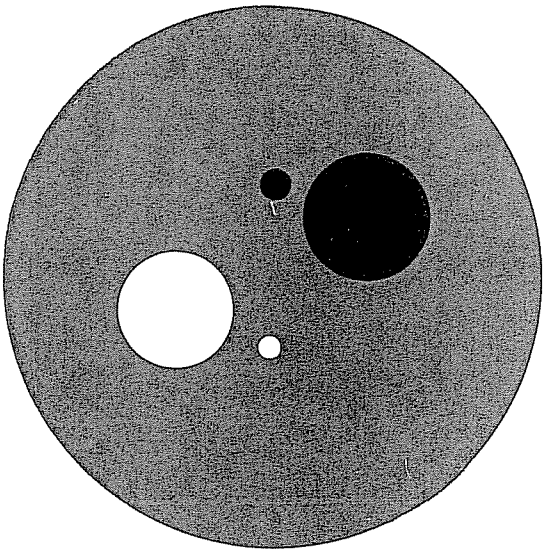
COMPUTER SCIENCES
DEPARTMENT

University of Wisconsin-
Madison



PARING: A TOKEN RING LOCAL AREA NETWORK WITH CONCURRENCY

by

Yaoshuang Qu
Lawrence H. Landweber
Miron Livny

Computer Sciences Technical Report #616
September 1985

PARING: A TOKEN RING LOCAL AREA NETWORK WITH CONCURRENCY

Yaoshuang Qu, Lawrence H. Landweber, Miron Livny

Department of Computer Sciences

University of Wisconsin

Madison, Wisconsin

ABSTRACT

In this paper we introduce the PaRing, a token ring architecture. The PaRing ring is distinguished from earlier ring architectures by its ability to concurrently support multiple communication paths on a single loop in an efficient and fair manner. This is accomplished by allowing more than one station to transmit a packet at a time. The result is higher throughput and shorter response time than is the case for the standard token ring or the register insertion ring.

1. INTRODUCTION

The ring and the *bus*[1] today comprise the two major categories of the local area network (LAN). Unlike the bus architecture, in which only one station is allowed to transmit at a given instance, the 'store and forward' characteristic of the ring architecture can support multiple stations transmitting on a single loop at a given time. This concurrent multiple transmission property can produce high data throughput and short response time. However, in current ring architectures such as Distributed Loop Computer Network (*DLCN*)[2,3] taking advantage of this property excludes two other important features: automatic acknowledgement and short station latency time. Automatic acknowledgement, i. e., packet reception acknowledgement performed by the destination station without having to send another packet carrying

the acknowledgment information, lets interface hardware detect a undeliverable packet and packet destruction promptly, so it reduces the overhead of error detection and recovery and improves protocol performance. Short station latency time leads to good throughput-delay characteristics.

In order to take advantage of the concurrent multiple transmission capability without losing automatic acknowledgement or short station latency time, we introduce a new ring architecture, the PaRing. The PaRing utilizes a token for medium access control as in the case of the standard token ring architecture. The important difference from earlier work is the combination of the following characteristics. First, the stations of the PaRing may initiate transmission not only upon receipt of a token, but also upon receipt of a data packet that is addressed to it. Second, the PaRing uses a destination removal mechanism, in which packets are removed by their destination stations rather than by source stations.

If a transmission opportunity, i.e., the arrival of either the token or a data packet, comes to a station and the station does not have a packet to send, it either forwards the incoming token or generates a token and then transmits the token while removing the incoming packet. When a station is transmitting its own packet and, at the same time, another packet arrives, the station keeps transmitting and either removes the arriving packet if it is a packet destined for the station or delays it, otherwise. After the station's transmission is completed, the station forwards delayed packets onto the ring. Since a station can remove an incoming packet which is transmitted by another station and transmit its own packet simultaneously, the PaRing supports concurrent multiple transmission. Because of the concurrent multiple transmission, the PaRing has a higher throughput and shorter delay time than the standard token ring architecture.

A packet transmitter knows whether or not its packet has been removed according to whether or not it receives the token or a different packet within an

anticipated time after its transmission. This feature leads to automatic acknowledgment. In the PaRing, each station only introduces a single bit latency time (as compared to the DLCN, which needs twenty four bit latency time.)

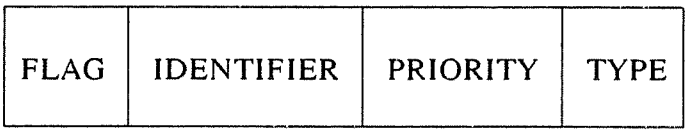
In this paper we present the medium access protocol and performance simulation studies of the PaRing. The paper is organized as follows. Section 2 gives details of the protocol, which includes packet formats, medium access and packet removal algorithms. Section 3 describes the automatic acknowledgment and a medium access priority assignment method that ensures a fair sharing of the medium bandwidth among all stations and enables the PaRing to be used for real time applications. Section 4 explains error detection and recovery mechanisms as well as system initiation. Section 5 is a brief description of the hardware organization of interfaces. Section 6 presents performance simulation studies, which compare the PaRing with two other ring architectures: the DLCN (a register insertion ring), and the *PRONET*[4] (a token ring). In the simulation, there is a distributed file server system, which is built on top of each of the three architectures. The comparison is based on how well each architecture supports the server. Section 7 provides conclusions.

2. MEDIUM ACCESS PROTOCOL

Two types of packets constitute the traffic of the PaRing - access control packets and data packets. Unlike the standard token ring where only an arrival of an access control packet carrying a free token can enable a data packet transmission, the arrival of either one of these packets may trigger the transmission of a packet in the PaRing. Thus in the PaRing, each packet has the potential to provide a station with the right to access the shared communication medium. All the information needed to determine whether transmission may be initiated is included in the header of the packet. A PaRing packet may consist of up to three fields - *Header, Body* and *Tail.*

An access control packet consists of only a header whereas all three fields have to be included in a data packet. Figure 1 illustrates formats of the header, the token, and the data packet. The PaRing access control scheme utilizes three types of access control packets - *token, invalid packet,* and *error recovery packet.* The packet header has four sub-fields: *flag, identifier, priority,* and *type.* The flag is a unique bit pattern that marks the beginning of the header. The type field distinguishes the three different control packets. The type field along with the identifier distinguishes between control packets and data packets. If both the identifier and type fields are "zeros", the packet represents a token. The priority sub-field is used for access control. More details on this sub-field will be given later.

For a data packet, the identifier of the header contains its destination address. The body part has four sub-fields: *monitor, source address, data,* and *data end.* The monitor sub-field (one bit) is used for error detection. The source address is the packet transmitter's address. The data sub-field can be of variable length but cannot exceed an upper bound. The data end, which is another unique bit pattern, marks the end of data. The tail part has two sub-fields, *error check* and *acknowledgement.* Both fields are for error detection.



Type:

00	Token or Data Packet
11	Error Recovery Packet
01 or 10	Invalid Packet

(a) Packet Header Format

FLAG	00.....00	PRIORITY	00

(b) Token Format

FLAG	DA ED SD TR	PRIORITY	00	MONITOR	SA OD UD RR CE	DATA	DATA END	EC RH RE OC RK	ACK
HEADER				BODY				TAIL	

(c) Data Packet Format

Figure 1 Formats of Packet Header, Token, Data Packet

A station with a ready data packet cannot make an attempt to transmit the packet until the arrival of a packet header in which the identifier contains either all "zeros" or the station's address, the priority level is lower than the station's level, and the type field is "00". A header that meets the above conditions is either the token or the header of a data packet that is addressed to this station. If an arriving header is addressed to the station but the priority level is not lower than the station's priority level, a new token is generated by the station and transmitted. In both cases, the header is removed before the transmission starts. If it is the header of a data packet,

the other parts of the data packet are removed during the transmission. If the station is too busy to remove an arriving data packet, it forwards the packet and sets its acknowledgement bit on the fly. When the arriving header does not meet the above conditions, the entire packet is forwarded by the station.

In order to have good throughput-delay characteristics, the latency introduced by a PaRing station is only one bit time. Since a station can hold only one bit of a transient packet, when a station recognizes an arriving header, the whole header except the last bit has left the station. If the station intends to use the header for transmission, the station must remove it before the transmission starts. Instead of physically removing the whole header, the station just changes its last bit and converts it into an invalid packet. (See Figure 1 (a)) The invalid packet will be removed from the ring when it is delayed by a station.

The PaRing access protocol can be described by a state diagram that consists of eight states, eight events, and three conditions. The events and conditions are summarized in Tables 1 and 2 and the state diagram is presented in Figure 2. Eight states are briefly described as follows.

Repeat:

A station directly forwards a packet arriving from the medium.

Trans:

A station transmits its own data packet.

Remove:

A station removes an arriving data packet which is destined for it.

Dump:

A station forwards a delayed packet.

Trans.Rmv:

A station transmits its own data packet and, at the same time, removes an

arriving data packet which is destined for it.

Trans.Delay:

A station transmits its own data packet and, at the same time, delays a packet.

Dump.Rmv:

A station forwards a delayed packet and, at the same time, removes an arriving data packet which is destined for it.

Trans.Delay.Rmv:

A station transmits its own data packet, removes an arriving data packet which is destined for it, and, at the same time, delays a packet.

In the introduction, we indicated that a station delays arriving packets not destined for it upon the transmission of its own packet and forwards the delayed packets after the transmission. This corresponds to the state transition sequence from Trans state via Trans.Delay state to Dump state with events (A + C) (the arrival of either the token or a data packet not addressed to the station) and E (the completion of transmission of a data packet), respectively. The transition from Trans state to Trans.Delay.Rmv state is done by event B, arrival of a data packet that is addressed to the station, as follows. Upon the arrival, the station keeps transmitting its own packet, removes the incoming packet, generates a token, and delays (or buffers) the token. In order to simplify the diagram, we assume that once a station detects a delayed invalid packet, it eliminates the packet immediately. Thus the arrival of an invalid packet does not change the state of a station.

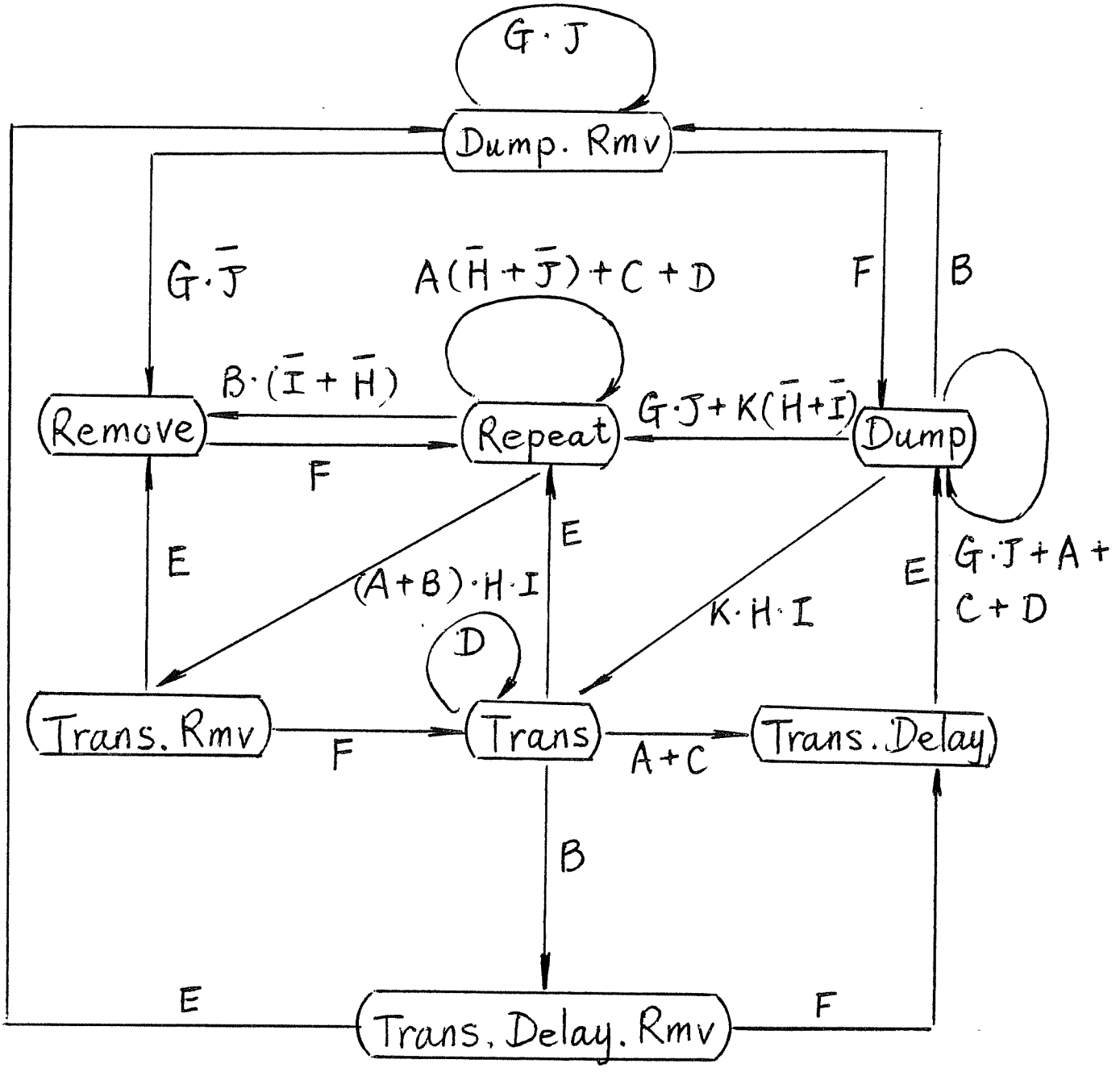


Figure 2 State Diagram for Protocol

Table 1 Events of State Diagram

A.	Arrival of the token
B.	Arrival of a data packet addressed to the station
C.	Arrival of a data packet not addressed to the station
D.	Arrival of an invalid packet
E.	Completion of transmission of a data packet
F.	Completion of removal of a data packet
G.	Completion of forwarding of a delayed data packet
K.	Completion of forwarding of the delayed token

Table 2 Conditions of State Diagram

H.	Station has a packet ready to transmit
I.	Station has a higher priority level than incoming packet
J.	There is more delayed packet(s) in the station

3. AUTOMATIC ACKNOWLEDGMENT, ACCESS PRIORITY ASSIGNMENT, AND FAIRNESS

In the PaRing, we regard the communication loop together with the facilities that store all delayed packets as an integral environment. Call this environment the *system.* Call the header of the token and that of a data packet a *valid header.* When the system is initiated, there is only one valid header. (See 4. ERROR DETECTION, RECOVERY, AND SYSTEM INITIATION) According to the medium access and packet removal algorithms, a station cannot transmit a valid header into the system without removing the existing valid header from the system, so there is no more than one valid header in the system at any time. This single valid header keeps circulating around the system. When the transmitter of a data

packet sees a valid header arriving in the period after the transmission of its packet and finds that the header is either the token or the header of another data packet, the transmitter knows that its packet has been removed. If it sees the transmitted packet returning, the transmitter, by checking the acknowledgement bit of the packet, can know why the packet has not been removed. This feature leads to the automatic acknowledgment.

As for the medium access priority assignment, first we discuss how a station is dynamically assigned a priority level, then explain how a data packet and token obtain their priority levels. Once a station has a data packet ready to transmit, it is assigned a medium access priority level, and waits for a transmission opportunity to come. The priority level is determined by the time constraint of the application the packet is used for. In addition, each station has a timer called a *priority timer*. Whenever the station is assigned a priority level, it initializes the timer. If the station cannot transmit its own packet before the timer times out, it raises the priority level and reinitializes the timer. When the station starts transmission of the packet, it stops the timer and decreases the priority level. Therefore the stricter time constraint a station has or the longer the station waits for transmission, the higher priority level the station has.

When a data packet that is not addressed to a station is passing through, the station compares its priority level with that of the packet. If the station has a higher level, it records the packet's level, enters its own level into the packet's priority sub-field, and sets the packet's monitor bit. All these actions are done on the fly. The recorded level is called the *original level* of the packet at this station. By doing this, a station may enter its priority level into an invalid packet. However, that does not cause any trouble. When a station removes an arriving valid header, either the header of a data packet or the token, and transmits its own packet, the monitor bit of the transmitted packet is not set. The priority level of the transmitted packet is

decided as follows. If the transmitting station entered into the previous valid header the same priority level as the removed packet has, i. e., no other station has increased the priority level entered by this station in the previous header, the new transmitted packet inherits the original level of the previous header. Otherwise, the transmitted packet copies the priority level of the removed header. The same principle is used to decide a token's priority level when it is generated by a station. For a station, entering its priority level into a passing packet is, in effect, a request for the next transmission right. Generally, the station entering the highest priority level obtains the next transmission opportunity unless the destination station of the passing packet has a higher priority level than the highest entered level. Along with the medium access and packet removal algorithms, the medium access priority assignment method ensures a fair sharing of the medium bandwidth between all stations and allows satisfaction of time constraints for accessing the medium by different applications.

4. ERROR DETECTION, RECOVERY, AND SYSTEM INITIATION

In the PaRing, there are five error cases: packet destruction, valid header loss, multiple valid headers, undeliverable data packet, and system break. The error detection and recovery are completely distributed.

Upon removing a packet, a destination station computes parity value of the packet and compares the value with the packet's parity bit. A mismatch between these two values represents a destruction error. The second and third error cases can be detected by the fact that the single valid header circulates around the ring periodically. Each station keeps track of the valid header with the aid of a timer called a *round trip timer,* in which the time limit is equal to the period. Whenever a station sees a valid header, it initializes the round trip timer and waits for the next arrival of the header. If the valid head comes too early, there is a multiple valid

header error. If it does not arrive until the timer times out, there is a valid header loss error. As for the undeliverable data packet, there are two ways to detect it. First, if the transmitter of a data packet sees its own packet returning without the acknowledgement bit set, the packet is undeliverable. Second, if a station that overwrote the priority sub-field of a data packet and set its monitor bit sees a data packet coming with the same sub-field values as the station entered, this packet is an undeliverable packet as well. The system break error is discussed below.

Upon detecting an error, a station transmits an error recovery packet (Figure 1 (a)), in which the address sub-field contains the station's address and the type is "11", and initializes its round trip timer. After transmission of the error recovery packet, the station keeps removing all incoming packets until its own error recovery packet returns. When the station receives its own error recovery packet, all undeliverable packets and other access control packet(s) have been eliminated from the system if they were there. Then the station can transmit a new valid header, which is either a token or its own data packet. At this point, the system starts to operate normally. If the station cannot get back its own error recovery packet until the round trip timer times out, there is a system break error and some actions from operators should be taken. After seeing an error recovery packet, stations that have not transmitted an error recovery packet wait for the system to be recovered.

If more than one station detects an error and transmits an error recovery packet simultaneously, there is a collision problem. In order to solve the problem, a station with a higher address removes error recovery packets with lower addresses. Here, removing an error recovery packet is also done by changing its last bit and turning it into an invalid packet. Thus eventually the transmitter with the highest address will get the ring recovered.

The system initiation case is the same as the case of the valid header loss since the system does not have a valid header at initiation time.

5. HARDWARE ORGANIZATION

The PaRing consists of a single loop of the communication medium and a number of stations. A station comprises a host and an interface. Through the interface the host is connected to the loop. Like the PRONET, the interface of the PaRing consists of *modem, input machine, output machine, register unit,* and *buffer unit.* However, in addition to an input buffer and an output buffer, the buffer unit of the PaRing has a delay buffer, which is used to store incoming packets when the interface is transmitting its own packets. In this section, we compute only the size of the delay buffer. The delay buffer is at least as big as is the untransmitted portion of the packet when the incoming packets arrive at the interface. Therefore the size of the delay buffer, SDB, depends on the number of interfaces, N, the interface latency time, LT, and the upper bound of data packet sizes, UB. Define the shortest round trip time, *SRT*, to be the time for the valid header to complete a circulation around the loop without being buffered by any interface. We have

$$SRT = LT * N$$

and

$$SDB = \begin{cases} UB - SRT & \text{if } UB > SRT \\ 0 & O.W.. \end{cases}$$

6. PERFORMANCE SIMULATION STUDIES

This section presents a performance simulation comparison of three ring architectures: the PaRing, the DLCN, and the PRONET. In the simulation, each architecture supports an identical distributed file server system, respectively. The protocol details of the DLCN and the PRONET can be found in the references. Table 3 illustrates their interface latency times and packet overheads, which are equal to the total length of nondata fields.

Table 3 Packet Overhead and Interface Latency Time

	Packet Overhead (bits)	Interface Latency Time (bits)
PaRing	40	1
DLCN	49	16
PRONET	27	1

For the PaRing, the flag, identifier, address, and data-end sub-fields are eight bits long. Priority sub-field is three bits long. Error check sub-field is one bit long. For simplicity, assume the absence of the priority scheme in the simulation. For the DLCN, the error check field of packets is one bit long and the interface latency time is sixteen bits long, instead of twenty four bits, so that a fair comparison can be carried out. Delay buffers are as big as the maximal packet size is. Packets in delay buffers have higher priorities to enter the medium than packets in output buffers.

In the system there are several identical file servers and many disc-less users, which must get files from the file servers. All users have the same request generation rate to get files from servers. After receipt of a file, an user sends an acknowledgement to the server. A server does not start to send the next file until it receives the receipt acknowledgement for the previous file. A user does not send the next request until it receives the previous file. Packet size is 512 bytes, which is the upper bound of the DLCN packet. A file needs more than one packet to carry. On the other hand, a request or an acknowledgement only needs one packet to carry. Each station comprises two layers: a client layer and a datalink layer. The user client layer generates both requests and acknowledgements. The server client layer generates files and divides files into frames, each of which fits a packet.

Datalink layers obtain packets from their client layers and transmit them onto the medium or remove packets from the medium and deliver them to the client layers. In addition, the stop-and-wait method is used at the datalink layer, i. e. a transmitter will not send the next packet until it receives the receipt acknowledgement for the previous packet from the recipient. Thus, for the DLCN, a recipient has to generate a separate packet as an acknowledgment because it does not have the automatic acknowledgement feature. Each user gets files from the closest upstream server and each server serves the same number of users (if possible). Figure 3 illustrates the distributed file server system. Users 1 up to n get their files from server 1 and users i up to i+n get their files from server k.

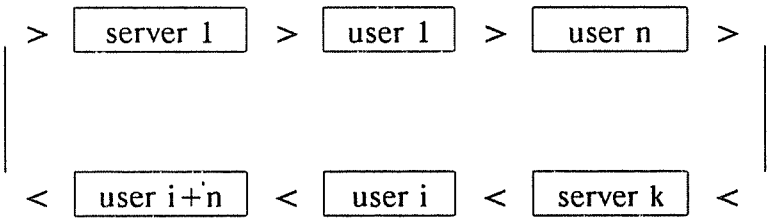


Figure 3 Distributed File Server System

In the simulation, three architectures have the same workload. At user sites, request generation interval times are exponential distributed with the same mean values. Files have the same length. The medium bandwidth is 1Mbit per second and its transmission error rate is "zero".

In the simulation, we study the effect of the size of files, the number of stations, and the number of servers on the response time. The response time is defined to be the time interval between the request generation at the user client layer and the

acknowledgement receipt at the server client layer. The simulators are written in *DISS*[5].

Figures 4 through 7 are results of the simulation. By P/N/S/F, we denote the PaRing system in which there are N stations, S servers, and each file consists of F packets. Similarly, the D/N/S/F and T/N/S/F denote the DLCN and the PRONET systems, respectively. Figures 4 and 5 show the characteristics of response times versus request arrival rates of the three systems. For a further comparison between the PaRing and the DLCN, figures 6 and 7 display the characteristic of response times versus server numbers and file sizes, respectively. The simulation results indicate that the PaRing provides the system with a shorter response time than the other two architectures because of the concurrent multiple transmission, the automatic acknowledgement, and the short station latency time.

7. CONCLUSION

This paper presents the architecture of the PaRing and its performance simulation studies. The main advantages of the PaRing are the following.

(1) Good throughput-response time characteristic accomplished by concurrent multiple transmission, automatic acknowledgment, and short interface latency time.

(2) Access fairness and adaptability to different applications obtained through the medium access priority assignment method.

(3) Reliability achieved through completely distributed error detection and recovery mechanisms.

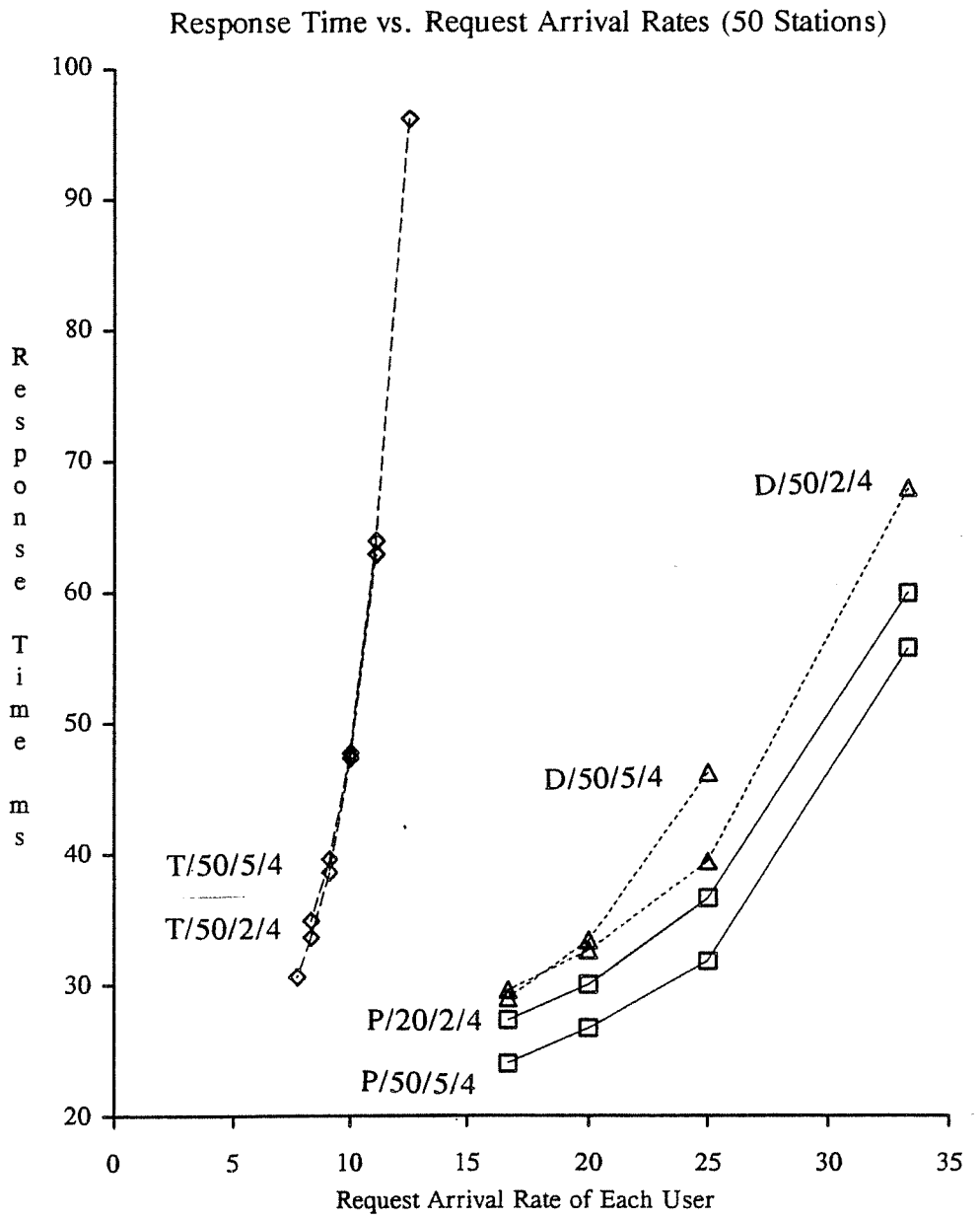

Response Time vs. Request Arrival Rates (50 Stations)
Response Time ms
100
90
80
70
60
50
40
30
20
0
5
10
15
20
25
30
35
Request Arrival Rate of Each User
T/50/5/4
T/50/2/4
D/50/5/4
D/50/2/4
P/20/2/4
P/50/5/4

Figure 4

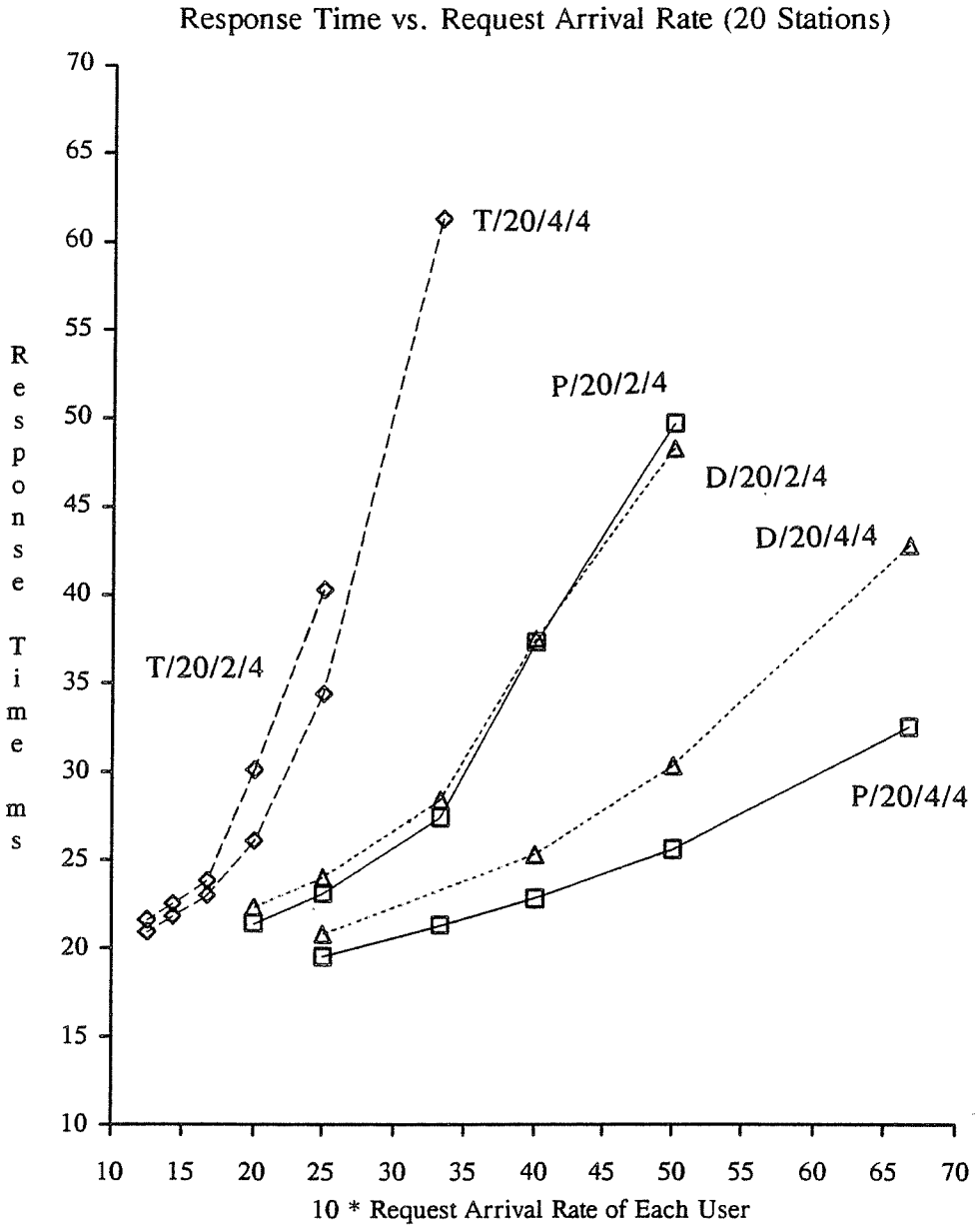

Response Time vs. Request Arrival Rate (20 Stations)
Response Time ms
70
65
60
55
50
45
40
35
30
25
20
15
10
T/20/4/4
P/20/2/4
D/20/2/4
D/20/4/4
T/20/2/4
P/20/4/4
10 15 20 25 30 35 40 45 50 55 60 65 70
10 * Request Arrival Rate of Each User

Figure 5

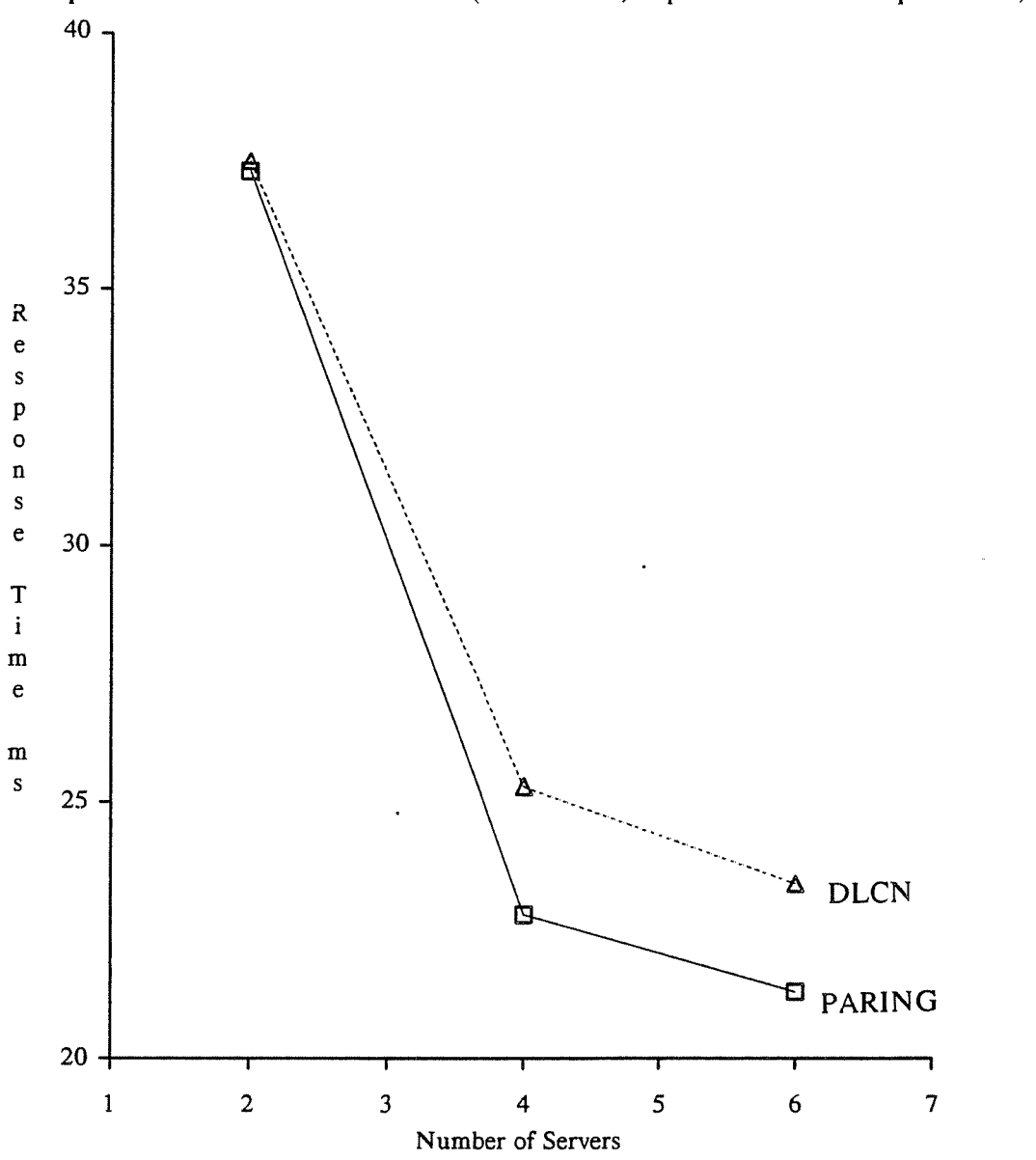

Response Time vs. No of Servers (20 Stations, 4 packets/File 4 Requests/sec)
40
35
30
25
20
Response Time ms
1
2
3
4
5
6
7
Number of Servers
DLCN
PARING

Figure 6

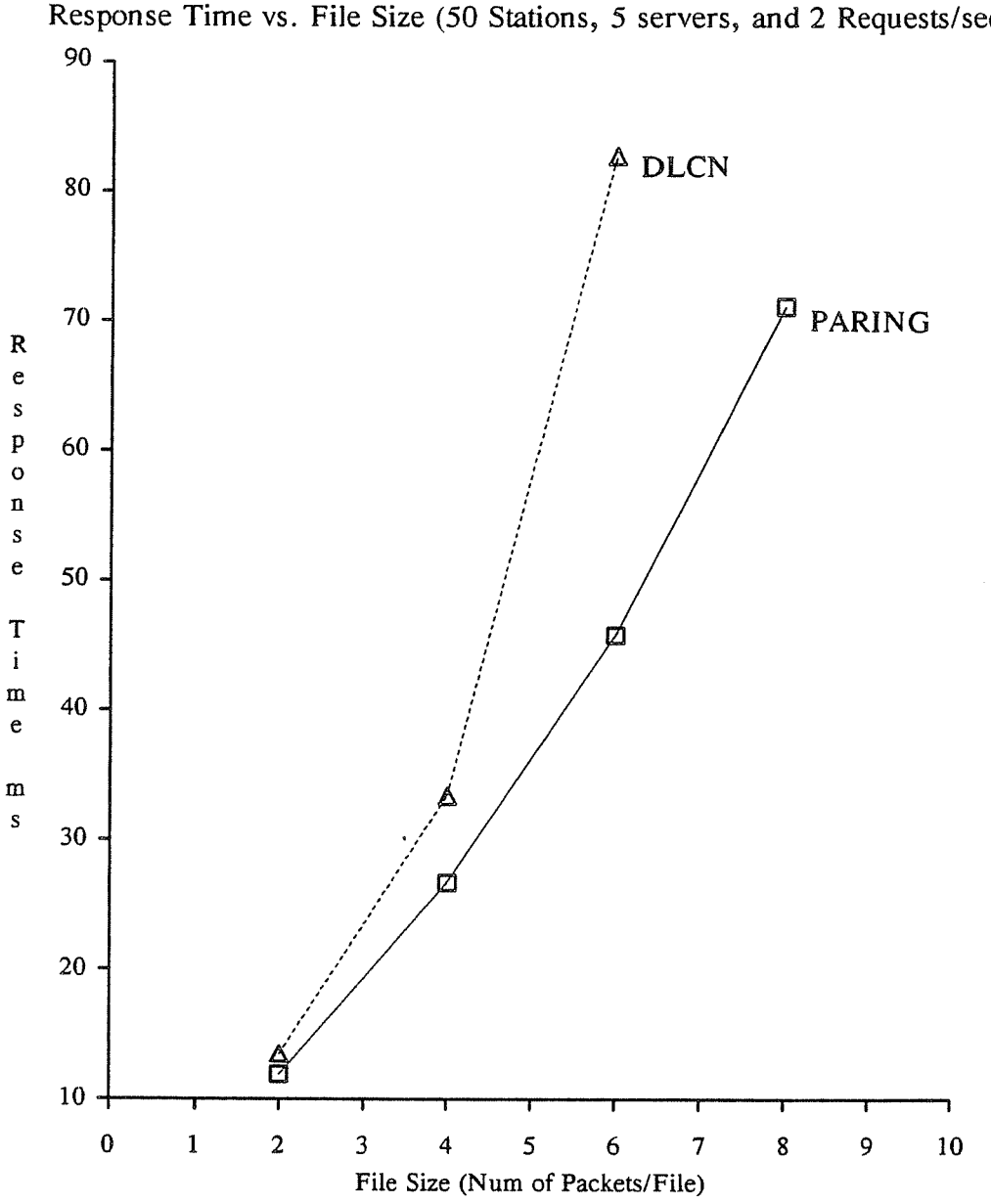

Response Time vs. File Size (50 Stations, 5 servers, and 2 Requests/sec)
90
80
70
60
50
40
30
20
10
Response Time ms
DLCN
PARING
0
1
2
3
4
5
6
7
8
9
10
File Size (Num of Packets/File)

Figure 7

References

[1] Metcalfe, R. M., and Boggs, D. R. "Ethernet: Distributed Packet Switching for Local Computer Networks", Commun. ACM 19, July, 1976, pp. 395-404.

[2] Reames, Cecil C. and Liu, Ming T. "A Loop Network for Simultaneous Transmission of Variable-Length Message", Proc. 2nd Annual Symp., Comp. Arch., Jan. 1975, pp. 124-129.

[3] Liu, M. T. and Rouse, D. M. "A Study of Ring Networks", Proc. of the IFIP WG6.4/Univ of Kent Workshop on Ring Technology Local Area Networks, Kent, U. K., Sep. 1983, pp. 1-39.

[4] Proteon Associates, Inc., "Operation and Maintenance Manual for the PRONET Local Network Interface", Issue No. 5, April 30, 1982.

[5] Melman, Myron and Livny, Miron, "The DISS Methodology of Distributed System Simulation", SIMULATION, April 1984, pp. 163-176.