_{th} <= *avg*
        mark the arriving packet.

Figure 1:General algorithm for RED gateways.

Two computations are done in the algorithm. The average queue size computed is used to determine the degree of burstiness that will be allowed in the queue of the router. The packet marking probability determines how frequently the router drops packets. The goal of the router is to drop packets at fairly even spaced intervals in order to avoid biases, to avoid global synchronization and to control average queue size.

As the average varies from minimum threshold to maximum threshold, the packet marking probability p_b varies linearly from 0 to max_p.

$p_b = max_p \times (avg - min_{th})/(max_{th} - min_{th})$ [1]

Where $p_a = p_b/(1 - count \times p_b)$

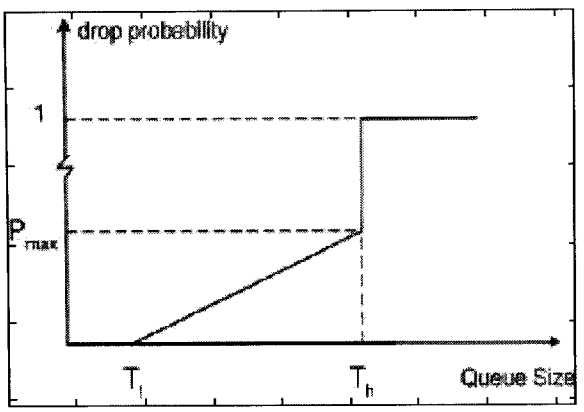


Figure 2: Packet Marking Probability

RED does not keep the state of each flow. That is, it puts the data from the all the flows into one queue, and focuses on their overall performance. The length of the single queue gives very little information as to the severity of congestion i.e., the number of competing connections sharing the link. The main limitation in the RED is that there occurs a continuous loss of packets when the exponentially weighted moving average queue size exceeds the set maximum threshold. This affects the stability of packet loss sensitive networks.

1.2 Proposed System and its advantages

The **Altered RED** is a modified version of the original RED. The modifications are done in such a manner that it improves the performance of the congested network and reduces the packet loss when compared to the original RED. It does so by segmenting the bandwidth of the link.

The Altered RED algorithm functions at a basic level on the original RED. Its main parameters are minimum threshold, maximum threshold, and packet marking probability and moving average queue size. It also requires the computations of the link bandwidth. The Altered RED algorithm is triggered when the congestion in the network is at its peak i.e., when the average queue size exceeds the maximum threshold. In case of the original RED, the packets are randomly dropped when the moving average queue size exceeds the minimum threshold and all the incoming packets are dropped when the moving average queue size exceeds the maximum threshold.

After the Altered RED starts to function, thereafter all the incoming traffic will be routed through the link in a parallel manner utilizing the maximum bandwidth of the link. Thus when compared to the original RED, there is a profound reduction of packet loss in Altered RED. Also the traffic in the network is cleared sufficiently.

Proposed Line of Attack

2.PROPOSED LINE OF ATTACK

The project aims to justify the performance of Altered RED in comparison to prominent Active Queue Management algorithms BLUE and ECN using the simulator Network II.5.

2.1 NETWORK II.5

NETWORK II.5 is a design tool, which takes a computer system description the user specifies and provides measures of hardware utilization, software execution, message delivery times, response times and contention. It is intended for the person who designs or specifies computer systems and computer networks. NETWORK II.5 is used both to evaluate the ability of a proposed system configuration to meet the required workload and to evaluate competing designs. NETWORK II.5 is designed to model a wide variety of computer architectures, from a single processor to a complex system of processors and storage devices connected in a network. It is extremely flexible, allowing the portions of a computer system of special interest to be modeled at a detailed level while the rest of the system is modeled at a coarser level. Most importantly, NETWORK II.5 is a tool and not a computer language.

2.1.1 Functions of NII.5

The NETWORK II.5 package provides three main functions; system description, system simulation and simulation analysis. System description is handled by graphical layout of the hardware elements and the use of graphics based forms to describe the component attributes. System simulation is simply a matter of invoking the simulation program for the system description that has been built. NETWORK II.5 does not require any programming, so there are never any programming delays.

The computer system to be simulated is described in a data structure consisting of Processing Elements, Gateways, Transfer Devices, LANs, Storage Devices, Modules, and Files. Each of these building blocks has a series of attributes whose values are supplied by the user.

The powerful NETWORK II.5 icon-oriented graphic user interface allows a user to build and maintain the requisite data structure by means of simple but powerful graphical operations. NETWORK II.5 stores the system description created graphically in a text network description data file, which describes the hardware and software of the system to be simulated. After acquiring the runtime control parameters (such as simulation length, devices to graph) from the user, NETWORK II.5 builds and executes the simulation. The user can optionally monitor the simulation as it progresses through the use of trace and snapshot reports.

Running NETWORK II.5 produces periodic and end-of simulation reports. Animated displays of NETWORK II.5 simulations provide a view of the

modeled computer system in operation. It is easy to define the location, color and icon of every device in a NETWORK II.5 simulation.

2.2 Simulation of Altered RED in NII.5

A simple topology consisting of two networks connected over a bridge is defined in the Network (User interface of NII.5). The Altered RED algorithm is implemented in the server that is connected to the bridge. This server is located in the area of the network where there is a high probability of congestion occurrences. This server is connected to two other servers, which frequently transmit data to servers located in the other side of the bridge. There occurs congestion when these servers simultaneously request transmission of data. This results in a persistent queue building up and the Altered RED algorithm is triggered. The functioning of the Altered RED in transmitting data over the bridge is obtained in the form of a dynamic chart at the end of simulation. This represents the average utilization of the link.

2.3 Comparison of Altered RED with BLUE and ECN- A Simulation

The performance of the Altered RED algorithm is justified by means of making a comparison with the prominent Active Queue Management algorithms BLUE and ECN. The parameters of comparison are link utilization and packet loss .In order to show the comparison being made, the algorithms BLUE and ECN are implemented in servers that are located in the second network.

There is a continuous flow of data across the two networks, due to which there is a high probability of occurrence of congestion in the servers connected to the bridge. When the congestion is detected, the algorithms BLUE and ECN are triggered and start taking measures in controlling the congestion. This is depicted in the resulting chart at the end of simulation. With the implementation of Altered RED in the network there is a profound reduction in packet loss. The algorithms BLUE and ECN achieve almost the same amount of reduction in packet loss. From the simulation results, it is clearly visible that the Altered RED algorithm is working in equal to the algorithms BLUE and ECN

Proposed Methodology

3.PROPOSED METHODOLOGY

Introduction

In current day networks, congestion scenarios occur frequently resulting in packet loss. Nominal packet loss is expected but with every packet loss there is a waste of a large number of resources consumed in transit. In order to prevent such heavy loss or at least reduce, many congestion control algorithms have been proposed. The traditional way for controlling queue length at routers is the Drop Tail. Drop tail sets a maximum length for each queue at the router and accepts every packet until the maximum queue length is reached. Once the maximum queue size is reached, the algorithm drops packets until the queue size is again below the maximum. This technique results in some serious drawbacks.

In Active queue management approach, the packets are dropped even before the queue becomes full so that sources slow down because of this congestion notification, thus preventing the queue from overflowing. RED is one such algorithm. In RED random packet loss occurs when the average queue size exceeds the preset minimum threshold. But when the sources are relentlessly transmitting data, the congestion is at its peak and the average queue size exceeds the maximum threshold. In this case RED fails to control congestion and reduce packet loss, in lieu of which the methodology of Altered RED is proposed.

3.1 The Altered RED

The Altered RED is an algorithm that can be implemented in networks that transmit highly sensitive data packets. It makes an effort in reducing the packet loss to a minimum in comparison to that achieved by the original RED. Its main goal is to control the congestion by orienting the data in a parallel fashion, making them travel simultaneously occupying the maximum bandwidth of the link.

3.1.1 Parameters

The main parameters of Altered RED are

Exponential weighted moving average	avg
Minimum threshold	min_{th}
Maximum threshold	max_{th}
Packet marking Probability	p_a
Bandwidth of the link	b

Table 1: Parameters of Altered RED

3.1.2 The Algorithm

The exponential weighted moving average queue size is computed for each incoming packet. When the average queue size exceeds the minimum threshold, the incoming packets are dropped randomly depending on the calculated marking probability. When the congestion in the network is persistent and when the average queue size exceeds the maximum threshold, the Altered RED algorithm segments the bandwidth of the link. Depending on the segmented width of the bandwidth, the arriving packets are made to traverse in a parallel fashion. Hence the bandwidth of the link is utilized to the maximum. Thus when compared to the original RED where the packets are dropped once the average queue size exceeds the maximum threshold, there is a profound reduction of packet loss in Altered RED. Also the traffic in the network is cleared sufficiently.

for each packet arrival

calculate the average queue size **avg**

if min_{th} <= *avg* < max_{th}

calculate the probability p_a

with probability p_a: mark the arriving packet

else if max_{th} <= *avg*

/**Trigger Altered RED**/

calculate the maximum bandwidth of the link.

Segment the bandwidth b of the link

forward the packets from the single queue in a parallel manner.

Figure 3: General algorithm for Altered RED.

Programming Environment

4. PROGRAMMING ENVIRONMENT

4.1 Hardware Requirements

Processor: Intel Pentium 4

Memory: 112 MB RAM

Hard disk Drive: 60 GB

4.2 Software Requirements

Operating Systems: Windows 98

Simulator used: Network II.5 Release 12.0, Nov 1997

Memory 13.9 MB

Compiler: Microsoft visual C++ 6.0

Detailed Implementation

5. DETAILED IMPLEMENTATION

5.1 The Topology

The simulated system consists of two "rings", where a "bridge" connects the rings. A ring is a store and forward network where nodes are interconnected by point-to-point, uni-directional links. Each node is addressable from any point in the network. When a node sends a message, the message is circulated through each node on the loop. Each node that receives the message must forward the message to the next downstream node. The message will eventually arrive at the original source node where it is removed from circulation. A node may have only one outstanding sourced message at any given time. It will, however, forward a message not sourced by it whether it has an outstanding sourced message or not. If a message is destined for a node on the other ring, the node attached to the bridge will make a copy of the message and forward the copy to the bridge. The bridge will then become the source of the message on the opposite ring.

In the simulated system, each of the three algorithms i.e., Altered RED, BLUE and ECN are implemented in the nodes where there is a high probability of congestion occurrence. There is a continuous transmission of data through the rings. Each of the three algorithms will be triggered at different stages of congestion periods. Using the advanced functioning of the simulator, graphs are obtained which is an apt tool for comparison of the algorithms.

5.2 Implementation of Altered RED

The Altered RED implemented in the server connected to the bridge on ring1 contains a series of instructions. RED_START, RED_PROCESS, RED_END are the instructions which are executed by the processing element Nip 5.A message list size of 500 bits is allocated for the Altered RED queue. The limitation cycle time is set as 23 microseconds. The Time slice used to transfer data between the bridge and the ring1is set as "exponential iteration period". The exponential iteration period means, doubling the transmission of the data each time the data is sent successfully. This creates a persistent state of congestion at the server. Keep block separate is an option used to break the packets into queues rather than use them as one single block.

For each of the servers and clients several modules are defined. The modules are the units of execution. For example when the client has data to send, it initiates the module Start A Message by executing the corresponding instructions available in this module, the client communicates to the server the required data. Under preset conditions it is decided that the different clients will start communicating to their respective servers in a random fashion. The traffic around the bottleneck server Nip5 is built slowly, with persistent flow of data from the servers Nip1 and Nip4.

With persistent queuing at the server Nip5, when the exponential weighted moving average computed exceeds the maximum threshold, the Altered RED algorithm intervenes. It determines the bandwidth of the link, and

breaks the single queue into segments of packets. These packets use the maximum bandwidth by transiting in a parallel manner through the link. In order to achieve high performance these packets are required to travel in a high speed at the same level.

With effective control of the traffic, the congestion is cleared and thereafter the Altered RED takes preventive measures for avoiding another congestion situation. It continuously transmits the incoming packets in the parallel manner thus utilizing the maximum available bandwidth possible. This avoids the periods of low utilization of the link, which was prevalent in the original RED algorithm.

In order to justify and establish the performance of the Altered RED, comparative studies are made between the Altered RED and BLUE, Altered RED and ECN. The main parameters of comparison are Average Link utilization and Packet loss reduction. From the comparative studies, it is determined that the Altered RED functions in equivalence to BLUE and ECN. A deeper analysis provides the conclusion that the original RED with very slight modifications can behave in equal to BLUE and ECN.

5.3 Implementation of BLUE

The BLUE algorithm is implemented in the server connected to the bridge on ring2. It consists of a series of instructions like BLUE_START, BLUE_PROCESS, and BLUE_END. The blue algorithm starts functioning once the congestion is detected and takes measures to control the congestion.

5.3.1 The algorithm

The key idea behind BLUE is to perform queue management based directly on packet loss and link utilization rather than on the instantaneous or average queue lengths. This is in contrast to all known active queue a management scheme, which use some form of queue occupancy in their congestion management. BLUE maintains a single probability, *Pm* that, it uses to mark (or drop) packets when they are enqueued. If the queue is continually dropping packets due to buffer overflow, BLUE increments *Pm*, thus increasing the rate at which it sends back congestion notification. Conversely, if the queue becomes empty or if the link is idle, BLUE decreases its marking probability. This effectively allows BLUE to "learn" the correct rate it needs to send back congestion notification. In the algorithm the marking probability is updated when the queue length exceeds a certain value. This modification allows room to be left in the queue for transient bursts and allows the queue to control queuing delay when the size of the queue being used is large. Besides the marking probability, BLUE uses two other parameters which control how quickly the marking probability changes over time. The first is *freeze time*. This parameter determines the minimum time interval between two successive updates of

Pm. This allows the changes in the marking probability to take effect before the value is updated again. The *freeze time* value should be randomized in order to avoid global synchronization .The other parameters used, (d1 and d2), determine the amount by which *Pm* is incremented when the queue overflows or is decremented when the link is idle. Usually, d1 is set significantly larger than d2. This is because link underutilization can occur when congestion management is either too conservative or too aggressive, but packet loss occurs only when congestion management is too conservative. By weighting heavily against packet loss, BLUE can quickly react to a substantial increase in traffic load

Upon packet loss (or Qlen > L) event:
if ((now - last update) > *freeze time*) then
Pm = *Pm* + d1
last update = now
Upon link idle event:
if ((now - last update) > *freeze time*) then
Pm = *Pm* - d2
last update = now

Figure 4: BLUE Algorithm

5.3.2 The Parameters

The main parameters of BLUE are

Freeze time	Update frequency, could be randomized to avoid global synchronization
d1	Increment step
d2	Decrement step
Packet marking Probability	pm

Table 2: Parameters of BLUE

It is interesting to note that as queue weight gets smaller, the impact of queue length on RED's congestion management algorithm gets smaller. For extremely small values of queue weight, RED's algorithm becomes decoupled from the queue length and thus acts more like BLUE.

5.4 Implementation of ECN

The Explicit congestion notification scheme is implemented in the ring2 of the network. The ECN is a bit–based congestion control mechanism intending to avoid packet drops of AQM algorithms before the congestion. ECN is a highly recommended mechanism because of the many advantages it provides and can easily be supported by RED. When the average queue size is below the upper threshold value, instead of a router dropping a packet that is decided after the calculations of RED, it marks some bits on the IP header for notifying the sender about the potential congestion in the future. The sender then does not need to retransmit any packets. However, the router drops all the incoming packets when the average queue size is larger than the upper threshold without considering ECN.

5.4.1 ECN –TCP Header

ECN-TCP makes use of two bits in the header: one to indicate that the connection is ECN capable (*ECT-bit*) and one to indicate congestion (*CE-bit*). If a connection uses ECN for congestion control, the ECT-bit is set to 1 in all packets; otherwise it is set to 0. The CE-bit is set at a congested router. Furthermore, ECN introduces two new flags in the reserved field of the TCP header. The first flag is the *ECN echo flag*, which is set in an ACK by the receiver if an ECN packet has the CE-bit set. The second flag is the *congestion window reduced (CWR) flag*. The sender sets this flag after it has reduced it congestion window for any reason (i.e. retransmission timer timeouts, duplicate ACKs, and ECN echo ACK).

5.4.2 ECN-TCP Sender

In the connection setup phase, the sender and receiver need to negotiate their ECN capability. This is done by the sender setting the CWR and ECN echo flags in the SYN packet and by the receiver setting the ECN echo flag in the SYN-ACK packet. In the case that just the sender is ECN capable, the connection should not make use of the ECT and CE bit. Otherwise, the ECT bit should be set by the sender in every packet and must react to ECN echo ACKs. The sender should treat an ECN echo ACK as a lost packet, but without the need to retransmit the marked packet. Although this ACK acknowledges a data packet, it does not increase the congestion window. If the sender receives multiple congestion indications, including timeout, duplicate ACKs and ECN echo ACK, it should react just once per RTT to the congestion indication. In the case that a retransmitted packet is marked or dropped, the sender should react to congestion indication again. However, it should not reduce the slow start threshold ssthresh if it has been reduced within the last RTT. After the sender responds to a congestion indication, it sets the CWR flag in the next data packet sent after the reduction of the window.

5.4.3 ECN-TCP Receiver

The receiver echoes the congestion indication of a CE packet back to the sender. After it has received a CE packet, it sets the ECN echo flag of the subsequent ACK packet. The receiver continues to set the echo flag until it receives a data packet with the CWR flag set. This provides robustness in case an ACK packet with the echo flag set gets lost.

Future Enhancements

6. FUTURE ENHANCEMENTS

Congestion in the Internet and other high frequency networks is a persistent threat to data transmission. The solution to this problem is to implement congestion control algorithms in bottleneck routers. Each of these congestion control algorithms focuses on ways and means of reducing the packet loss and enabling loss free transmission of data In the proposed Altered RED there is a significant improvement in reducing the packet loss when compared to that of original RED.

There are many areas for further enhancements on Altered RED. Some of the important enhancement suggestions are

- Determination of the optimum average queue size for maximizing throughput and minimizing delay for various network configurations. This determination is heavily dependent of the characterization of the network traffic as well as on the physical characteristics of the network.
- The traffic dynamics with a mix of Drop Tail and Altered RED gateways, as would result from partial deployment of Altered RED gateways in the current Internet.
- The behavior of the RED gateway machinery with transport protocols other than TCP, including open- or closed-loop rate-based protocols.
- The current implementation of the Altered RED in NII.5 does not indicate if the queue size should be measured in bytes or packets.

For networks with a range of packet sizes at the congested gateway the difference can be significant. This includes networks with two-way traffic where the queue at the congested gateway contains large FTP packets, small TELNET packets, and small control packets. For a network where the time required to transmit a packet is proportional to the size of the packet, and the gateway queue is measured in bytes, the queue size reflects the delay in seconds for a packet arriving at the gateway.

Conclusion

7. CONCLUSION

The Active Queue Management Algorithms Altered RED, BLUE and ECN are implemented using Nii.5 and the simulation results are obtained. The main parameter that is used as a comparison between the algorithms is the average link utilization. Some modifications have been made to the original RED in order to make its performance i.e. the average link utilization in equivalence to the prominent Active Queue Management Algorithms BLUE and ECN.

In the Active queue management algorithm RED, the length of the single queue is used as the indicator. This length does not provide information of the severity of congestion i.e. which source is transmitting the maximum amount of data. This results in unfair share of bandwidth. Also RED does not avoid packet loss in extreme cases of congestion. In the implementation of Altered RED, the single queue is segmented into packets that are made to use the maximum bandwidth of the link.

Thus from the simulation results and analysis we conclude that Altered RED achieves a reduction in the number of packet losses, maximum link utilization and also works in equivalence to BLUE and ECN.

References

8. REFERENCES

1. Victor Firoiu and Marty Borden "A study of Active Queue Management for Congestion Control".
2. Wu-chang Feng, *Member, ACM*, Kang G. Shin, *Fellow, IEEE and ACM*, Dilip D.Kandlur "The BLUE Active Queue Management Algorithm "
3. Chris Chen, Hariharan Krishnan, Steven Leung, Nelson Tang " Implementing Explicit Congestion Notification (ECN) in TCP"
4. Tilo Hamann and Jean Walrand "A New Fair Window Algorithm for ECN Capable TCP"
5. Prasad Bagal, Shivkumar Kalyanaraman, Bob Packer " Comparitive study of RED, ECN and TCP rate control "
6. Sally Floyd and Van Jacobson "Random Early Detection Gateways for Congestion Avoidance "
7. Chunlei Liu and Raj Jain " Improving Explicit Congestion Notification"
8. Haining Wang and Kang.G.Shin " Refined Design of RED Gateways".
9. Simulation of congestion control and avoidance algorithms for TCP/IP Networks by D.SRINIVASA RAO and Dr.K.C.REDDY.
10. "An End-to-end Congestion Control Mechanism For Internet "by Young Gook Kim,You Jun Chung,Jong Won Kim .

11 Fairness and Stability of Congestion Control Mechanisms of TCP" by the authors G.Hasegawa ,Masayuki Murata, and Hideo Miyahara, Department of Infomatics and Mathematical Science.

12. Andrew S.Tanenbaum, "Computer Networks", Prentice Hall,Inc .

13. Behrouz A.Forouzan, "TCP/IP Protocol suite ", Tata Mc Graw Hill Edition, 1999.

14.. Network II.5 User's Manual , Release 12, November 1997.

15. Douglas E.Comer " Internetworking with TCP/IP ".

16. www.caciasl.com

Appendices

APPENDIX 1

SAMPLE CODE:

```
PROC IML;
USE TWO VAR{a,b,c};
READ ALL INTO REGDAT;
ID = REGDAT[,1];
Y  = REGDAT[,2];
X  = REGDAT[,3:6];
YNAME  = 'HamD';
XNAMES = {'a' 'b' 'c'};
NTOT = NROW(ID);
PRINT NTOT;
N = 1;
IDOLD = 0;
COUNT = 0;
NITEMP = J(NTOT,1,0);
DO I = 1 TO NTOT;
  IDTEMP = ID[I];
  IF IDOLD = 0  THEN IDOLD = IDTEMP;
  IF IDTEMP = IDOLD THEN COUNT = COUNT + 1;
  ELSE
      DO;
       NITEMP[N] = COUNT;
       N = N + 1;
       COUNT = 1;
```

```
        IDOLD = IDTEMP;
      END;
  IF I = NTOT THEN NITEMP[N] = COUNT;
END;

NI = J(N,1,0);
DO K = 1 TO N;
   NI[K] = NITEMP[K];
END;
TRANSNI = T(NI);

BETA = INV(X` * X) * (X`* Y);
YHAT = X * BETA;
RESID = Y - YHAT;
SSE = RESID` * RESID;
ERVAR = (1.0/NTOT)  #  SSE;
CLUSTVAR = 0.10 # ERVAR;


MAXCORR=1;
CONVERGE = 0.0001;
IT = 0;
LN2PI = LOG( 2 * 3.141592654);
P = NCOL(X);
VARDER = J( 2,1,0);
VARINF = J( 2,2,0);
ALPHA = J(N,1,0);
```

```
ALPHAVAR = J(N,1,0);

DO WHILE ( MAXCORR > CONVERGE);

  IT = IT + 1;
  U = Y - X * BETA;

/*  COMPUTE  EAT  ESTIMATES  */

  IS = 0;  IE = 0;
  DO  I =  1  TO  N;
     NII = NI [I] ;
     IS =  IE + 1;
     IE =  IS - 1 + NII;
     SB = (INTCLASS # NII) / ( 1 + (NII - 1) # INTCLASS ) ;
     V =  SUM (U [IS : IE,]) ;
     ALPHA [I] = (SB / NII) # V ;
     ALPHAVAR [I] = (1.0 - SB) # CLUSTVAR ;
  END;



  I=1;
  I2=0;
  DO   J = 1  TO NTOT;
      I2 = I2 + 1;
      U[J] = U[J] - ALPHA[I];
```

```
      IF (J ^=  NTOT & I2 = NI[I])  THEN
         DO;
 I2 = 0;  I = I + 1;
         END;
  END;

/*  COMPUTE  THE  FIRST  DERIVATIVES  */

  XX = T(X);
  BETADER = (1.0 / ERVAR) # (XX * U) ;
  SSA= ALPHA` * ALPHA ;
  SUMALPHA = SUM(ALPHAVAR) ;
  VARDER[1] = .5 # (1.0 / CLUSTVAR) # (SSA + SUMALPHA - N #
CLUSTVAR);
  SSU = U` * U;
  VARDER[2] = .5 # (1.0/ ERVAR) # (SSU + (TRANSNI * ALPHAVAR)
- (NTOT
         #  ERVAR));


  IS=0;
  IE=0;
  BETAINF = J(P,P,0);
  DO  K = 1 TO N;
      NII = NI [K] ;
      IS = IE + 1 ;
      IE = IS - 1 + NII;
```

```
      XTEMP = X [ IS : IE ,] ;
      ONEVEC = J(NII,1,1);
      SSXT = XTEMP` * XTEMP;
      TONE = T(ONEVEC) * XTEMP;
      SSO = TONE` * TONE;
      BETAINF = BETAINF + (ERVAR # SSXT) - (ALPHAVAR[K] #
SSO);
  END;

      BETAINF = (1.0 / (ERVAR * ERVAR)) # BETAINF;
      DIFVAR = CLUSTVAR - ALPHAVAR;
      SSVAR = DIFVAR` * DIFVAR;
      VARINF[1,1] = 0.5 # (1.0/ ( CLUSTVAR ** 2)) # SSVAR;
      VARINF[2,2] = 0.5 # (1.0/ ( ERVAR ** 2)) # ( TRANSNI *
   (ERVAR - ALPHAVAR) ## 2);
      VARINF[2,1] = 0.5 # (1.0 / ERVAR) # (1.0 / CLUSTVAR ) #
        ( TRANSNI * (ALPHAVAR ## 2 ));
      VARINF[1,2] = VARINF[2,1];

/*  COMPUTE THE CORRECTIONS */

      BETACORR = INV(BETAINF) * BETADER;
      VARCORR = INV(VARINF) * VARDER;
  /*    PRINT ' BETA  CORRECTIONS ';
      PRINT BETACORR;
      PRINT ' VARIANCE CORRECTIONS ';
      PRINT VARCORR;    */
```

```
      BETA = BETA +  BETACORR;

      PI = LOG(CLUSTVAR) + VARCORR[1] ;
      CLUSTVAR = EXP(PI);
      IF CLUSTVAR < 0.0000001 THEN CLUSTVAR = 0.0000001;
      TAU = LOG(ERVAR) + VARCORR[2];
      ERVAR = EXP (TAU);

/*    PRINT  ' REGRESSION COEFFICIENTS ';
      PRINT  BETA;
      PRINT 'ERROR  VARIANCE ';
      PRINT ERVAR;
      PRINT '  CLUSTER  VARIANCE';
      PRINT  CLUSTVAR; */

      LOGLIK = - 0.5 # ( SUM(NI) # (LN2PI + LOG(ERVAR)) +
    N # LOG(CLUSTVAR)  -  SUM(LOG(ALPHAVAR))  +
   ((ALPHA` * ALPHA) / CLUSTVAR) +  ( U` * U) /
      ERVAR ) ;
/*    PRINT  ' LOG-LIKELIHOOD  ';
      PRINT LOGLIK;  */

      MAXCORRB = MAX(ABS(BETACORR));
      MAXCORRV = ABS(VARCORR[2]);
      IF  (  CLUSTVAR  >  0.0000001  &  ABS(VARCORR[1])  >
ABS(VARCORR[2]))
```

```
        THEN MAXCORRV = ABS(VARCORR[1]) ;
    MAXCORR = MAXCORRV;
    IF(MAXCORRB > MAXCORRV) THEN MAXCORR =
MAXCORRB;
END;

/* COMPUTE STANDARD ERRORS, Z STATISTICS, AND P-VALUES
FOR ALL
  PARAMETERS AND PUT THESE VECTORS AS COLUMN
VECTORS IN BETARES
  & VARRES MATRICES */

    BETARES = J(P,4,0);
    BETARES[1:P,1] = BETA;
    BETARES[1:P,2] = SQRT(VECDIAG(INV(BETAINF)));
    BETARES[1:P,3] = BETARES[1:P,1] / BETARES[1:P,2];
    BETARES[1:P,4] = 2 * PROBNORM( - ABS(BETARES[1:P,3]));


/*  THE INFORMATION MATRIX NEEDS TO BE PUT IN TERMS OF
THE RAW
   PARAMETERS  */

    VARRES = J(2,4,0) ;
    VARRES[1,1] = CLUSTVAR;
    VARRES[2,1] = ERVAR;
    VARINF[1,1] = VARINF[1,1] / (CLUSTVAR * CLUSTVAR);
```

```
      VARINF[2,2] = VARINF[2,2] / (ERVAR * ERVAR) ;
      VARINF[1,2] = VARINF[1,2] / (ERVAR * CLUSTVAR);
      VARINF[2,1] = VARINF[2,1] / (ERVAR * CLUSTVAR) ;

      VARRES[1:2,2] = SQRT(VECDIAG(INV(VARINF)));
      VARRES[1:2,3] = VARRES[1:2,1] / VARRES[1:2,2] ;
      VARRES[1:2,4] = PROBNORM ( - ABS(VARRES[1:2,3]));

      INTCLASS =  CLUSTVAR / (CLUSTVAR + ERVAR) ;

MATTRIB  BETARES ROWNAME = (XNAMES)
  COLNAME = ({ESTIMATE   SE   Z  'P-VAL(2)'})
  LABEL   = ' VARIABLE  '
  FORMAT  = 12.6;
PRINT BETARES;

MATTRIB VARRES ROWNAME = ({CLUSTER   ERROR})
 COLNAME =  ({ESTIMATE  SE  Z  'P-VAL(1)'})
 LABEL   = '         '
 FORMAT  = 12.6;

      ALPHASD = SQRT(ALPHAVAR);
```

APPENDIX 2

SAMPLE REPORT

CACI NETWORK II.5 RELEASE 12.10 03/27/2005 04:26:46
Page 17

A Sample Store and Forward Model

Transfer Device Utilization Statistics

from 0. to 2. Seconds
(All times reported in microseconds)

Transfer Device Name	TD 3.2.1	NIP 2 TO N3.1	TD 6
Transfer Requests Granted	10,577	36	0
Requests Interrupted	0	0	0
avg Request Delay	.055	0.	0.
max Request Delay	113.456	0.	0.
std dev Delay	2.347	0.	0.
Interrupted Transfers	0	0	0
avg Interrupt Time	0.	0.	0.
max Interrupt Time	0.	0.	0.
std dev Time	0.	0.	0.
Completed Transfers	10,576	36	0
avg Usage Time	122.814	0.	0.
max Usage Time	123.000	0.	0.
min Usage Time	0.	0.	0.
std dev Time	4.781	0.	0.
avg Queue Size	.000	0.	0.
max Queue Size	1	1	0
std dev Size	.017	0.	0.

Lost Transfers			
Transmit Error	0	0	0
Queue Flag	0	0	0
Buffer Overflow	0	0	0
Throughput			
Data Bits/Second	649,469.000	0.	0.
Total Bits/Second	649,469.000	0.	0.
Transfers/Second	5,288.000	18.000	0.
Per Cent Utilization	64.947	0.	0.

A Sample Store and Forward Model

S n a p s h o t R e p o r t

at 2. seconds

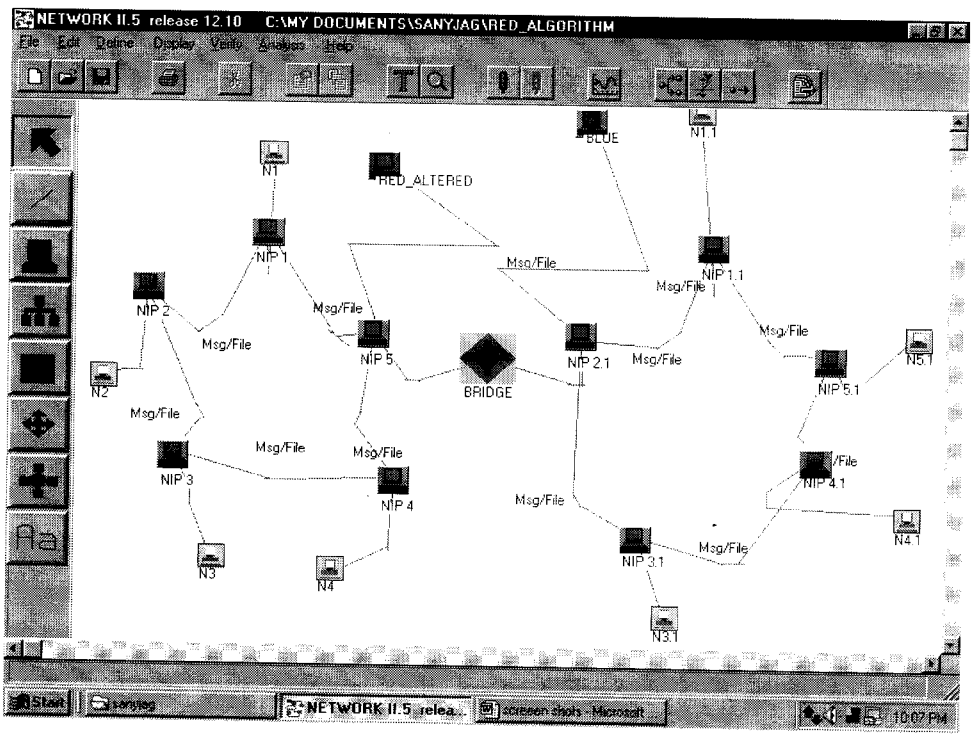
pe N1 is BUSY executing instruction START A MESSAGE
resident module is START A MESSAGE (400133)
it holds transfer device NIP 1 TO N1
pe NIP 1 is BUSY executing instruction INITIATE MESSAGE ON RING
resident module is START A NEW MESSAGE ON RING (59597)
it holds transfer device TD 1
IT'S ONE THAT I SENT (8) is in the pe local queue
IT'S ONE THAT I SENT (9) is in the pe local queue
IT'S ONE THAT I SENT (10) is in the pe local queue
IT'S NOT FOR ME (23644) is in the pe local queue
IT'S NOT FOR ME (23670) is in the pe local queue
IT'S FOR ME (NIP 1) (36) is in the pe local queue
IT'S NOT FOR ME (23678) is in the pe local queue
IT'S NOT FOR ME (23679) is in the pe local queue
pe NIP 5 is IDLE
pe BRIDGE is IDLE
pe NIP 4 is BUSY executing instruction INITIATE MESSAGE ON RING
resident module is START A NEW MESSAGE ON RING (59600)
it holds transfer device TD 4
START A NEW MESSAGE ON RING (60298) is in the pe local queue
START A NEW MESSAGE ON RING (60302) is in the pe local queue
pe NIP 3 is BUSY executing instruction INITIATE MESSAGE ON RING
resident module is START A NEW MESSAGE ON RING (60299)
it holds transfer device TD 3
START A NEW MESSAGE ON RING (60303) is in the pe local queue
pe NIP 2 is IDLE
pe N4 is IDLE
pe N3 is IDLE
pe N2 is BUSY executing instruction START A MESSAGE

resident module is START A MESSAGE (400082)
it holds transfer device NIP 2 TO N3
START A MESSAGE (400130) is in the pe local queue
START A MESSAGE (400152) is in the pe local queue
pe N1.1 is BUSY executing instruction START A MESSAGE
resident module is START A MESSAGE (400124)
it holds transfer device NIP 1 TO N1.1
START A MESSAGE (400125) is in the pe local queue
START A MESSAGE (400155) is in the pe local queue
pe NIP 1.1 is IDLE
pe NIP 5.1 is BUSY executing instruction INITIATE MESSAGE ON RING
resident module is START A NEW MESSAGE ON RING (59596)
it holds transfer device TD 5.1
START A NEW MESSAGE ON RING (59602) is in the pe local queue
START A NEW MESSAGE ON RING (60300) is in the pe local queue
IT'S NOT FOR ME (23653) is in the pe local queue
IT'S NOT FOR ME (23684) is in the pe local queue

APPENDIX 3

Sample Results

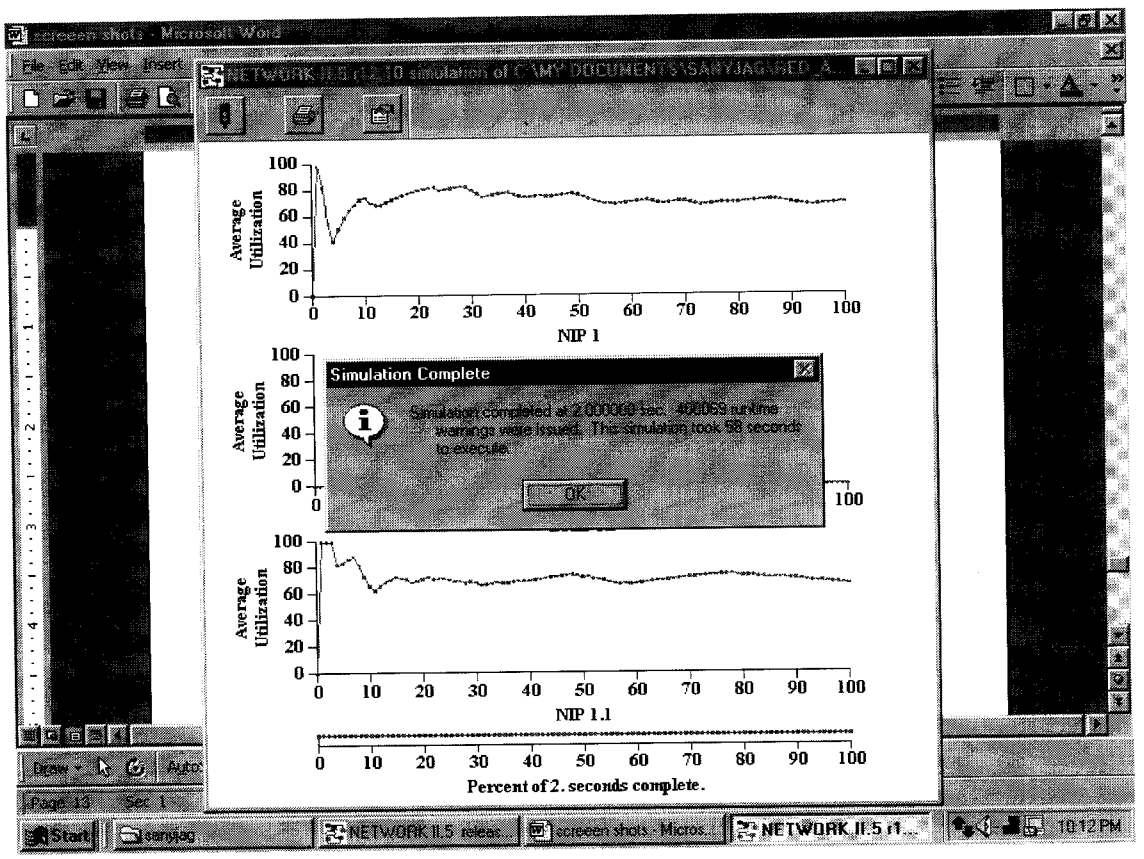
Topology



This is the network topology wherein the two algorithms Altered RED and BLUE are implemented in Nii.5 There are two rings each connected to the bridge. The algorithms are implemented on either side of the rings in order to show a comparison between them. The algorithms are implemented in the server that is connected to the bridge where the chances of congestion occurrences are more.

Comparison between Altered RED and BLUE:

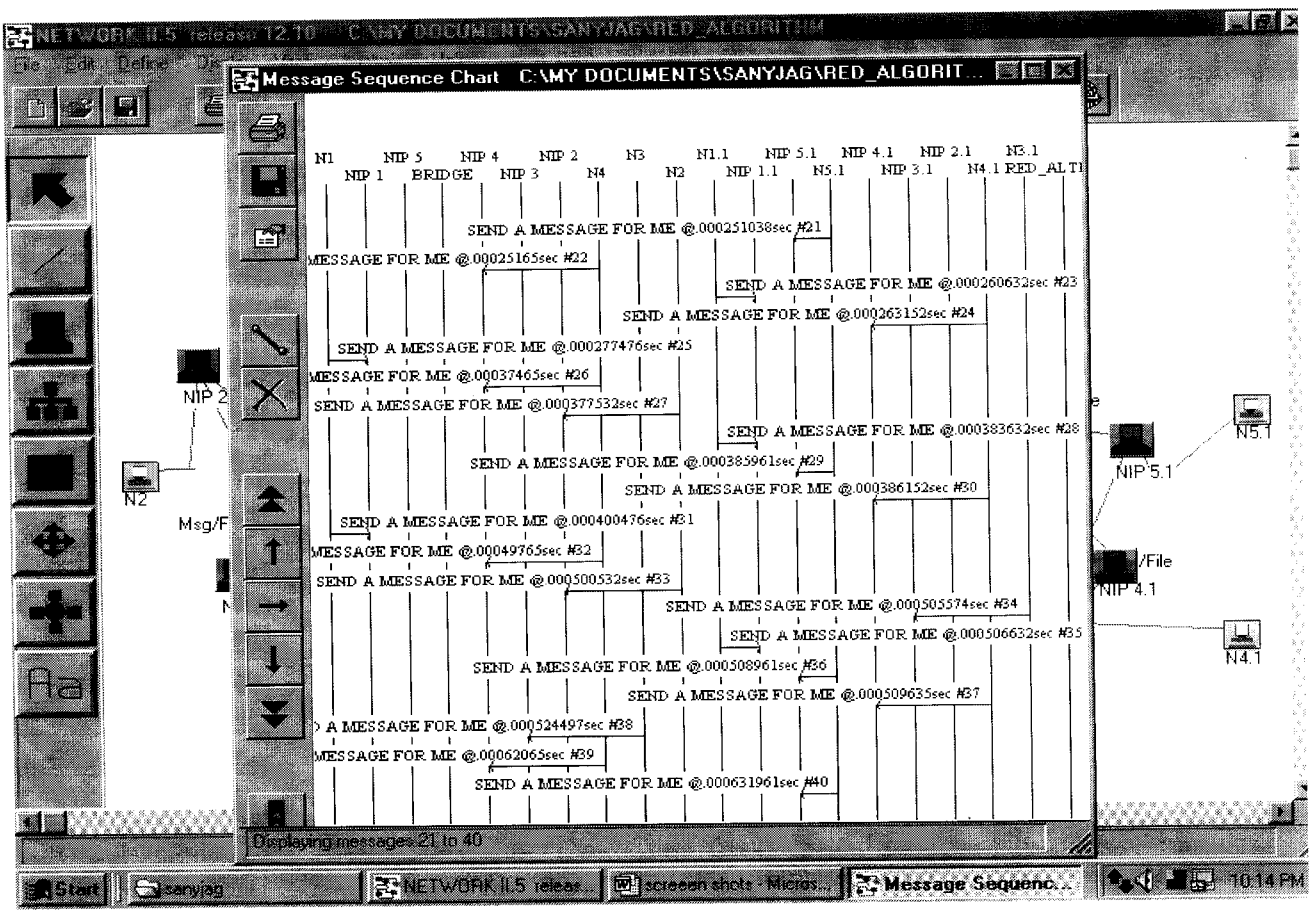
This is the simulation result obtained. Here the main parameter in comparison is the Average Utilization.



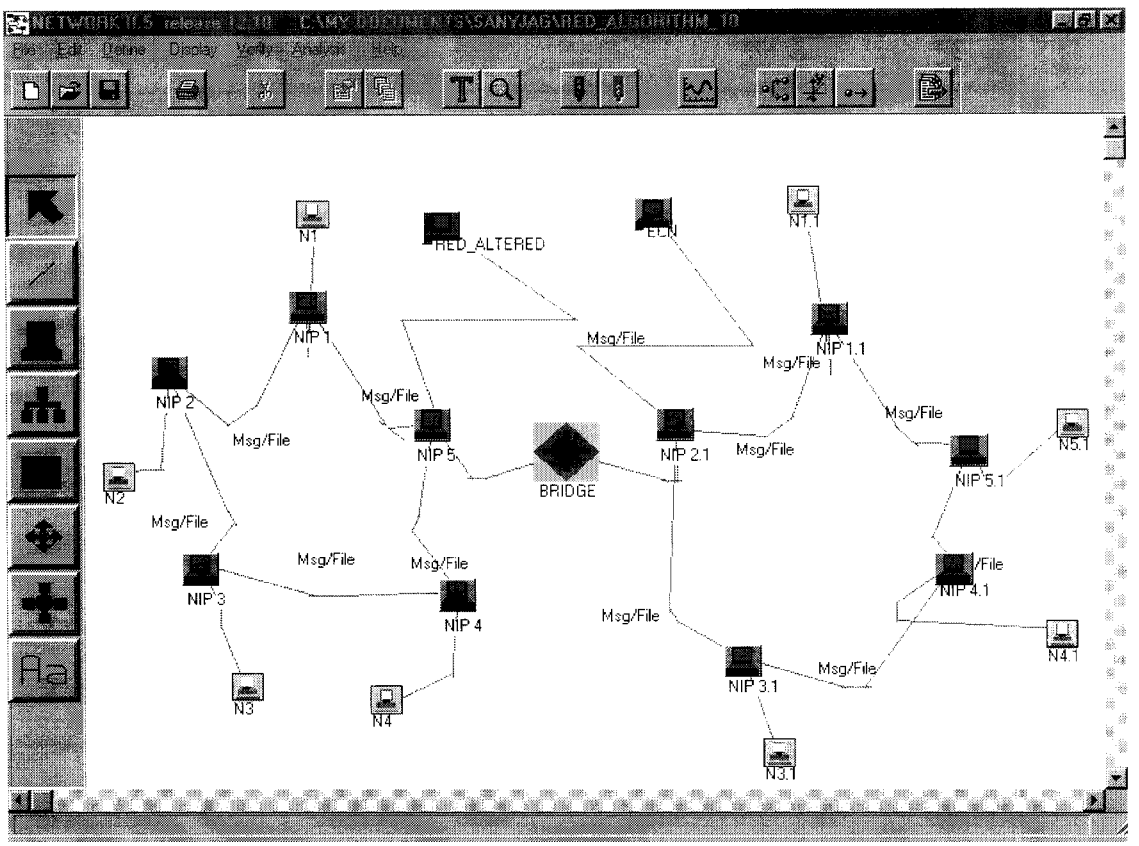
This plot depicts the transfer of data from NIP 1 to NIP 1.1. As the amount bytes being transferred increases, the curve in Nip1 and Nip1.1 also increase but the sudden dip represents the triggering of the algorithms Altered RED in Nip1 and BLUE in Nip1.1.After the triggering, both the plots reach a constant and equal value. Thus from the graph obtained, we can conclude that the performance of Altered RED is Equivalent to BLUE

Message Sequence chart:

The Message Sequence Chart depicts the transfer of messages form one processing element to another during the simulation run. It also shows the various time instances when the message was generated.

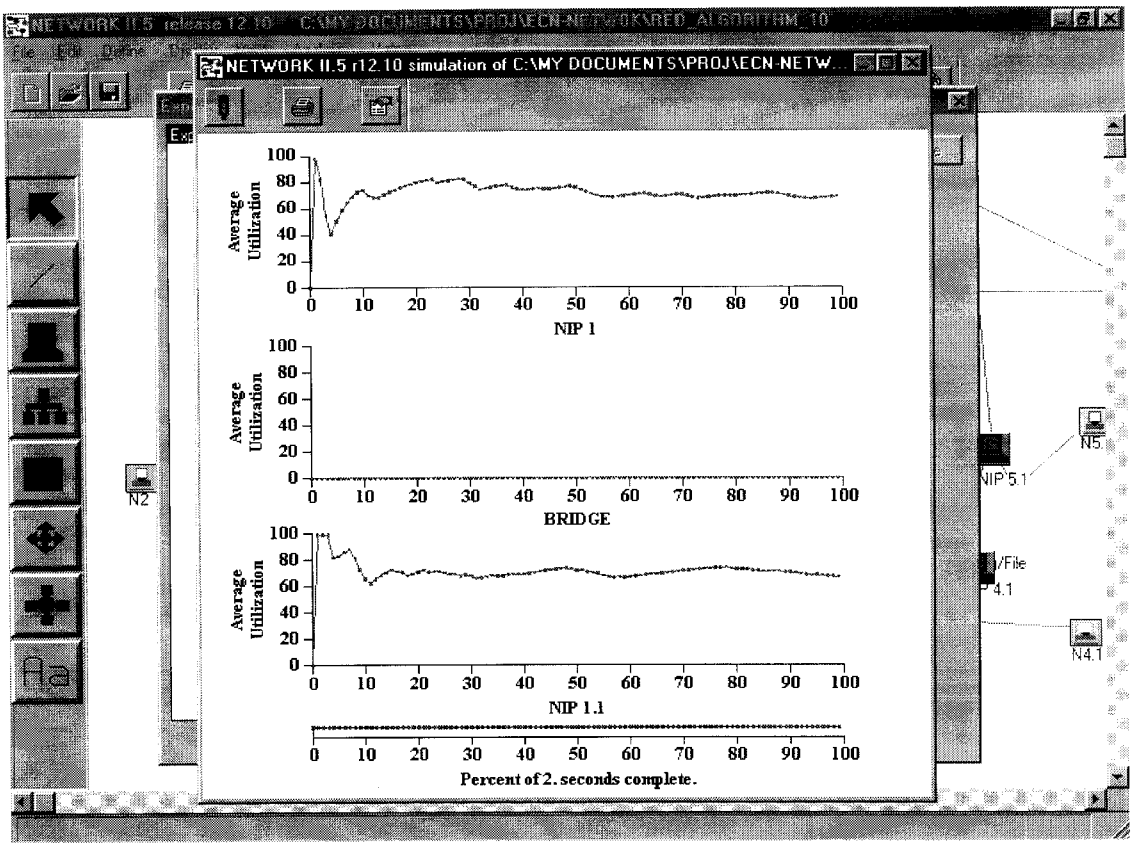


The Network with Altered RED and ECN



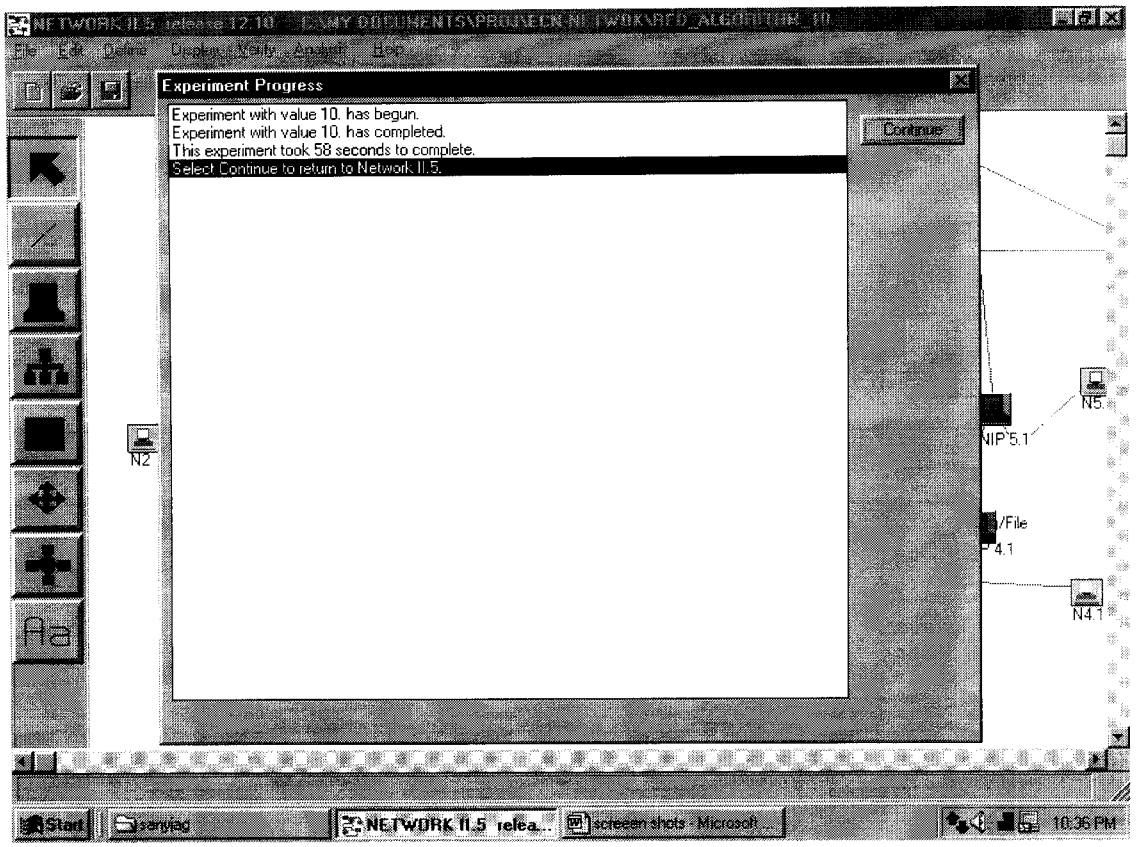
In this network the algorithms Altered RED and ECN are implemented in the servers connected to the bridge.

Comparison between Altered RED and ECN



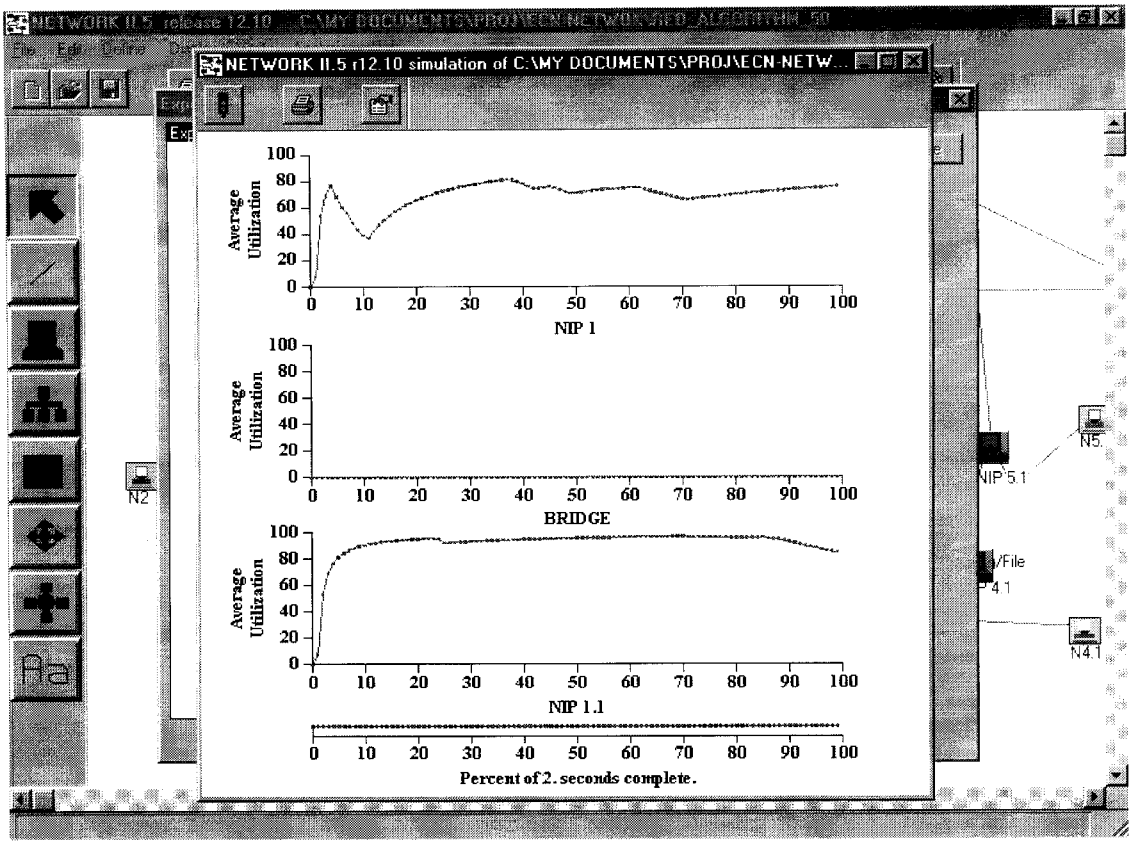
The above graph depicts the simulation run for transferring 10 bytes of data from Nip1 to Nip1.1. When the transfer is in the beginning stages, there is a sudden increase in the graph. Later when the algorithms Altered RED and ECN are triggered, there is a sharp dip after which they attain an almost constant and equal value. From the above graph it is visible that the Altered RED implemented functions almost in equal to that of ECN.

Simulation Report:



This simulation result depicts the time that has taken to transfer 10 bytes of data and from this we can view that it has taken 58 seconds to transfer 10 bytes of data.

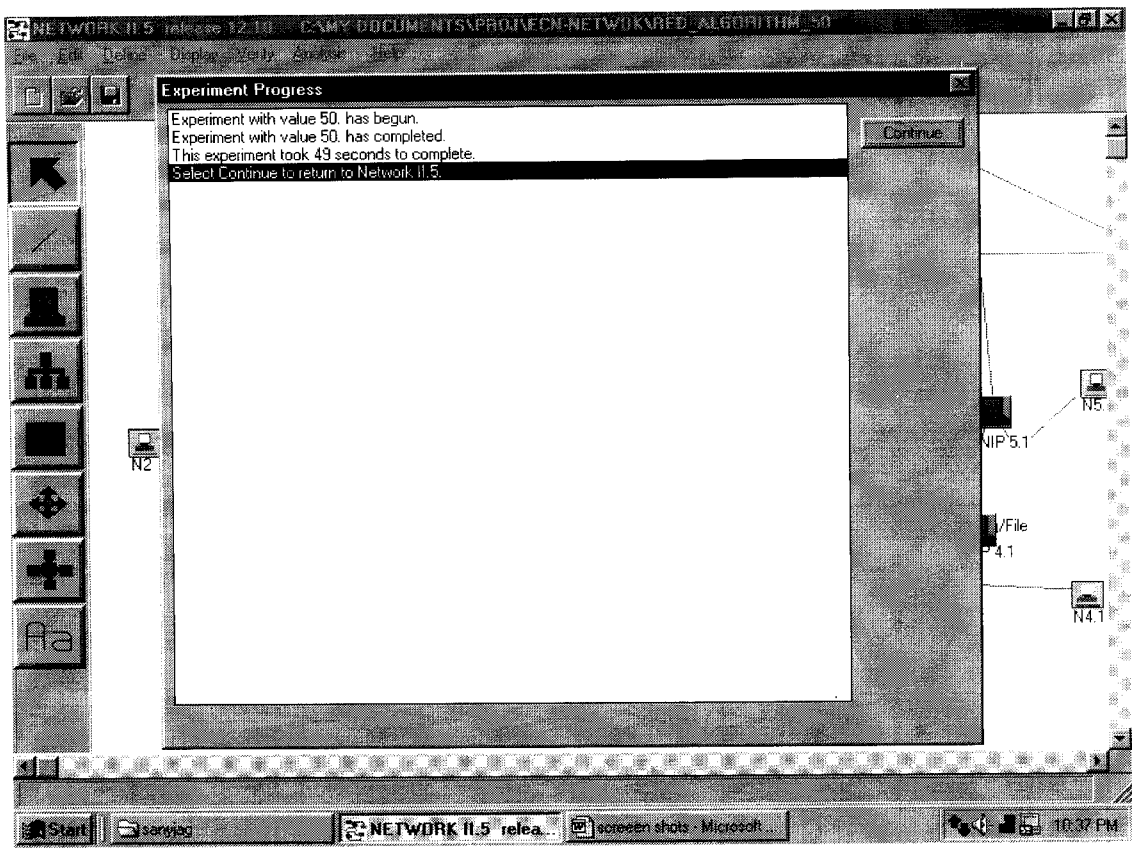
This screenshot depicts the simulation result for sending 50 bytes of data.



Nip1: Altered RED is implemented.
Nip1.1: ECN is implemented.

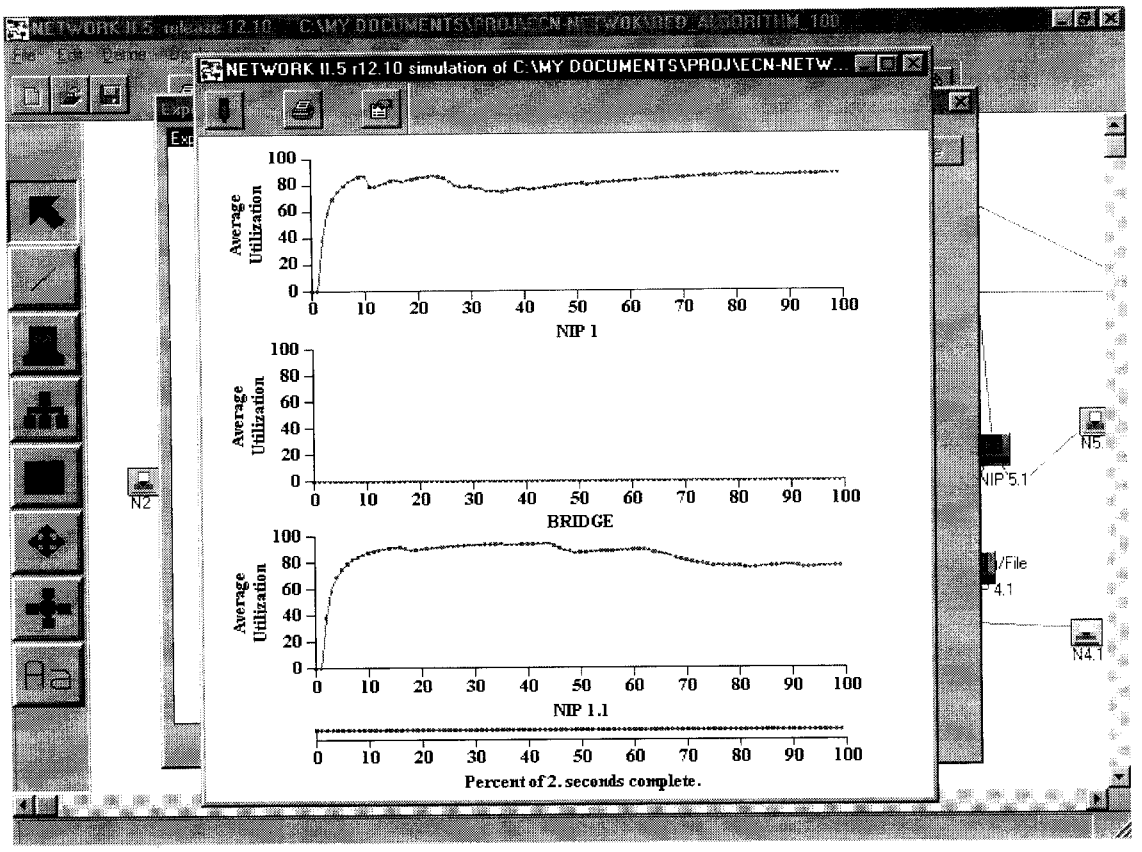
Even for transferring 50 bytes of data, the Altered RED and ECN achieve an almost equal value after the initial triggering action.

Simulation Report:



In this the time that has taken to transfer 50 bytes of data is only 49 seconds. There is a profound decrease in the time taken to transfer 50 bytes than to transfer 10 bytes. this shows that the algorithms have been triggered and with the increase in data congestion has occurred but been effectively controlled.

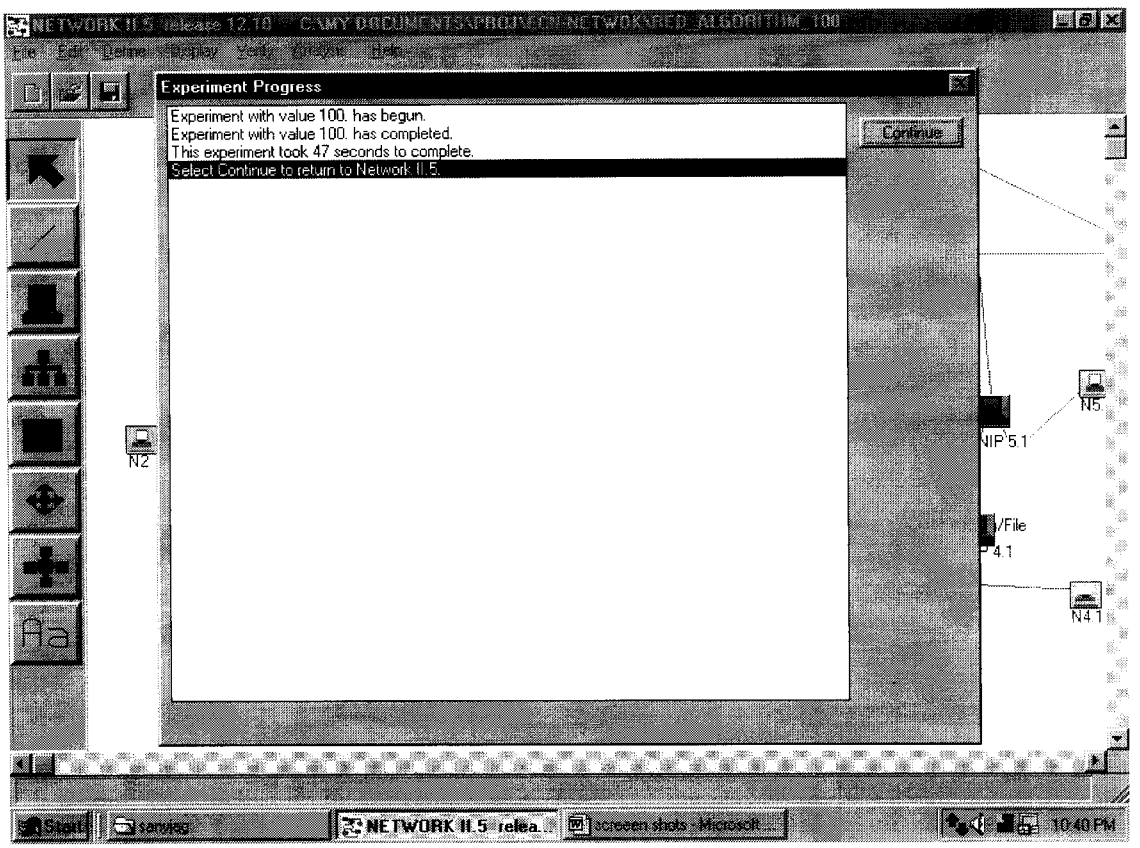
This screenshot depicts the simulation result for sending 100 bytes of data



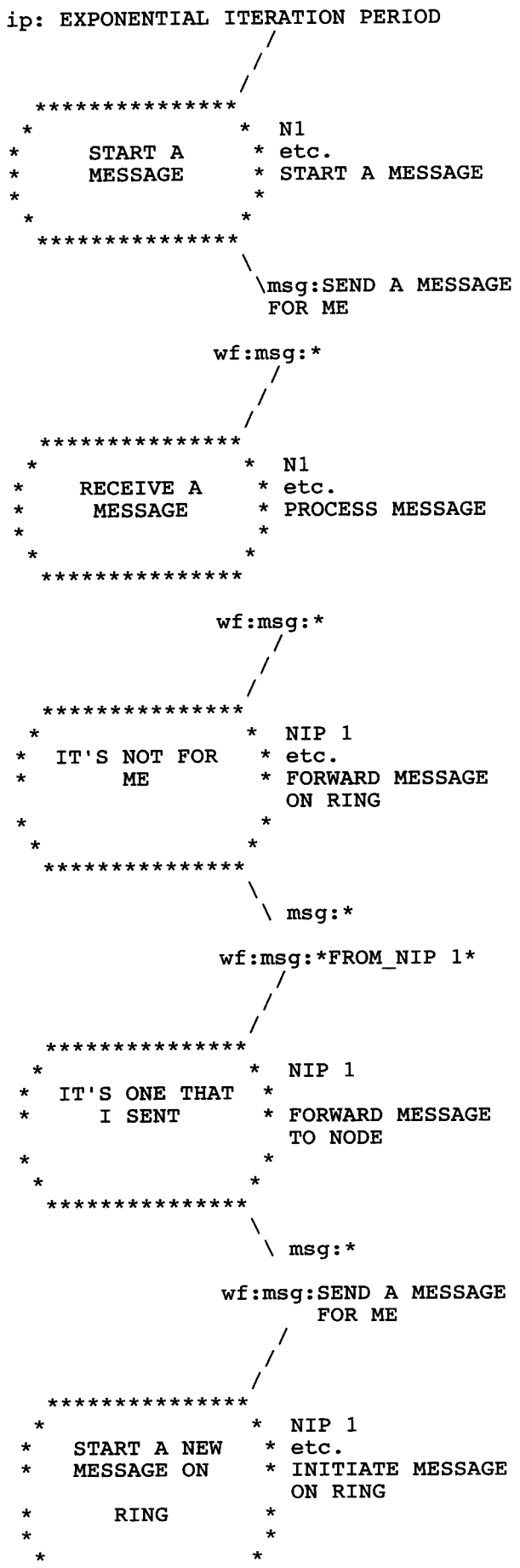
Nip1: Altered RED is implemented
Nip1.1: ECN is implemented

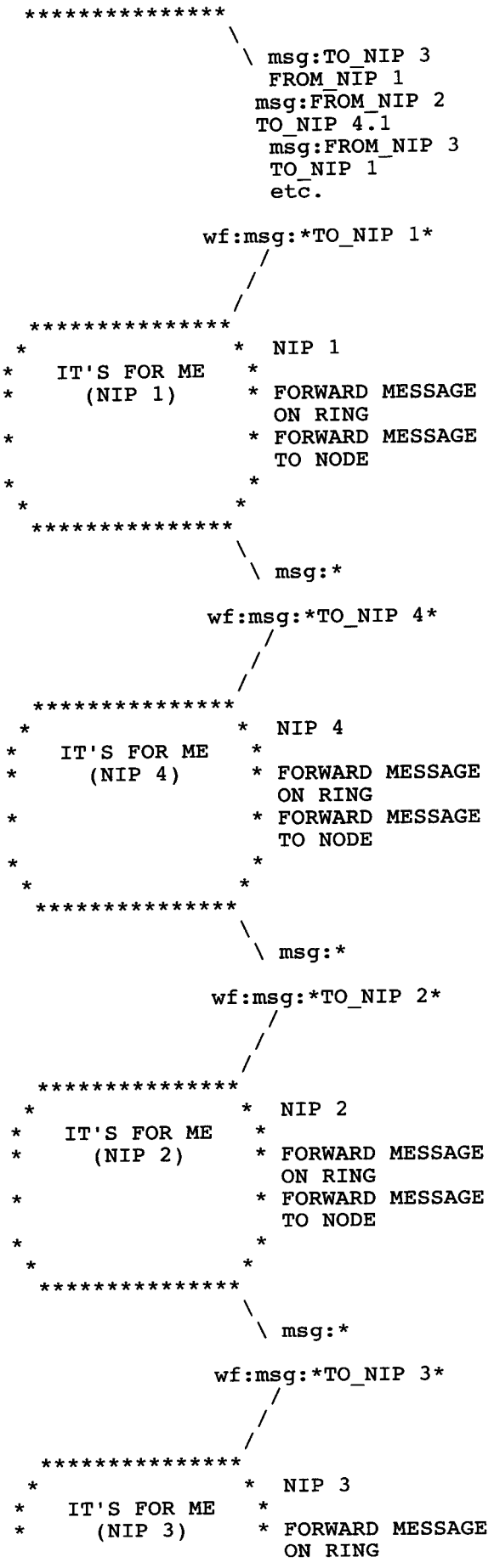
With a major increase in the bytes of transfer, still Altered RED and ECN are functioning in an almost equal manner. From the above graph, it is seen that both are attaining almost equal and constant values.

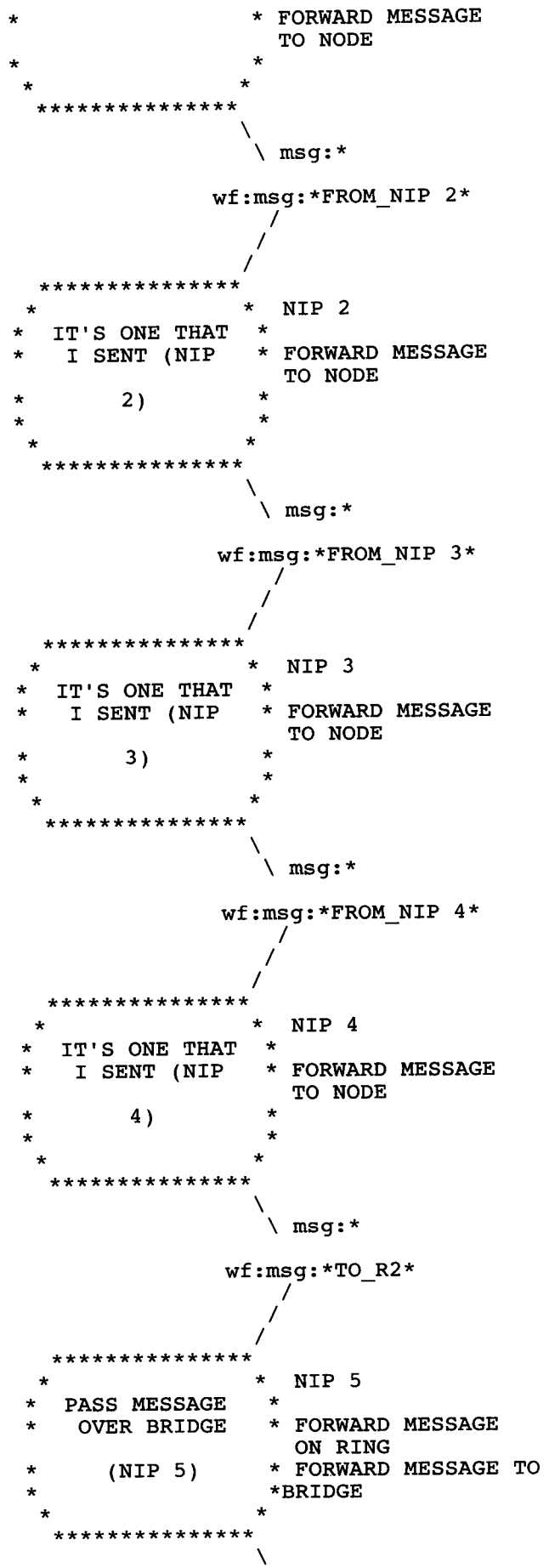
Simulation Report:

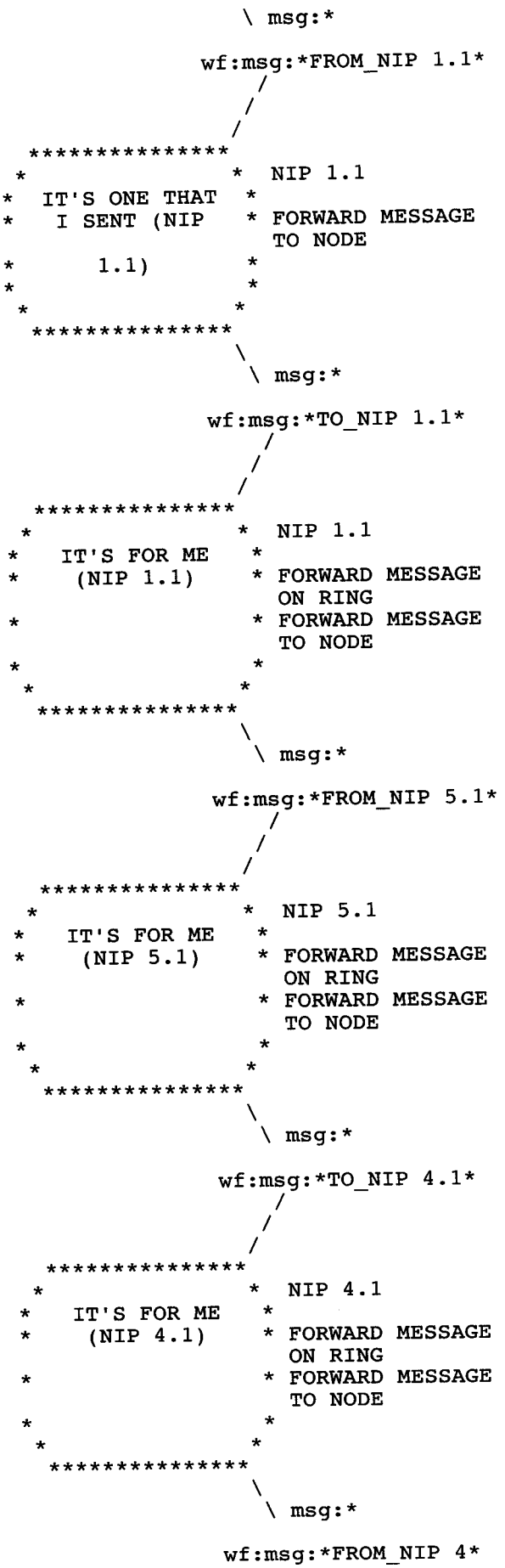


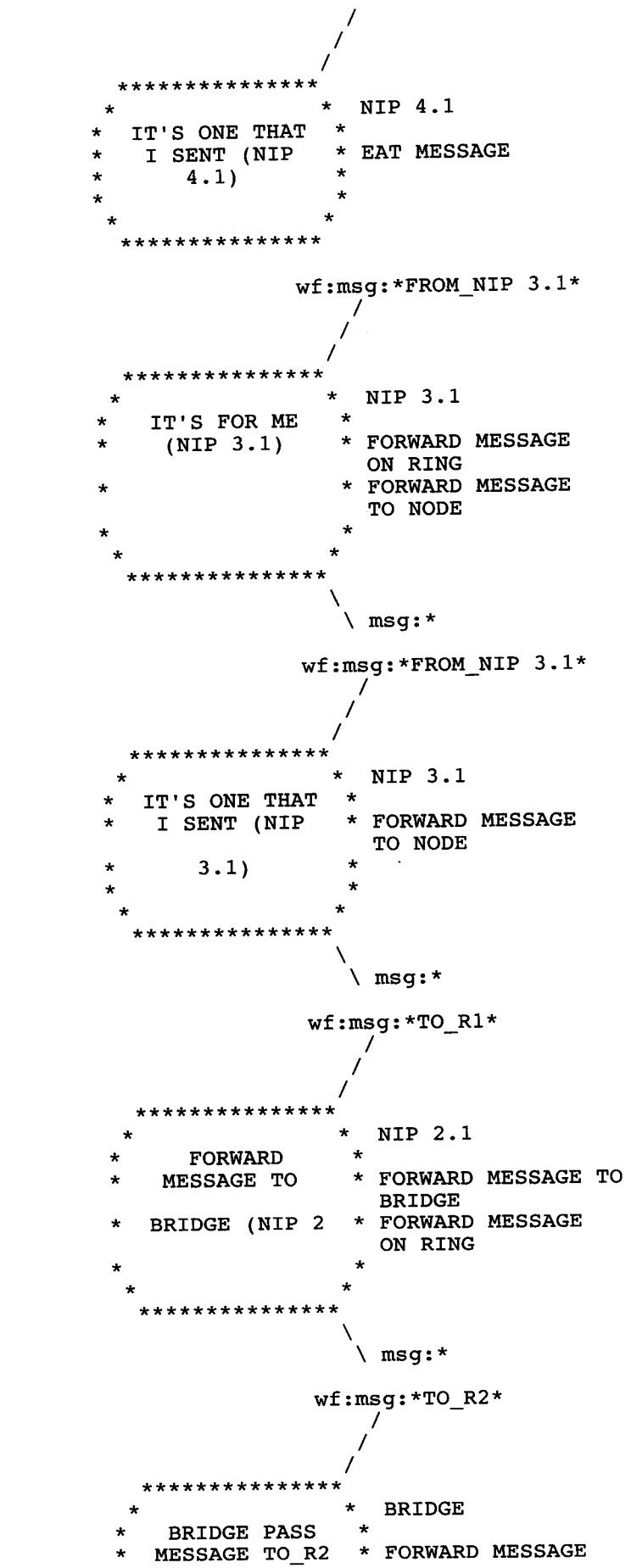
This simulation results shows that the time taken to transfer 100 bytes of data is only 47 seconds. With the intervention of the algorithms as the number of bytes increases, the congestion is controlled and hence the speed of data transfer is also increased. From this we conclude that Altered RED controls the congestion effectively and also works in equal to BLUE and ECN.

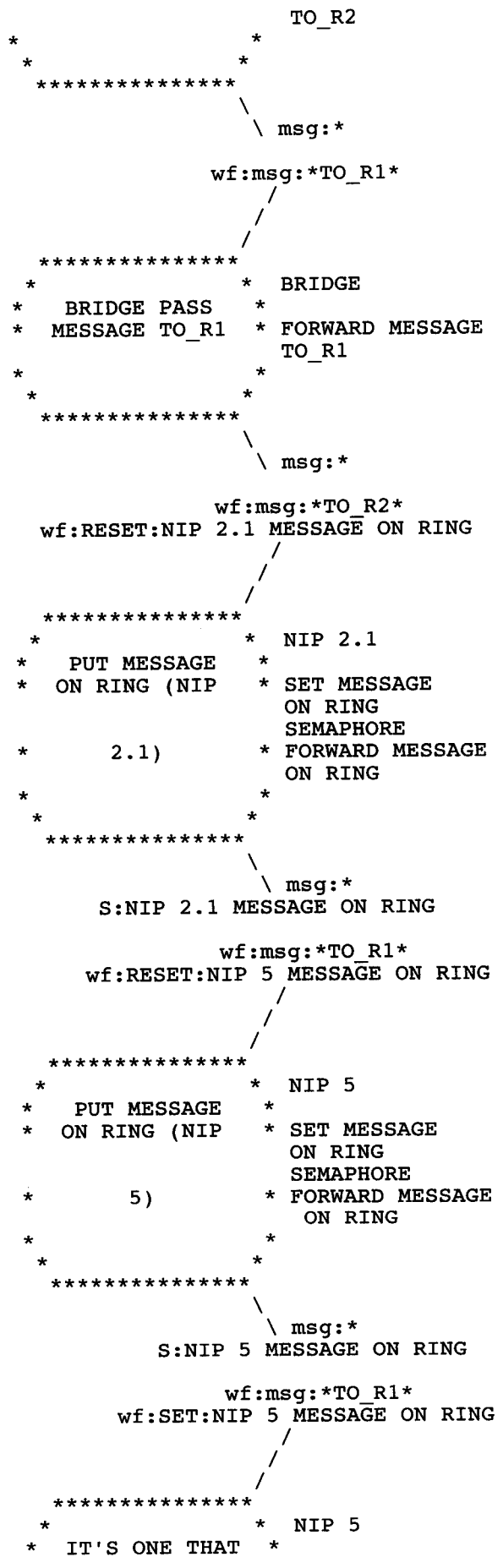

ip: EXPONENTIAL ITERATION PERIOD
START A MESSAGE
N1
etc.
START A MESSAGE
\msg:SEND A MESSAGE FOR ME
wf:msg:*
RECEIVE A MESSAGE
N1
etc.
PROCESS MESSAGE
wf:msg:*
IT'S NOT FOR ME
NIP 1
etc.
FORWARD MESSAGE ON RING
\ msg:*
wf:msg:*FROM_NIP 1*
IT'S ONE THAT I SENT
NIP 1
FORWARD MESSAGE TO NODE
\ msg:*
wf:msg:SEND A MESSAGE FOR ME
START A NEW MESSAGE ON RING
NIP 1
etc.
INITIATE MESSAGE ON RING


msg:TO_NIP 3
FROM_NIP 1
msg:FROM_NIP 2
TO_NIP 4.1
msg:FROM_NIP 3
TO_NIP 1
etc.
wf:msg:*TO_NIP 1*
NIP 1
IT'S FOR ME
(NIP 1)
FORWARD MESSAGE
ON RING
FORWARD MESSAGE
TO NODE
msg:*
wf:msg:*TO_NIP 4*
NIP 4
IT'S FOR ME
(NIP 4)
FORWARD MESSAGE
ON RING
FORWARD MESSAGE
TO NODE
msg:*
wf:msg:*TO_NIP 2*
NIP 2
IT'S FOR ME
(NIP 2)
FORWARD MESSAGE
ON RING
FORWARD MESSAGE
TO NODE
msg:*
wf:msg:*TO_NIP 3*
NIP 3
IT'S FOR ME
(NIP 3)
FORWARD MESSAGE
ON RING


* FORWARD MESSAGE TO NODE
msg:*
wf:msg:*FROM_NIP 2*
NIP 2
IT'S ONE THAT I SENT (NIP 2)
* FORWARD MESSAGE TO NODE
msg:*
wf:msg:*FROM_NIP 3*
NIP 3
IT'S ONE THAT I SENT (NIP 3)
* FORWARD MESSAGE TO NODE
msg:*
wf:msg:*FROM_NIP 4*
NIP 4
IT'S ONE THAT I SENT (NIP 4)
* FORWARD MESSAGE TO NODE
msg:*
wf:msg:*TO_R2*
NIP 5
PASS MESSAGE OVER BRIDGE (NIP 5)
* FORWARD MESSAGE ON RING
* FORWARD MESSAGE TO *BRIDGE


\ msg:*
wf:msg:*FROM_NIP 1.1*
NIP 1.1
IT'S ONE THAT I SENT (NIP 1.1)
FORWARD MESSAGE TO NODE
\ msg:*
wf:msg:*TO_NIP 1.1*
NIP 1.1
IT'S FOR ME (NIP 1.1)
FORWARD MESSAGE ON RING
FORWARD MESSAGE TO NODE
\ msg:*
wf:msg:*FROM_NIP 5.1*
NIP 5.1
IT'S FOR ME (NIP 5.1)
FORWARD MESSAGE ON RING
FORWARD MESSAGE TO NODE
\ msg:*
wf:msg:*TO_NIP 4.1*
NIP 4.1
IT'S FOR ME (NIP 4.1)
FORWARD MESSAGE ON RING
FORWARD MESSAGE TO NODE
\ msg:*
wf:msg:*FROM_NIP 4*


NIP 4.1
IT'S ONE THAT I SENT (NIP 4.1)
EAT MESSAGE
wf:msg:*FROM_NIP 3.1*
NIP 3.1
IT'S FOR ME (NIP 3.1)
FORWARD MESSAGE ON RING
FORWARD MESSAGE TO NODE
msg:*
wf:msg:*FROM_NIP 3.1*
NIP 3.1
IT'S ONE THAT I SENT (NIP 3.1)
FORWARD MESSAGE TO NODE
msg:*
wf:msg:*TO_R1*
NIP 2.1
FORWARD MESSAGE TO BRIDGE (NIP 2
FORWARD MESSAGE TO BRIDGE
FORWARD MESSAGE ON RING
msg:*
wf:msg:*TO_R2*
BRIDGE
BRIDGE PASS MESSAGE TO_R2
FORWARD MESSAGE


TO_R2
BRIDGE
BRIDGE PASS MESSAGE TO_R1
FORWARD MESSAGE TO_R1
msg:*
wf:msg:*TO_R1*
wf:msg:*TO_R2*
wf:RESET:NIP 2.1 MESSAGE ON RING
NIP 2.1
PUT MESSAGE ON RING (NIP 2.1)
SET MESSAGE ON RING SEMAPHORE
FORWARD MESSAGE ON RING
S:NIP 2.1 MESSAGE ON RING
wf:msg:*TO_R1*
wf:RESET:NIP 5 MESSAGE ON RING
NIP 5
PUT MESSAGE ON RING (NIP 5)
SET MESSAGE ON RING SEMAPHORE
FORWARD MESSAGE ON RING
S:NIP 5 MESSAGE ON RING
wf:msg:*TO_R1*
wf:SET:NIP 5 MESSAGE ON RING
NIP 5
IT'S ONE THAT