RING COMMUNICATION PROTOCOLS

Donald C. Loomis

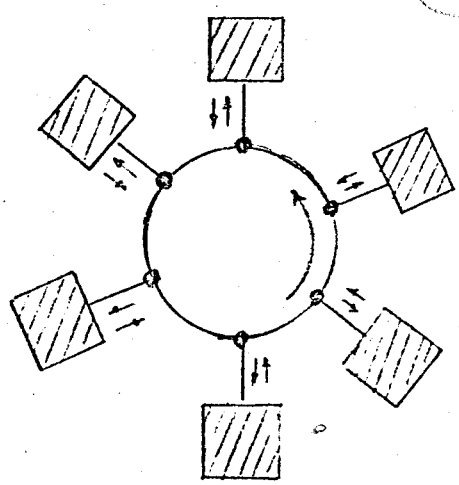
TECHNICAL REPORT #26 - 1973

Ring Communication Protocols

Donald C. Loomis

In this paper we will examine a number of schemes for coordinating message transmission among computing components which are connected together by a single unidirectional communication channel. In order that each component can send messages to every other component, the communication channel is continuous and circular. This channel is the communication ring. It is also known as a looped communication system. By using appropriate communication protocols, it is possible for each component to communicate with any of the other components without knowing their physical placement around the ring.

The interconnection may be shown graphically:



Here the squares represent computing components, and the dots represent the interconnection of the components to the circular communication channel. The term "ring interface" is used to designate each of these interconnection points. The arrows indicate direction of information flow.

In principle a computing component could be any of the elements from which a computing system could be built. We are particularly interested in the following types of components.

1. A central processor and main memory
2. A single terminal
3. A number of terminals connected by a data concentrator
4. A data file storage device
5. An input-output device or devices
6. A central processor and swapping storage
7. A computing system with primary and secondary storage and possibly I/O devices.

The only real restriction on the nature of the component is that the component in conjunction with the circuitry of the ring interface be able to obey the communication protocol. As a practical matter because of the limited bandwidth of the communication channel, high bandwidth communication such as between a processor and its main memory must not be done over

the ring. Instead a processor and its memory will be connected to the ring as a single component. Similarly a storage device which is used to extend the main memory size of a processor (e.g. swapping) may not be connected to the ring as a separate component. On the other hand, it is desirable to connect a file storage device as a separate component in order to allow access by any of the processors on the ring.

It is often more useful to think of components as functional units rather than to think of the hardware of which they are made. A component can be a terminal, a file storage machine, an APL processor, a BASIC processor, etc.

With a general idea of what components must communicate with each other we turn to some of the problems in connecting them. There are a number of characteristics inherent in the ring which are worth noting. The first of these is symmetry. Every component connects to the others through the ring. The only difference among the connections is the physical locations around the ring. The communication protocols we will consider make this difference transparent.

Simplicity arises from the symmetry since each ring interface functions in the same way as every other ring interface. The fact that the ring interface must connect the component to only one input and one output also contributes

to the ring interface simplicity.

On the negative side there are some severe problems to be solved. First, if the communication channel malfunctions at any point, conversation among components ceases. Only certain one way messages along the functioning portion will be possible. Discussion of solutions will be deferred until later.

Another problem is one of timing. Since there is a single communication channel, all messages must travel via that channel. All the delays incurred along the channel are cumulative and contribute to the total delay between the time the message originated and the time it reached its destination. Delays are introduced along the channel due both to propagation time and the delays at each ring interface it must go through. Consequently, the time it takes for a transmission between two components via the ring could be substantially greater than via a direct point-to-point connection. Keeping such delay small is an important consideration for two reasons. First, this delay increases the elasped time required to perform those computing tasks which require communication among components. This affects response to user requests. Second, computing resources which are held during the delay are not available for other use. This affects the utilization of computing

resources. Because of these factors, the delays in message transmission will be an important consideration in the choice of communication protocols. Several other performance factors are important in the selection of the communication protocol. The capacity for data transfers between components must be adequate to support the system's particular applications. In addition to the communication protocol, the bandwidth of the technology used to implement the ring is an important determinent of this.

The protocol must contribute to the overall reliability of the system. Hence it must be fairly simple but complex enough to allow detection of errors. In addition to the requirement of reliability, the protocol must facilitate recovery from a variety of errors. These range from intermittent transmission errors to failure of a ring interface to complete interruption of the transmission facility. The system must be able to correct simple errors as if they had never occured. For more complex errors only slight interruption in service can be allowed. The design of the ring interface is further constrained by the cost. Hence this must be taken into account in selecting a communication protocol.

A basic issue is whether the communication ring is multiplexed or used as a single high rate channel. Another

possibility is to use it in both ways--sometimes as a single high speed channel and other times as several lower speed channels. This might be done by providing mechanisms for combining channels into high speed channels and also for dividing them into lower speed channels.

There are a number of factors which affect the decision on how to allocate the communication path into channels. The first of these is component speeds. Included in component speeds are both the physical data rate capabilities of the devices and the rates at which they are typically used. Transfers from one computer memory to another can physically be made very quickly--frequently at the speed of the memory cycle. In some cases this high speed communication is desired. In other cases, such as when each character must be processed individually by the receiver a much lower rate is satisfactory. A slow speed output device such as a Teletype terminal can physically transfer at no more than 10 characters per second. Used as an input device a Teletype can physically realize only 10 characters per second but typically the rate is even slower.

In addition to the fact that a range of devices from high speed to low speed must be accommodated, there is another problem. Many devices have a large variability in the rate at which data is available for transmission.

Terminal input typically is typed at an uneven rate and hence exhibits uneven availability. Even data fetched directly from the main memory of a computer may be available at a varying rate if there are other demands for memory access. To efficiently multiplex communication of these devices on the communication channel would require a rather elaborate channel allocation mechanism. One should expect a substantial portion of the channel bandwidth to be used for allocation information. Also the construction of the nodes would become quite complex.

To avoid the problems associated with many data rates and variability of data rates it is possible to accumulate an entire block of data into a buffer before any of it is transmitted. Similarly, a block of data is buffered at the receiving component. This leads directly to using the communication path as a single high speed channel and transmitting the entire block at a high speed. Substantial simplicity is thus gained. For devices with low or erratic data rates, buffers capable of burst transmissions at the channel's data rate must be provided. The cost of these buffers is quite low.

There are, however, some disadvantages arising from this scheme. First, when an entire block is accumulated before it is sent, it is not possible for the component which is to

receive the message to do any processing of the message as it is being generated. If the message were not buffered, the receiving component could begin processing at the beginning of the block before all of the data in the block had been assembled by the sending component.

A second disadvantage arises if the transmission block size is very large. Since an entire block is accumulated before any data is transmitted, there can be no overlap of transmission during the assembly of the message.

Both of these restrictions are a question of overlap of processing with block transmission. With buffered transmission block assembly, transmission, and block disassembly must be performed serially. The total time is the sum of the time for each.

Without buffering, it may be possible to overlap all three. In this case, the total time is the longest of the three. Although conceptually important, the time involved is usually small and shouldn't affect practical performance. It is interesting to note that some overlap can be achieved even with buffering if a message is transmitted as a number of blocks. Also when the length of a message exceeds that which can conveniently be held in a buffer it must be transmitted as several blocks.

The conclusion of this analysis is then that a block of

data will be accumulated in a buffer before being transmitted via a high speed channel to a buffer at the receiving component. When necessary, a message will be transmitted as several blocks.

Given a buffered message transmission scheme, it is necessary to decide how to allocate the use of the channel in order that ready-to-send messages can be transmitted. All messages cannot be sent at once since a message will become garbled when it circulates around the ring past a ring interface which is also placing a message on the ring. Hence, a mechanism for the orderly transmission of messages via the ring must be provided.

The usual solution is to specify a special component (controller) which allocates authority to the components to transmit messages. For proper operation of the communication ring it is necessary that the controller always function properly. To avoid the reliability problems which result from this limitation, it is desirable to have a scheme in which no single part of the system has all of the responsibility for control. Instead, components which are functioning properly coordinate with each other for message transmissions. Components which malfunction are not allowed to interfere with the rest of the system. Such a control mechanism is one of distributed control. Instead of being

centralized, the control is distributed among many components which must coordinate with each other.

In the following section we will examine mechanisms using distributed control to allow message transmission by nodes without interference from each other. Each of the schemes will be described briefly. Then they will be compared and contrasted on a number of features.

The following three mechanisms are to be considered:

1. Control passing
2. Lazy Susan
3. Delay relaying of messages.

Control Passing

With control passing a single ring interface is authorized to transmit a message on the ring at any one time. All other ring interfaces must simply relay all data received to the next ring interface. When the authorized ring interface is through transmitting, it will transmit a special sequence of bits known as a control token. Any other node which wishes to transmit will replace the control token with a message followed by a control token. If no ring interface needs to transmit, the control token will circulate around the ring until a ring interface is ready to transmit. When there are interfaces which desire to transmit, the control token will be delayed at each ring interface which is ready to transmit while a message is transmitted.

Because only one point in the ring is authorized to alter the information passing around the ring, messages will always circulate completely around the ring until they reach the point which is authorized to transmit. This must be completely around the ring and may be more if the point authorized to transmit moved while the message was going around the ring. This guarantees that it can be received by any of the nodes to which it might have been destined.

Instead of allowing a message to continue around the

ring until it arrives at a point which was transmitting, it might be logical to absorb it at either of two places. It might be absorbed by the ring interface at the message's destination. To do this requires that every message be delayed at every ring interface enough to recognize whether the message is destined for that ring interface. This delay must be at least enough to recognize the destination address field in the message. With a ring of n nodes the address field and consequent delay must be at least log2 (n) bits. Imposing this delay at every ring interface is undesirable. Since there do not appear to be significant advantages to absorbing a message at the destination ring interface, doing so and imposing this delay is undesirable. On the contrary, allowing messages to travel completely around the ring provides a convenient way to send broadcast messages which are addressed so they will be received by several or all of the components.

The other logical point for absorbing a message is the point at which it originated. When messages are absorbed at the point where they originated, they are guaranteed to have gone completely around the ring once and only once. The originating ring interface can check the message as it is received and determine if there were transmission errors. If the received message is different than that transmitted,

the originating interface is aware that the destination interface might have received an erroneous message. Guaranteeing that a message normally passes each ring interface only once reduces the burden of receiving and ignoring successive copies of the same message. It is rather simple for an originating ring interface to absorb the messages it transmits if all ring interfaces absorb the messages they transmit. Since the control token always follows the last message transmitted, all messages which were transmitted before a node starts transmitting will have already passed that node. These earlier messages will be absorbed where they were created. As a result, the next message the ring interface receives after it begins transmitting will be the message it transmitted. Since an address need not be recognized, a delay at each node is not necessary.

The only delay required at each node is that needed to reform the transmission signal pulses. Aside from this all relaying can be on a bit by bit basis.

There is nothing in this protocol which restricts the length of messages to a fixed size. Once a node has received the control token, it can transmit a message of any length. If it desires it may transmit several separate messages each of arbitrary length. However, transmission of

several messages delays passing the control token.

It is important that the control token circulate around the ring frequently so that no interface will be prevented from sending messages for too long. By limiting the amount of transmision by any ring interface before passing the control token, this can be achieved. The number of messages might be limited (e.g. 1) Or the total length of all messages (time) might be limited, or both.

Lazy Susan

Another mechanism for coordinating use of the communication ring gets its name from its resemblance to a lazy Susan. When the physical size of the communication ring is large or there are substantial delays in the transmission (e.g. at each ring interface), the time it takes for a signal to go around the ring and return to its origin may be quite long. If by the time a message of a fixed size is completely transmitted the beginning of the message has only gotten 1/3 of the way around the ring, we can say the distance around the ring measured in bits is three times the size of the message. This interval in the time-space around the ring as well as the two other intervals of equal size can be viewed as trays circulating around the ring. If there is no message in the tray when it

passes a ring interface, the ring interface may place a message in the tray. As long as the transmission is complete by the end of the tray interval, the message in the next tray can be passed on without damage. As a result, three ring interfaces can be transmitting at the same time without garbling each other's messages. In general the number of nodes able to transmit will be equal to the integer part of the delay around the ring in bits divided by the fixed message size.

Since there can be several trays, the possibility of transmitting a message by a ring interface which has one to send occurs several times as often (as each tray passes). However, when a tray is full it cannot be used and the ring interface must wait until an empty tray is available.

The lazy Susan scheme fixes the length of messages at the tray size. Shorter messages can be accommodated, but the transmission capacity is not recovered. When using the lazy Susan scheme, messages must always be explicitly removed to make the message slot available. Otherwise messages would continue to circulate around the ring indefinitely. They may be removed either at the destination or at the origin after traveling completely around the ring. Since the destination cannot be expecting the message at any particular time, it cannot delete the message unless it

never sends a message on to the next node until it has determined whether or not to send the message or to transmit an empty slot. To determine this requires that it has received and analyzed a destination address at the beginning of the message. To delay the relaying of every message long enough to do this is undesirable.

On the other hand, the originating ring interface can anticipate the time the message will have gone completely around the ring. It only needs to know the delay around the ring to the precision of one slot-time. When the appropriate slot arrives, it can flag the slot as empty and available without any delay in the ring interface.

Message Delaying

A third method of preventing interference among ring interfaces is not to regulate when ring interfaces may start transmitting, but instead require that a ring interface delay incoming messages when required. If it is transmitting a message of its own, it must delay the relaying of incoming messages until it has completed the transmission of its own message. A ring interface may not start sending a message in the middle of relaying a message but is otherwise unconstrained. The net effect of this is to delay all messages which encounter conflict at a node by

the amount of conflict.

Again the message can be deleted from the ring either by the destination ring interface or by the origin ring interface. However, since the time it takes a message to go around the ring is variable, the destination or origin ring interface must be able to identify the message. This is best done by placing an address field at the beginning of the message. The address might be that of the destination if the destination ring interface is to remove the message or that of the originating ring interface if the originating ring interface is to remove the message. To address all ring interfaces uniquely the length of this field in bits must be log2 the number of ring interfaces. In order to delete a message, a ring interface must recognize this field and either alter it or alter a previous field to indicate "no message". Hence each node must delay retransmission by at least the length of this field.

A slight variation of this scheme is less efficient but is quite easy to implement in hardware. A ring interface which receives a message to relay while it is transmitting a message does not merely delay it until it is through transmitting. Instead it delays relaying of all messages received by the length of the message it is transmitting. It removes this delay and "catches up" when the message it

transmitted returns (by not relaying it on).

This is easily implemented in hardware by using a FIFO shift register. The message to be transmitted is first placed in the shift register. Anytime except when in the middle of relaying a message, the message may be transmitted by shifting it out of the shift register. Received data is entered into the shift register and is thereby delayed by the length of the message. Shifting all received data through the register continues until the message transmitted has circulated completely around the ring and back into the shift register. At this time direct relaying of the messages received without delaying them in the shift register can be resumed.

Evaluation

The primary criteria on which evaluation of these schemes is to be made is the amount of delay inflicted on messages. This comparison is made assuming a fixed bandwidth for the inter-node links of the ring. Other criteria such as ease of initialization, recovery from errors, and cost of equipment are also pertinent.

The following analysis is based on the assumption that the average traffic or demands to send messages are symmetrically distributed and equal at each of the nodes.

The delay between the time a component has assembled a message and the time it is acknowledged as received at the destination component consists mainly of three parts:

1. Queueing. Waiting at the sending node until previous transmission requests by the node have been completed.
2. Latency. Waiting until it is appropriate for the ring interface to start transmitting.
3. Transmission time. The time it takes for the message to be transmitted and return to the originating ring interface for verification.

Since the latency and transmission times determine the service rate for the queue, the length of the queue and consequent delays are dependent on the latency and transmission times as well as the rate of arrival of messages to transmit. The latency and transmission times are dependent upon both the mechanism for preventing interference in the transmission of messages as well as the probability of a message to be transmitted at each of the nodes. The probability that there is a message at a node to be transmitted is simply the probability that the queue length is non-zero. Assuming a fixed poisson arrival rate of messages to be transmitted, we find that for arbitrary service times the probability the queue

length is non-zero is inversely proportional to the average service rate. The average service rate must, of course, be greater than the arrival rate. We conclude that the probability that a ring interface has a message to transmit is monotonic with respect to the latency and transmission times.

To compare the mechanisms for preventing interference in transmitting messages we will calculate for each mechanism:

1. The expected latency time when the queue of messages to be sent is empty
2. The time for transmission and validation of a message
3. The expected latency and transmission time (service time) when the queue is empty
4. The service time for a non-empty queue.

Let

n = the number of nodes in the ring

b = length of each message in bits

r = transmission clock rate in bits per second

c = circumference of the ring in bits including propagation delays and 1/2 bit delay at each node

for reforming

p_m = the probability that a ring interface has a message to transmit at the time when it is permissible for it to do so under mechanism m where

m = cp is the control passing mechanism

m = ls is the lazy Susan mechanism

m = md is the message delaying mechanism

Control Passing

1. The latency time for zero length queue with control passing is uniformly distributed between the delay when the control token arrives at the same time as the message to be transmitted and the delay when the control token must pass all nodes.

In the best case

Delay = 0

In the worst case

Delay at each node which has a message to transmit = time for one block = b/r

Expected delay at each node = $p_{cp}\, b/r$

Expected delay at n nodes = $np_{cp}\, b/r$

Delay in transmission = c/r

Total delay = $np_{cp}\,b/r + c/r$

Expected latency delay = $\frac{np_{cp}\,b+c}{2r} = \frac{1}{2rc/b}\left(np_{cp}c + \frac{c^2}{b}\right.$ /

2. The time for transmission and validation of a mes
is the block length divided by the transmission rate p
the delay for propagation around the ring.

Transmission time = $b/r + c/r$

3. The expected latency and transmission time when t
queue is empty is the sum of (1) and (2).

Expected latency and transmission time =

$$\frac{np_{cp}\,b+c}{2r} + \frac{b}{r} + \frac{c}{r}$$

4. The service time for a non-empty queue is the wor
case latency time.

$$\text{Service time} = \frac{np_{cp}\,b + c}{r} = \frac{1}{rc/b}\left(np_{cp} + \frac{c^2}{b}\right)$$

Lazy Susan

With the lazy Susan mechanism there are $\lfloor c/b \rfloor$ slots available for placing messages on the ring.

1. The latency time for zero length queue is $1/\lfloor c/b \rfloor$ times the latency for a single slot. The latency with a single slot is uniformly distributed between the delay when an empty slot arrives at the same time as the message to be transmitted and the delay when the slot must pass by all nodes.

In the best case

Delay = 0

In the worst case

Time to pass each ring interface = 1 + 1/n times around the ring with probability p_{ls} ; 1/n times around the ring with probability $1-p_{ls}$

Expected time to pass each ring interface

$$= p_{ls}\left(1+\frac{1}{n}\right)\frac{c}{r} + \left(1-p_{ls}\right)\frac{1}{n}\,\frac{c}{r}$$

$$= \left(p_{ls} + \frac{1}{n}\right)\frac{c}{r}$$

Time to pass n ring interfaces

$$= (n p_{ls} + 1)\frac{c}{r} = \frac{1}{r}(n p_{ls} c + c)$$

Expected latency delay with only one slot

$$= \frac{n p_{ls} c + c}{2r}$$

$$\text{Expected latency delay} = \frac{1}{\lfloor c/b \rfloor} \frac{n p_{ls} c + c}{2r}$$

2. The time for transmission of a message is the block length divided by the transmission rate plus the propagation delay around the ring.

Transmission time = b/r + c/r

3. The expected latency and transmission time when the queue is empty is the sum of (1) and (2).

Expected latency and transmission time

$$= \frac{1}{\lfloor c/b \rfloor} \frac{(n p_{ls} + 1) c}{2r} + \frac{b}{r} + \frac{c}{r}$$

4. The service time for a non-empty queue is the worst case latency time.

$$\text{Service time} = \frac{1}{\lfloor c/b \rfloor} \frac{(n p_{ls} + 1)\, c}{r}$$

Delay Relaying Messages When Required

1. The zero length queue latency time is distributed between zero delay and the delay of one message time (0,b/r). However, this is not uniform: for a lightly loaded ring it is biased toward zero delay. We will assume the best case of zero delay.

2. The expected time for transmitting and verifying a message is the time for the message to go completely around the ring plus the delays encountered at each ring interface.

Expected transmission and verification time

$$= \frac{b}{r} + \frac{c}{r} + p_{md} n \frac{b}{r} = \frac{c + n p_{md} b + b}{r}$$

3. The expected latency and transmission time when the queue is empty is the sum of (1) and (2).

Expected latency and transmission time

$$= \frac{c + np_{md}b + b}{r}$$

4. The expected service time for a non-empty queue is the expected time for transmitting and verifying.

$$\text{Service time} = \frac{c + np_{md}b + b}{r}$$

$$= \frac{1}{rc/b}\left(\frac{c^2}{b} + np_{md}c + c\right)$$

Since the probability that a ring interface has a message to transmit is monotonic with respect to the latency and transmission times, the service time for non-empty queues is a good measure of the efficiency of the mechanism.

$$\text{Control passing} \qquad \frac{1}{rc/b}\left(np_{cp} + \frac{c^2}{b}\right)$$

Lazy Susuan $$\frac{1}{r\lfloor c/b \rfloor}\left(np_{ls}\,c + c\right)$$

Message delaying $$\frac{1}{r\,c/b}\left(np_{md}\,c + c + \frac{c^2}{b}\right)$$

Control passing is always preferred to message delaying. Comparison of control passing and the lazy Susan is somewhat more difficult.

When c/b is less than one, there is not enough room for a whole message on the ring. In this case the lazy Susan scheme makes no sense. For c/b = 1, the two schemes are identical (in operation as well as in performance). For $1<c/b<2$, control passing is always preferred. For c/b = integer>1, the lazy Susan scheme is preferred. For c/b not integer and greater than 2, the preferred scheme depends on the expected number of nodes with messages to transmit. When the expected number is large, control passing is preferred. When the expected number is small, the lazy Susan is preferred.

By setting the service times equal to each other and solving for np in terms of c/b it is possible to state the trade off.

$$np = \frac{\frac{1}{\lfloor c/b \rfloor} - 1}{\frac{1}{c/b} - \frac{1}{\lfloor c/b \rfloor}}$$

This function is plotted in Figure 1. For np above the traces, control passing is preferable. For np below the traces, the lazy Susan is preferable. The graph shows that for integral c/b where a number of trays fit exactly around the ring an infinite expected number of nodes ready to transmit would be necessary to make control passing preferable. As the size of the ring increases so that the number of trays does not fit well, the expected number of nodes ready to transmit to make control passing preferable decreases. When there is almost a whole tray space wasted, control passing is most favorable. The expected number of nodes ready to transmit at this point depends on the number of slots:

$$np = \lceil c/b \rceil \left(\lceil c/b \rceil - 2 \right)$$

All of the previous analysis has assumed that messages were of a fixed size. In any real system it is likely that the size of messages to be transmitted will vary. If only one fixed size of message blocks can be accomodated some messages will only partially fill blocks and others will need to be transmitted in several blocks. This results in wasted transmission capacity for partially filled blocks and additional overhead for multi-block messages. The problem is quite similar to storage

fragmentation in computer memories.

The amount of waste is dependent upon the distribution of message sizes. If there are many short control messages, the waste will be great. If most messages completely fill a message block, the waste will be slight. For example, if the length of messages is uniformly distributed between 1 and the message block size, about one-half of the transmission capacity will be wasted.

If the communication mechanism allows, this waste can be avoided by transmitting messages which are the exact size needed rather than to fit messages into fixed sized blocks. The only restriction on the size of messages for the control passing mechanism is that messages may not be so big that the use of the ring is monopolized for an unacceptable length of time. The lazy Susan can only accomodate a fixed size block. For this reason the control passing mechanism has an advantage where the size of messages varies considerably.

To complete the analysis we will examine the trade-offs with some typical parameters:

$n \approx 10$

$b \approx 1000$ bits

$r \approx 2$ mHz

$$c \approx \frac{200 \text{ mi}}{2 \times 10^5 \text{ mi/sec}} \times 10 \text{ bits/sec} = 2000 \text{ bits}$$

Service time for control passing

$$= \left(\frac{1}{rc/b} np_{cp} c + \frac{c^2}{b}\right)$$

$$= \frac{5p_{cp} + 2}{10^3} \text{ seconds}$$

Service time for lazy Susan

$$= \frac{1}{rc/b} (np_{ls} c + c)$$

$$= \frac{5p_{ls} + .5}{10^3} \text{ seconds}$$

Both p_{cp} and p_{ls} are between 0 and 1. However, p_{cp} will be larger than p_{ls} since the lazy Susan service time is slightly better.

For fixed length blocks the lazy Susan mechanism is best for this situation. Since c/b is an integer the maximum advantage is given to the lazy Susan. If the control passing mechanism were to also take advantage of

variable length messages, the average message length transmitted might be only 500 bits. In this case the service time is calculated:

$$= \frac{1}{rc/b}\left(np_{cp}c + \frac{c^2}{b}\right) = \frac{1}{r}\left(np_{cp}b + c\right)$$

$$= \frac{10\,p_{cp}\,500 + 2000}{2 \times 10^6}$$

$$= \frac{2.5\,p_{cp} + 1}{10^3} \text{ seconds}$$

In this case control passing is likely to be best.

Maintaining Loop Operation

A system using a single unidirectional communication ring would be extremely vulnerable to total failure since a failure in any link or at any node would prevent all pairs of nodes from conversing. To avoid this, in

addition to the primary links which connect the nodes, secondary links which skip nodes are provided. (See Figure 2.) Upon malfunction of a node or link, the successor node can take its input from the secondary link. By ignoring any data from its primary link, the malfunctioning part of the system has been logically excluded from the system. Such exclusions may occur in many places around the ring. As long as there is not a malfunction at two adjacent nodes or in certain combinations of their links, the communication ring can continue to function. This exclusion should occur automatically while the system is running when a malfunctioning section of the system is recognized.

As a result of this automatic exclusion, the communication ring is relatively secure against failure except when two adjacent nodes malfunction. Two adjacent nodes may malfunction because of external environmental factors or because of randomly coincident malfunctions.

A physical by-pass path, which is independent of the power-on/off status of the ring interface and attached component, could allow connection of the ring interface inputs and outputs. By activating this path the ring could be reconfigured as if the node did not exist. This would be used if the ring interface is known to be

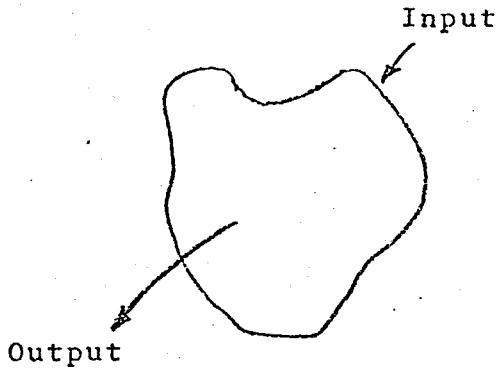
defective, during maintenance, and when the computer is not programmed to operate on the ring. The path could be invoked automatically upon loss of power at the node.

The node skipping links in conjunction with the by-pass paths at each node provide a means of attaining high reliability while excluding only the malfunctioning node from the ring. Strategically placed links which skip more than a single node could be used to gain even more reliability by allowing operation of the ring even when there are multiple, contiguous malfunctions. Use of these paths may involve exclusion of properly functioning nodes as well as those which have failed. However, the effects are limited to a locality of the ring.

We now turn to the questions of what constitutes an acceptably functioning section of the ring and when to exclude a section which appears to be malfunctioning. We are only interested in those malfunctions which affect the whole system or major portions of it since systems can continue to operate with slightly decreased performance when some components are not available. Failures which afect only a single node, while they must be recognized and corrected, are of secondary importance to a malfunction which prohibits proper operation of the whole system.

Because of this it is important only that a ring interface be able to properly relay messages and maintain the transmission protocol, whether it can successfully accept messages destined for it or can orginate messages is inconsequential.

We are not concerned with the internal operation but only the relation between what is received and transmitted. Ignoring for the moment any node skipping links, the node can be thought of as an open subsystem with an input from the previous node in the ring and an output to the successor node in the ring.



A portion of the ring consisting of several nodes can be represented by the same subsystem: the input going to the first node and the output from the last node with internal connections not observed. The only difference between multiple node subsystems and a single node is the amount of delay between input and output.

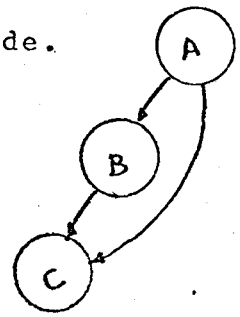
With the control passing protocol a correctly operating subsystem must:

1. Relay bit by bit exactly what was received except for (3) and (4).
2. Emit a control token sequence within a bounded time after a control token sequence is received. (This bound is based on the number and length of messages allowed for (4).)
3. The subsystem may transmit its own messages between the time it receives the control token

sequence and the time it sends the control token sequence.

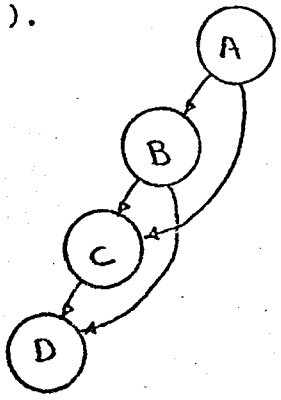
4. The subsystem need not relay from the time it began to emit the control token sequence for the time it takes information to propagate around the ring. This period corresponds to the time that messages transmitted by the subsystem are returning after traveling completely around the ring.

Circuitry to implement this test can be located at the successor of each node. This circuitry could use as its inputs the signal received from the previous node and the signal received on a node skipping link from the second previous node.



Two problems arise in this comparison. If there are differences in the two signals received, it is not possible for node C to tell, a priori, whether the errors were induced in the primary path or in the node skipping

link. Also use of the information from the alternate node skipping link would be viewed as a malfunction by the next node in the ring (D).



It would detect a discrepancy between the two signals received but would not be able to differentiate between a valid exclusion operation and a malfunction of node C.

If in addition to the links from the previous ring interface and the second previous ring interface, each ring interface receives the output of the third previous ring interface via another link, it can compare these and make a correct decision on whether to accept the data from the previous ring interface or from the second previous interface.

Let I_1, I_2, and I_3 designate the inputs from the previous, second previous and third previous nodes respectively. Let C(x,y) = 1 if the subsystem with input x and output y is behaving according to the rules given previously, and C(x,y) = 0 otherwise. Then the

appropriate input from which to accept data is given:

	$C(I_{-1}, I_{-2})$	$C(I_{-1}, I_{-3})$	$C(I_{-2}, I_{-3})$	appropriate input
1.	1	1	1	I_{-1}
2.	0	1	0	I_{-1}
3.	0	0	1	I_{-1}
4.	1	0	0	I_{-1}
5.	0	0	0	I_{-1}

This table can be derived:

(1). If everything is working, the appropriate input is the output of the previous ring interface.

(2). If the previous node fails, both the subsystem for the previous node and both previous nodes will not be behaving correctly so it is appropriate to exclude the previous node.

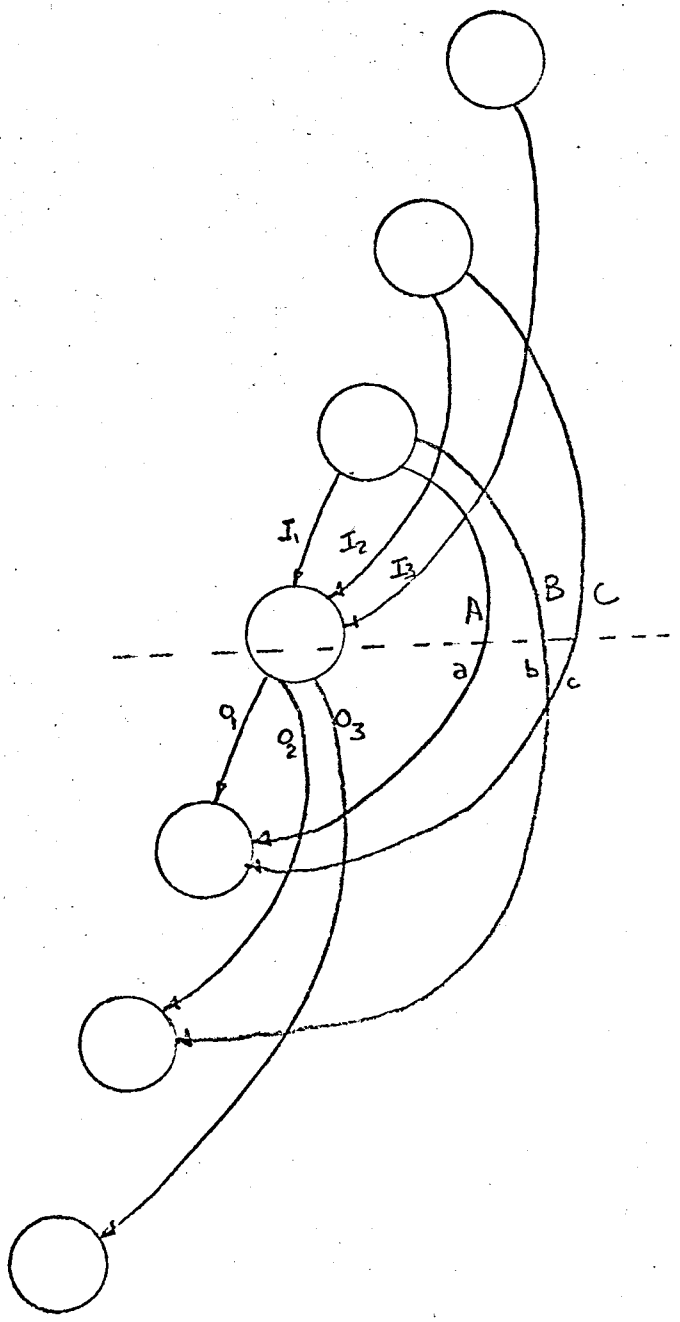
(3). If the signal from the second previous node indicates an error but the subsystem containing both previous nodes is behaving correctly, the appropriate input is from the previous node.

(4). If the second previous node does not appear to be working correctly, assume it is excluding a node but is

being monitored by the previous node and accept the input from the previous input. If none of the subsystems are working correctly the previous node is malfunctioning or else recovery is impossible so exclude the previous node. Note that it makes no difference whether a node itself has failed or a path connecting the node has failed.

To reconfigure the ring so that the ring operates as if a node were not present the following bypasses are necessary.

Normal configuration By-pass connection



I_1 to O_1
I_2 to a
I_3 to c
A to O_2
B to O_3
C to b

Conclusions

In the preceeding analysis we have determined that it is generally best that the communication ring use:

1. High speed full bandwidth data transfers
2. Buffering where necessary at the origin and destination nodes
3. A control passing protocol
4. Redundant links.

Design details for a ring and ring interfaces using these principles are described in appendix A.

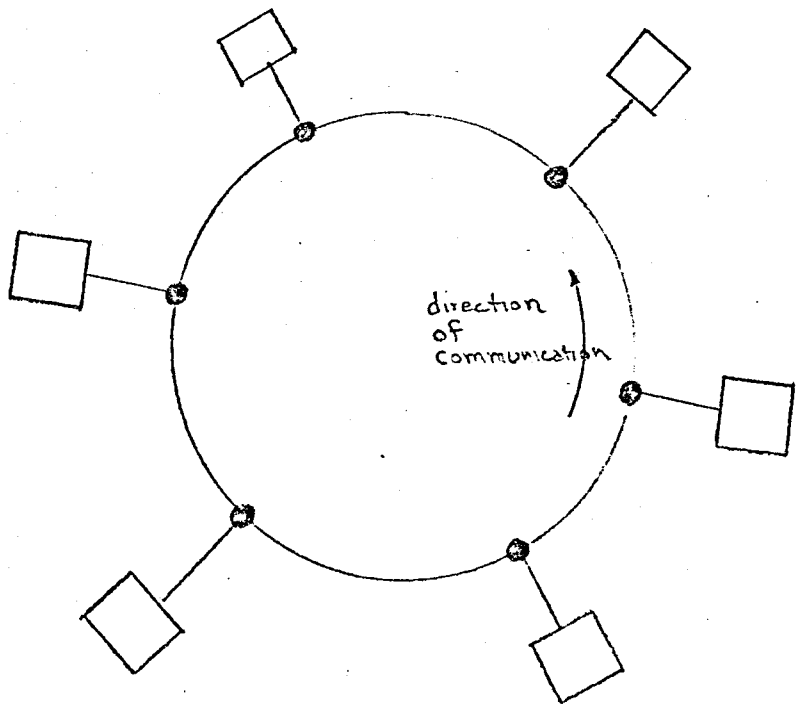
Appendix A

Ring Interface Hardware

DISTRIBUTED COMPUTER SYSTEM

University of California, Irvine

A ring interface provides for the connection of a computer or other computer system component to a unidirectional communication path. The use of several ring interfaces interconnected with a circular communication path results in a looped communication path which any of the components can use to send messages to any of the other components:



The functions of the ring interfaces are

1. Control use of the communication ring
2. Route messages from a component onto the communication ring
3. Route messages from the ring to the appropriate component.

A component is a computer or other computing system component which is connected to the communication ring with a ring interface. A component and its ring interface taken together constitute a node of the system.

Information is transmitted among components in packets called messages. A message may contain up to 65536 bits of information (8K bytes). Messages are addressed with logical addresses which are independent of the physical locations of nodes. Many logical addresses can be dynamically assigned to each node at the same time. The same logical address may be assigned to more than one node. When an address is assigned at more than one node, the components at both nodes will receive messages with that destination address. A unique logical address can be assigned for each process. If the process is moved to another component on the ring, its address (process name) can be moved to the new node so that senders of messages need not be aware it has moved.

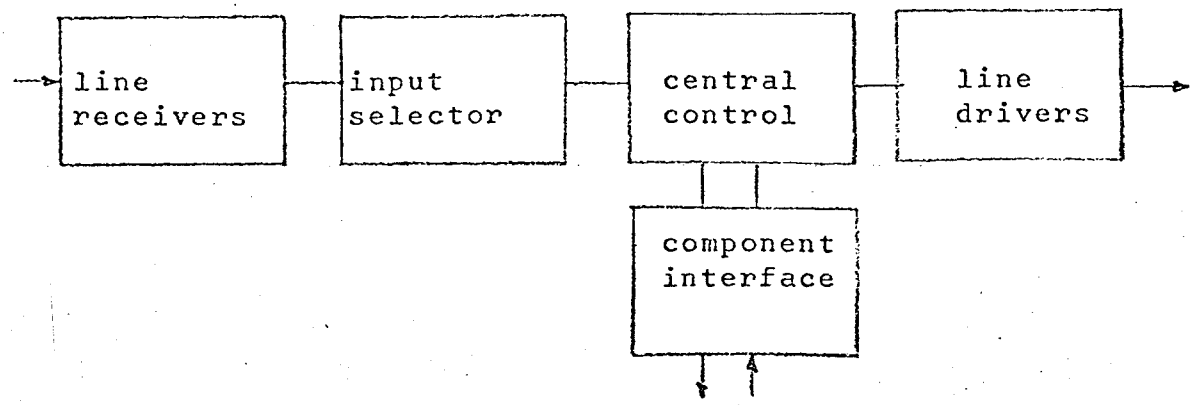
All messages travel on the ring from the ring interface at which they originate, all of the way around the ring, back to the originating ring interface. As a message passes through each ring interface where the destination address in the message matches one of the logical addresses of the node, the message is copied into the component. All comparison and copying occurs immediately as the message arrives at the ring interface so that it can be transmitted to the next ring interface bit by bit without delay.

Only one ring interface may be placing a message onto the ring at any one time. When it is through transmitting its message, a ring interface sends a special sequence of bits known as a control token. Any node which has a message to send must wait to see this sequence at the end of a message, alter the sequence to indicate the beginning of a message and transmit its message with a control token sequence affixed to the end. In this way only one ring interface will be transmitting at a time. With only one ring interface transmitting at a time the possibility that a message will be replaced with another message before it has circulated completely around the ring is avoided. Although only one ring interface will be transmitting at a time, there may be more than one message which is

traveling on the ring if the delay around the ring is large.

The ring interface consists of 5 logical parts.

1. Line receivers
2. Input selection
3. Central control
4. Component interface
5. Line drivers.



The line receivers accept the incoming ring communication signals and convert them from the communication medium being used to clock and data signals at internal logic levels.

The input selector provides for the selection of one of several incoming signals from other ring interfaces. Selection of an alternate incoming signal allows for the exclusion of malfunctioning components from the ring.

The central control recognizes incoming messages destined for the attached component and controls the data to be sent on to the next node.

The component interface transfers messages from the central control to the attached component. It also transfers messages from the component to the central control. The component interface is tailored to the requirements of the component being attached. Other parts of the ring interface are independent of the component. Normally the component interface will connect to the

computer data channel or direct memory access port and data transfers will occur at the data rate of the communication ring. For components not capable of operating at this rate buffering must be supplied.

The line drivers provide for conversion of the output of the ring interface to the appropriate signals for the ring communication medium.

Operation

Message Transmission

When a component has a message to send, it assembles the message in its memory in the following format and signals the ring interface.

destination address	origin address	length of text	text (including a sequence number)

The ring interface enters the waiting-to-send state until a control token sequence follows a message. It alters the final bit of the sequence to indicate the start of a message and begins transmitting the message it obtains

from the computer through the component interface. At the end of the text a 16 bit cyclic redundancy check code is transmitted followed by two bits for other interfaces to return match and accept status followed by a control token sequence. The ring interface will ignore all input received until the approximate time it expects the beginning of its message to have traveled completely around the ring. (This estimate of the transmission delay around the ring is adjusted every time a control token passes by observing the time it takes before either the control token or a message which replaced it returns.) Slightly before the expected time for the return of the message, the interface begins to look for the start-of-message sequence for the message it sent. When this is found the interface checks the following message's cyclic redundancy check code to assure no errors were introduced into the message. It then interrupts the sending component to indicate completion of the function. The transmission status flags are then available for interrogation by the component. If the start-of-message does not occur as expected, the ring interface will interrupt the component with the loop-error status set.

Message Transmission Status Flags

(available after end-of-transmission interrupt until the next request to send a message)

Flag name	Value	
	1	0
Transmit Overrun	Message transmission was aborted because data was not available at required rate	Message successfull transmitted
Loop Error	Message was not recognized at originating node	Message returned to originating ring interface
Transmit CRC Error	CRC in message when returned to originating node indicated error	CRC in message when returned to originating node indicated no error
Match	At least one ring interface responded to the logical destination address but could not copy the message	No ring interface responded to the logical destinatior without copying the message
Not accepted	No ring interface copied the message	At least one ring interface copied the message

The loop error status must be zero for the other indicators to have meaning. The match and accept flags

are not error checked and therefore are likely to be erroneous. They should not be used if the CRC error flag indicates there was an error in another part of the message.

Message Reception

Each ring interface has a table of logical destination addresses which it should recognize. (The entering of addresses into this table is described below.) For each message which passes on the ring, the destination address is compared bit by bit with each of the addresses in the table. If any of the addresses completely match this destination address, the destination address, origin address, length, and text will be routed through the component interface to the component. The component will then be interrupted and can interrogate the receive status flags.

Receive Status Flags

(available to be tested after receive interrupt until component initializes I/O to receive next message)

Flag name	Value	
	1	0
Receive Overrun	Part of message was lost because component did not accept data at required rate	Entire message was copied into component memory
Receive CRC Error	CRC in message indicated error	CRC in message indicated no error

All messages received by a ring interface are transmitted to the next ring interface with only the delay necessary to reshape each bit. If the destination address in a message matches one in the ring interface's table but the message was not successfully copied into the component's memory, the match bit will be set to a one regardless of the value received. If the destination address in the message matched one in the ring interface's table and the message was successfully copied into the component's memory, the accepted bit will be set to a one regardless of the value received.

When a control token is received by a ring interface which has no message to send, it will transmit the control token to the next ring interface. It will cease to relay until the control token or the message which replaced it arrives. The estimate of the transmission delay around

the ring will be used to commence looking for the control token or start-of-message. The error in this estimate will be used to calculate a new estimate.

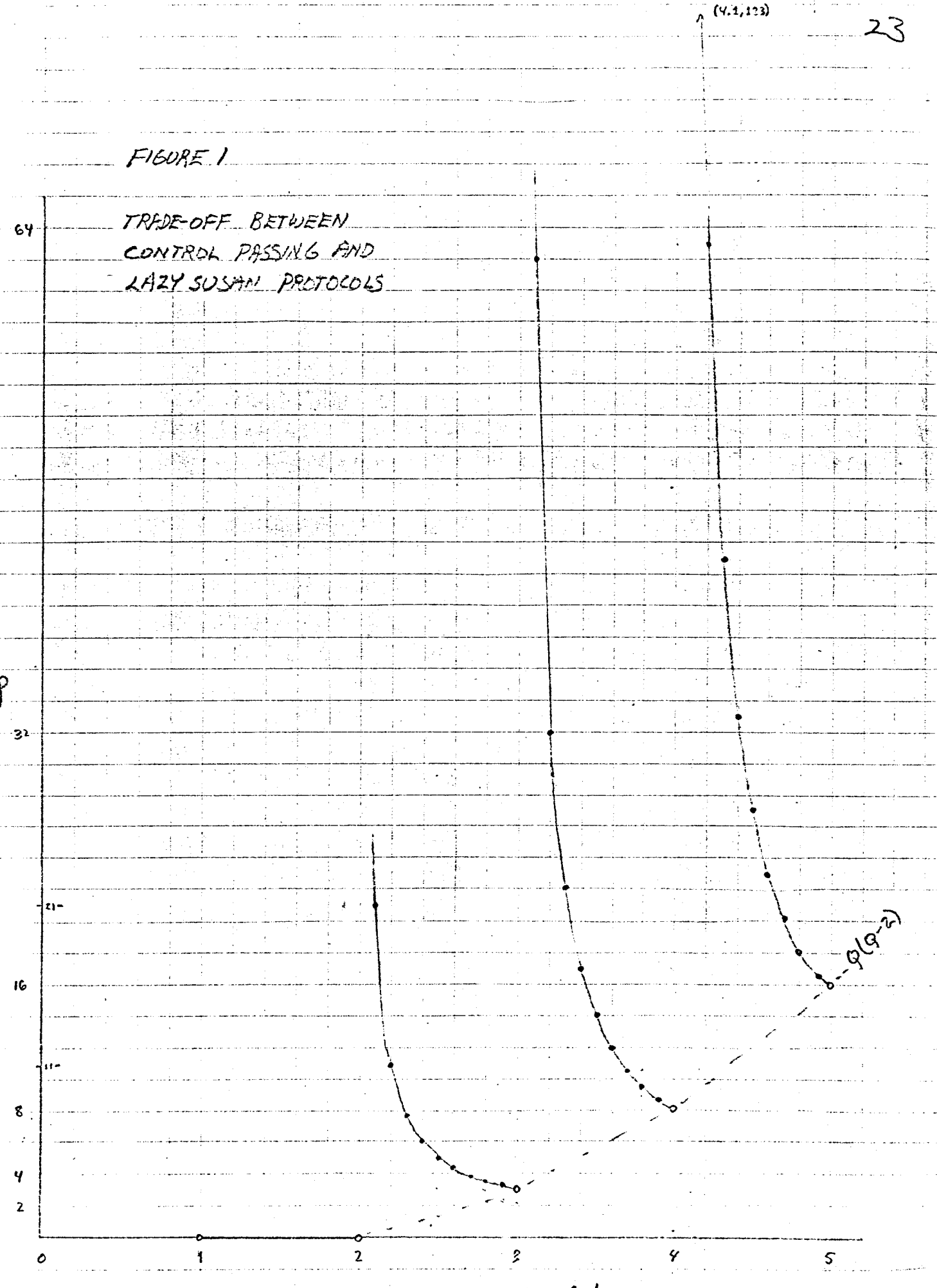
Loading the logical address table

An entry to be placed in the logical address table must be constructed in the component's memory in the following format:

position in table	logical address

The component will signal the ring interface to replace the previous value of the designated position with the new value. Since control of the logical address table must be kept in synchronization with messages being received on the ring, a delay may occur before it is possible to load the designated position. When the position can be loaded, the ring interface will do so and then cause a table loaded interrupt.

FIGURE 1


TRADE-OFF BETWEEN
CONTROL PASSING AND
LAZY SUSAN PROTOCOLS
(4.1, 123)
64
32
21
16
11
8
4
2
A=np
0
1
2
3
4
5
q(q-2)
n=4

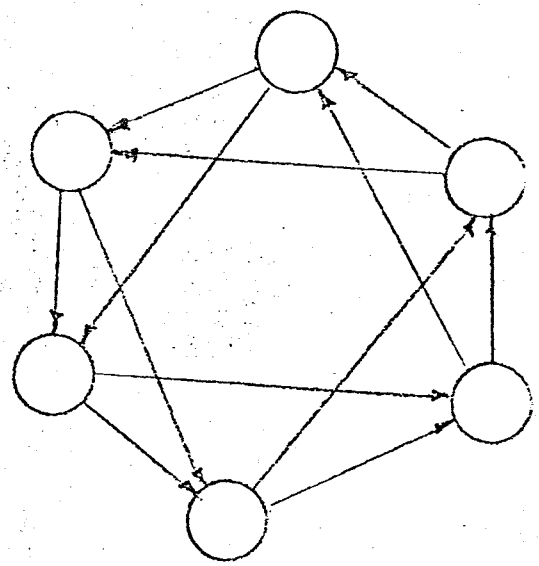


Figure 2

Message in Component Memory

destination logical address	origin logical address	text length	text (including sequence field)
16 bits	16 bits	16 bits	variable

Message As Transmitted

timing preamble	start of message	destination logical address	origin logical address	text length	text(including sequence field)	CRC	match flag	accept flag	timing preamble	control token
16 bits	6 bits	16 bits	16 bits	16 bits	variable	16 bits	1 bit	1 bit	16 bits	6 bits

Note:
1. The length field is coded as a binary number which is one less than the number of bits in the text.
2. Timing preamble is all zeros.
3. Start of message is 000011
4. Control token is 000010
5. CRC is calculated using polynomial $1 + x^2 + x^{15} + x^{16}$