1976

Shipboard application of a ring structured distributed computing system.

Jackson, Jeffrey Quentin

Monterey, California. Naval Postgraduate School

SHIPBOARD APPLICATION OF A RING STRUCTURED DISTRIBUTED COMPUTING SYSTEM

Jeffrey Quentin Jackson

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

SHIPBOARD APPLICATION OF A RING STRUCTURED DISTRIBUTED COMPUTING SYSTEM

by

JEFFREY QUENTIN JACKSON

Thesis Advisor: G. M. RAETZ

June 1976

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Shipboard Application of a Ring Structured Distributed Computing System		5. TYPE OF REPORT & PERIOD COVERED Master's thesis; June 1976
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Jeffrey Quentin JACKSON		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		12. REPORT DATE June 1976
		13. NUMBER OF PAGES
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, CA 93940		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

distributed computing system, ring communication network
fault tolerant computing systems

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Considerable research is currently going on into the application of distributed computing systems. They appear particularly suitable for the computing needs of a small warship. The particular constraints of the warship's environment are discussed. This is followed by a description of how a ring structured distributed computing system might be adapted to function in this environment. Included in this consideration are the feasibility of attaining adequate bus speed, the use of multiply addressed messages, and methods of handling real-time processing. Of particular interest is the ability to achieve controlled degradation of performance under failure, especially failure due to battle damage.

SHIPBOARD APPLICATION OF A RING STRUCTURED DISTRIBUTED COMPUTING SYSTEM

by

Jeffrey Quentin Jackson
Lieutenant-Commander, Canadian Forces
Bachelor of Science (Engineering Physics)

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

ABSTRACT

Considerable research is currently going on into the application of distributed computing systems. They appear particularly suitable for the computing needs of a small warship. The particular constraints of the warship's environment are discussed. This is followed by a description of how a ring structured distributed computing system might be adapted to function in this environment. Included in this consideration are the feasibility of attaining adequate bus speed, the use of multiply addressed messages, and methods of handling real-time processing. Of particular interest is the ability to achieve controlled degradation of performance under failure, especially failure due to battle damage.

TABLE OF CONTENTS

LIST OF FIGURES

I. INTRODUCTION

With the advance of large scale integration technology and the resulting decreasing cost of digital processing elements, the concept of using a number of small processors tied together by some form of data bus, rather than using one large computer, has been receiving increasing interest. The advantages of this type of system include greater flexibility in the system, lower costs, and increased reliability. Distributed computing systems may gain some or all of these advantages depending on the hardware and software utilized to bind the processors together into the system.

The advantages mentioned would be particularly useful for application to a warship's computing requirements. Perhaps the foremost advantage in a world of rapidly changing technology that makes warships virtually obsolescent before they can be made operational, is the ease with which a properly designed system could be reconfigured. When techniques for dynamically reconfiguring the system in the event of normal failure or battle damage are considered as well, the concept of a distributed computing system appear particularly attractive.

Farber [5,6] has been investigating ring structured distributed computing systems for several years. In reference 5 he discusses the concept of "fail soft" systems, that is, systems which exhibit the property of controlled system degradation, rather than catastrophic failure, as a result of component failure. In addition, a prototype ring utilizing many of these principles and employing microprocesser technology has been designed at the Naval Postgraduate School (NPS) [1]. It is not the intent here to reiterate arguments regarding the overall advantages of a ring structured system with respect to reliability and flexibility. These are covered in the references. What is intended is to investigate the modifications and extensions to such a system which would be required to adapt it to the computing needs of a small warship.

A. BASIC CONCEPTS OF A RING STRUCTURED DISTRIBUTED COMPUTING SYSTEM

The heart of the distributed computing system is the method of communication between the processors. To gain the maximum in flexibility and fault tolerance the system should:

1. avoid centralization of control,

2. permit communication between processes without regard for physical location of the process,

3. permit execution of processes without regard for their physical location.

The last two ease the job of dynamic reconfiguration of the system while the first is obviously necessary to prevent the failure of a particular component from bringing the whole system to a halt (catastrophic failure). To achieve these two goals the following basic system is proposed.

The communication network consists of a unidirectional ring to which a processor is attached via a ring interface. Only the ring interface in control can transmit a message.

On completion, control is passed to the next ring interface. Thus, control is decentralized. Independent timers in the interfaces ensure that no one ring interface monopolizes the ring and that if control is lost, it will be regained by one of the other ring interfaces.

The indication that a ring interface may take control is the receipt of a special message called a control token. When an interface has no message to transmit, it simply passes the token on to the next ring interface. If it has a message to send, the control token is not retransmitted. Instead, the message is placed onto the ring, is passed completely around the ring and removed by the originating ring interface. The control token is put back onto the ring after the end of the message.

Two tokens, the start of message (SOM) and end of message (EOM), are used to delineate the data being sent. The SOM is followed by the name of the addressee. The data comes next and then the EOM token followed by several bits used to check for the proper receipt of the message.

Each ring interface has a list of the processes which are operating in its host processor. Messages are addressed to these processes. If the ring interface finds a match between the address on an incoming message and one of the processes in its list, it passes the message to its host processor as well as passing it on around the ring. Status bits appended to the end of the message as it is passed on indicate to the originating ring interface whether or not a message has been matched and accepted during its curcumnavigation of the ring. This accomplishes the second objective, communicating between processes regardless of their physical location.

Finally, resource allocation is accomplished by a system of requests for bids. If a new process needs to be set up, a request for bids describing the resources required is sent around the ring. Each processor having those resources will reply with a bid giving its present resource availability state. The requestor of the bids will then select the most appropriate bid and confirm the new allocation with the accepted bidder. Hence there is a method of dynamic reallocation of resources.

This, then, is the basic distributed computing system which will be considered for adaptation to the small warship's needs. A more detailed description of this design can be found in references 1 and 2.

B. HARDWARE CONSIDERATIONS

As can be seen from the previous discussion, the heart of the ring structured distributed computing system is the ring interface. It must have the capability to recognize the various tokens, maintain lists of processes, and convert the sequential data format of the ring to the format required by its host. The basic difference between the rings designed at NPS [1] and at the University of California at Irvine [6] is the technology employed in construction of the ring interface.

The ring interface at Irvine is hard wired. As such it enjoys the advantage of greater efficiency of design and hence higher data transmission rates. However, it loses in flexibility and standardization. It is designed to interface with one particular host and to adapt such a ring interface to a different host is virtually impossible. Moreover, the resulting differences in ring interfaces designed for different hosts can pose a maintenance problem.

The ring interface at NPS is designed around a microprocessor. The ring speed in this case is limited by the firmware of the programme stored in its programmable read only memory (PROM) and the access time of this memory. For the greatest flexibility the NPS ring employs an erasable PROM. Thus if the ring protocol were to be changed or a new host required a different format for exchange of data, the microprocessor memory could br erased and a new set of firmware written. This entails a sacrifice in speed, however, as erasable PROM's have slower cycle times than non-erasable ones. By using a non-erasable PROM the speed can be increased but a change would entail replacing the memory with a different chip containing the new programme. In both these cases the microprocessor has the advantage of standard hardware for the ring with only a variation in the firmware to accomodate the various hosts.

Flexibility is of particular value in the shipboard application where a large number of the hosts may be "dumb", such as servo system controllers or monitoring devices, or may be in existance at the time of system design. Because of the advantages in maintenance and flexibility, the microprocessor technology will be considered here. The problems of ring data transmission rates will be addressed again after the requirements of the shipboard application have been discussed.

II. ANALYSIS OF SYSTEM REQUIREMENTS

A. EXTENT OF SYSTEM

The first question which must be addressed when considering system requirements is the extent of the system. What functions of a warship should be included for processing in the distributed computing system? There are some areas that are fairly obvious, i.e. those that are included in current command and control systems. These include evaluation and processing of sensor information (radar tracking, electronic warfare evaluation); tactical data processing (maintenance of track files, files of standard tactical procedures, etc.); tactical communications (LINK); fire control calculations; tactical displays; navigation routines (ship's position, closest point of approach calculations, course to intercept, etc.). But there is no reason to limit it to these.

Digital computer control of main engines and auto-pilots are already in existence. Including these in the distributed computing system would facilitate automatic control of emergency and evasive manoeuvres e.g. man-overboard, torpedo evasion; and also the automatic implementation of tactical manoeuvres e.g. lost contact searches, zig-zag courses, station keeping. Remote monitoring and automatic alarm systems for auxiliary machinery exists as a subsystem of the main machinery control. Expansion of this to monitoring other ship's conditions, smoke alarms, flooding, etc. , and making it part of the distributed computing system would provide an effective and flexible damage control monitoring system.

Finally, communications is a fruitful area for automation. Tactical data is already encoded, transmitted, decoded, sorted, displayed, and stored automatically. The problems of extension to the remaining communications nets do not seem overwhelming and the possible savings in personnel, paper and time appear to be worth-while. In addition, the use of a computer for frequency assignment algorithms would be a distinct advantage and if this were coupled with direct control of transmitters and receivers, electromagnetic emission control would be greatly facilitated. Making this computer part of the ship's distributed computing system would allow such problems as the efficient handling of priority traffic to be easily attacked.

Such an extended system is under development by the Canadian Armed Fordes under the title of "Shipboard Integrated Processing and Display System" [7,9]. It is the adaptation of a ring structured distributed computing system to the requirements of this system that will be considered. Most of the requirements in the following sections are based on the preliminary requirements for the Canadian system [9] and are in agreement with these requirements.

B. DETAILED SUBSYSTEM REQUIREMENTS

1. Displays

The display requirements for various departments on a ship vary widely. Communications can be satisfied with purely an alpha-numeric display. To the engineering department, a graphics capability would be a desirable feature. The greatest demand undoubtedly comes from the tactical displays which are required to present raw video from various sensors as well as the synthetic graphical and alpha-numeric data that are produced by the various processors, i.e. smoothed tracks, track designators, messages, etc.

All these requirements will affect the decisions with respect to display technology, i.e., what are the advantages and disadvantages of raster scan versus vector generation versus dot matrix presentation, etc. However, with respect to the system design of a processing and display system, most of this is irrelevant. What is important is the "intellegence" of the display, for it is this characteristic that will determine the type and quantity of data that must be passed to a display device to maintain the presentation desired by the operator.

An allied aspect of display technology is the method of maintaining data on the display - whether a long persistence cathode ray tube should be used or the data should be frequently refreshed. The latter is the only practical solution for any sort of dynamic display and for the rest of the discussion it will be assumed that the display must be refreshed at least 30 times a second. Refresh rate then provides a critical frequency in the implementation of the display system.

An important differentation can now be made in the types of data which are to be displayed, i.e. those data which may be expected to remain constant for more than one thirtieth of a second and those that may not. For the latter it makes no difference whether the display is "intelligent" enough to refresh itself or not. The data will have changed in the interval between successive refresh cycles.

For the less volatile data it is a distinct advantage for the display to be able to store the data for the present picture. It is then possible to send to the display only changes to the existing picture. Otherwise, the data for a complete picture must be transmitted to the display every refresh cycle.

Imposing this distinction on the types of data the ship's system must display, raw sensor data fall into the volatile category while the synthetic data are more stable. While some displays are required to present raw sensor data, all have a requirement for presenting synthetic data. It will be assumed then that self-refresh is a desirable characteristic and will be incorporated into displays used in the distributed computing system.

While the desirability of a self-refresh capability is fairly easy to show, the case for further increases in display intelligence is not so clear cut. Should the display have the ability to manipulate data? At one extreme one can conceive of a tactical display that would contain all synthetic data out to the maximum range it can accomodate, and all requests for lesser ranges, filtered data, etc. could be handled within the display itself. This would allow all update messages to be broadcast to all displays rather than having to be tailored to each display's current presentation and, hence, individually addressed. Also, such a display would require very little in the way of communication from the display to the system since there would be no need for the system to be aware of each display's particular presentation. It would, however, require a large memory and considerable computing power within each display. At the other extreme might be a simple routine so that several sequential iterations of data (sonar range sweeps, frequency scans) could be displayed such that as each iteration was displayed the remainder move up and the oldest is discarded. Intermediate between these two extremes might be the ability to identify groups of data on the screen and move the groups by incrementing the position.

It is apparent that actual data rates are very dependent on the type of data to be presented and the amount of processing that can be done within the display. The NATO Industrial Advisory Group [8] is using an average of 16,000 bits per second as the communication requirement for a self-refreshing display.

2. Active Sensors

The active sensors employed in a modern warship are radars and active sonar. Even in a small warship there could be a minimum of four to six radars; search, fire control, and navigation; as well as one or two active sonars, hull mounted and variable depth. The raw data rate of a radar set can be determined theoretically by the sampling rate required to capture all the information available in the bandwidth of the intermediate frequency amplifier chain. Since the IF bandwidth is in turn a critical function in the optimization of the design of the radar, the data rate can ultimately be related back to the operational parameters of the radar. For a typical search radar, pulse width one microsecond, the IF bandwidth is one MHz and the data rate required to ensure no loss of data is thus 1.5 to 2 million bits per second. A high definition radar with its shorter pulse width would obviously require a higher rate. Sonar, due to the more liesurely velocity of propogation of its energy, has a much lower raw data rate of 85,000 to 200,000 bits per second.

It is apparent that to multiplex one, let alone several, active sensors on a general usage data bus would impose a rather severe load unless the bus speed were very high. However, the requirement to display raw sensor data for evaluation by an operator must be met.

Next to be considered is the effect of processing the raw data. The first level of processing would appear to be the automatic detection of contacts. This would involve passing only data on signals that exceed a certain level. Here again a typical situation will be postulated to obtain an idea of required data rates. Assume a radar rotating at 60 rpm and the possibility of 200 valid contacts. To obtain a reasonable probability of detection, the threshold level for signals to be accepted must be set fairly low resulting in a large number of false alarms which must be filtered out by sweep to sweep correlation. Thus, the number of contacts accepted per scan would be in the order of 500 to 1000. On each of these a minimum of range and bearing, and sweep or time data must be passed. Even if only 10 bits of precision are used, the resulting bit rate reaches 15,000 to 30,000 bits per second.

Finally, the data rate resulting from auto-track processing can be evaluated. Here, only valid contact data are transmitted and these would include target identification, x and y coordinates, time, course and speed. Additional data need only be sent when there is a change. Total data per track may be in the order of 200 bits but an update message would be considerably less than that, and would be required much less often than once per track per scan. Although the instantaneous communication requirements would be much more variable, the average rate would be much lower, something in the order of 4000 bits per second for the assumed criterion of 200 valid tracks.

The discussion of processed data so far has been limited to radar data but it should be apparent that this is the limiting case. Sonar contacts would be handled in the same manner but the expected number of underwater contacts is in the order of one or two, and ten or more would be unlikely. The communication load would then be in the order of tens of bits per second.

There is one further area of processing of active sensor information that needs to be considered, that of multi-sensor correlation. While autodetection might be carried out with a microcontroller, auto-track processing has a sufficiently great computing requirement that a minicomputer would reasonably be employed. Why not then extend the process to compare the contacts of various sensors to determine if the same object had been detected by more than one sensor? With appropriate processing and feedback to control the sensitivity of the sensors, greater detection probabilities can be obtained in addition to the immediate result of providing more accurate information by combining the data from independent sources. This would have the additional benefit of further reducing the traffic on the communication net since redundant information on the same contact but originating from more than one sensor would be eliminated. However, the magnitude of the reduction would be quite small, particularly in relation to those obtained in going from autodetection to auto-tracking.

3. Passive Sensors

Passive sensors include acoustic detectors as well as electromagnetic energy detectors for frequencies from that of radar to that of visible light. The latter can be broken into two groups on the basis of the characteristics of their output data. Electronic warfare sensors cover the frequency bands from roughly one to fifty GHz. There is a considerable amount of preprocessing done and the raw data are of little or no interest. The second group, optical systems, produces a linescan video image as raw output. Included in this group are low light level television and infrared sensors. As the data from these sensors is compatible, processing may include multi-sensor correlation as well as digital enhancement of the image. The computing requirements for this are rather high and would probably require a dedicated minicomputer.

Now to consider each of these three sensor groups individually. Passive acoustic data is normally presented for human evaluation after preprocessing to obtain sound intensity versus frequency versus time information. The most common presentation method is as an intensity modulated frequency sweep with the past data being shifted along the time axis as each new frequency sweep is started. An alternative coming into greater use is to construct a three dimensional graph with the time axis appearing to recede into the screen. The latter would change less often, a new sweep being added every 2 to 5 seconds, whereas the intensity modulated display would initiate a new sweep 4 to 8 times a second.

If a systematic shifting routine were available, as discussed in section II.A.1, the update would require in the order of 5000 bits of information for the intensity modulation and 300,000 for the graphical plot. The resulting data rates would then be 20,000 to 40,000 bits per second for the intensity modulated method (5000 bits 4 to 8 times a second), or 60,000 to 150,000 bits per second for the graphical plot (300,000 bits every 2 to 5 seconds). Having to update the whole screen each time would increase the data rate to several million bits per second.

The optical systems present a volatile image that will change at a rate higher than the 30 Hz refresh rate.

Here the criterion of IF bandwidth can be applied as in the radar system and data rates of several million bits per second result.

In the case of both passive sonar and the optical systems much further processing can be done, including correlation with contacts held on other sensors both active and passive. The communications load produced as a result of this is of the nature of the update and amplifying material on valid contacts, as discussed in section II.A.2. Thus a data rate in the order of 10 to 100 bits per second would be reasonable.

E.W. sensors, as mentioned, produce a highly preprocessed output with the raw data being of little use. The computing requirements are quite high, undoubtedly requiring a dedicated processor. However, the communication requirement would again be that of passing update material and 500 bits per second would be adequate.

4. Weapons

The weapons outfit of a small warship would consist of a mix of missile launchers and guns and "soft weapons" such as jammers and chaff launchers. For remote control of servo devices data is usually supplied 32 times per second. Even for the most complicated launcher a few hundred bits of data should provide adequate information at each update and 100 bits would be a reasonable average. This results in a 3200 bits per second average rate for each weapon and even with 8 to 10 weapons operating simultaneously the total average rate would be 32,000 bits per second.

It must be considered, however, that these are real time systems and the timing of the messages is critical. It will therefore be necessary to devise a method of ensuring that weapons control messages do not get delayed i.e. left waiting in output queues.

5. Communications

Communications has one area in which the needs are already well defined. Computer controlled tactical data communication via LINK 11 has been in use for several years and 15,000 bits per second could be considered a reasonable estimate of bus loading by this system [4].

The requirements for automated handling of general message traffic are more obscure. Data exist on the number of messages per day and can be broken down by priority and classification. Storage requirements can thus be easily determined. Four million bits per day should suffice and two days' messages should be immediately retrievable [9]. Messages more than two days old can be relegated to off-line storage.

The effect on data bus loading is considerably less clear. It can be reasonably assumed, however, that with the exception of priority messages, general traffic will be read in periods of relative leisure, hence, in periods of low system activity. The effect of this on bus loading should then be minimal. Priority messages would go out on the bus immediately but this type of traffic is measured in messages per hour if not per day and in terms of bits per second would seem to be insignificant.

There is one other consideration and that is the case where the direct-access mass storage for current messages is a separate node on the bus. In this case all message traffic would put a load on the bus as it was transferred to the mass storage device. Most traffic is teletype at 120 characters per minute or 16 bits per second. Even with simultaneous sending and receiving on several frequencies, the bus loading would be in the order of 100 to 200 bits per second.

6. Propulsion Control

As mentioned previously, the main engines of some ships are already controlled by digital computers, and many ships are steered by autopilots. The step to digitizing the engine and steering orders and routing them via the ship data bus is a small one but opens up the possibility of many desirable features. Instead of having someone drive the ship around a lost contact search pattern displayed on his computer terminal, the computer can do it automatically. If a torpedo is detected on sonar, evasive action could be initiated automatically or by human intervention with automatic execution.

The cost of this would be rather small. The frequency and length of propulsion commands is very low, 20 to 50 bits not more often than every 2 to 5 seconds. Therefore, peak loading would conceivably be 50 bits per second.

The machinery control computer would itself need to monitor status information from the machinery. This would involve in the order of 100 points per second with 8 to 12 bits of data apiece [7]. These 1200 bits per second would not produce much of a load on the bus, however, for other reasons it may not be feasible to use the ship's bus for this information. There is obviously a need for a great deal of autonomy in the propulsion control system. As with no other, it is critical to the survival of the ship. The advantages in making it part of the ship's distributed computing system are features that are nice to have but not critical. There is, therefore, a strong argument for dedicating the propulsion system completely and only using the bus for these non-critical features, particularly since it would still permit the excess capacity in the machinery control computers to be used by other systems.

7. Ship Monitoring

Under this heading will be included all the various miscellaneous devices throughout the ship that provide information on the state of the ship. Included are existing devices such as gyro-compasses and stable elements, and other things which may not be remotely monitored at present, such as the state of auxiliary machinery, hull and fire pumps and air conditioning units, and emergency warning devices, flooding indicators and smoke alarms. There could conceivably be from several hundred to several thousand such devices. Some small number, less than 20 and including log and compass, would require several bits of information and monitoring several times a second. The vast majority would require 1 bit and monitoring every few seconds to few minutes. With appropriate preprocessing and data compression by microcontrollers before being put on the bus, the additional loading would be in the order of 1,000 to 2,000 bits per second to monitor 1,000 points.

C. SUMMARY

In the preceding sections of this chapter the data bus requirements of the various individual subsystems have been discussed. These data are summarized in table 1. The main

TABLE 1

SUBSYSTEM COMMUNICATION REQUIREMENTS

Subsystem	Bits/Second
Displays	16,000
Active Sensors	
Radar raw	2,000,000
processed	4,000
Sonar raw	150,000
processed	20
Passive Sensors	
Sonar raw	75,000
processed	50
Optical raw	4,000,000
processed	50
E.W	500
Weapons	
Control of a single weapon	3,200*
Communications	
Tactical	15,000
General	<10
Propulsion Control	50
Ship Monitoring	1,000

*Real-time requirement

emphasis has been on interprocessor communication or bus loading, since this is the critical requirement in a distributed computing system. The figures in table 1 are valid as long as, at a minimum, the computational requirements of any individual subsystem can be met in a single processor. Several processes active in one processor might decrease, but would in no way increase, the bus loading.

The question of how much computing power is required for the ship's system has had little attention. It is obviously an important problem but it is one which would require a much more detailed analysis of the various function than has been attempted here. It is not intended to discuss this in depth, but a few general comments can be made.

The processing requirements can be met by choosing the the appropriate number and size of computers. While a distributed computing system puts no constraint on the maximum size of computers to be used in it, one of its main advantages is to be able to use several smaller and, therefore, cheaper computers, rather than a large expensive one. The minimum size, as has been mentioned, is that which can handle the processing requirements for a single subsystem. Having exceeded this minimum size, the number of computers on the bus is transparent to the user and the software. Therefore, the system designer is basically free to vary the number and size of processors as he wishes.

D. DETAILED SYSTEM REQUIREMENTS

Having considered the individual requirements of the various subsystems and components of a distributed computing

TABLE 2

DATA DISPLAY REQUIREMENTS

function	CRT Displays number	raw	synthetic	Hard Copy
Command and control	10	x	x	
E.W.	1		x	
Sonar, active	2	x	x	
passive	1	x	x	
Weapons	3	x	x	
Communications	2		x	2
Propulsion control	2		x	1
Ship monitoring	2		x	

TABLE 3

SYSTEM REQUIREMENTS

Subsystem	Bits/Second
Displays 23 X 16000	368,000
Processed radar	4,000
Processed sonar	20
Passive sonar	50
Optical	50
E.W.	500
Weapons 6 X 3200	19,200*
Communications, tactical	15,000
general	10
Propulsion control	50
Ship monitoring	1,500
TOTAL	408,570

*Real time requirement

system, it is necessary to consider the system as a whole to determine the magnitude of the system (number of nodes, aggregate bus loading, etc.) and any system constraints. Table 2 summarizes a typical data display requirement for a small general purpose warship. Since raw sensor data rates are such that a separate dedicated bus will be required, those displays which must handle this data are identified. Table 3 is a list of all systems tied to the general purpose bus with their estimated average communications requirements. The individual requirements are summed to give a total average bus load of 400,000 bits per second.

Note, however, that this does not include communication requirements associated with operating system overhead. This requirement would have to be determined as the system design evolved. Also these are average figures, peak loading could be much higher and the effect of real-time control messages must be considered.

There is a requirement to look at the various subsystems from the point of view of autonomy or dedication. Do some subsystems because of their importance or unique requirements warrant being configured as autonomous systems? Are there groupings of subsystems that might have an autonomous function? The importance of main machinery control has been mentioned and is considered sufficiently great to justify autonomy, probably requiring 100% redundancy of processors and complete cross coupling. This is the only case that is sufficiently unique and important for such treatment.

The arguments for grouping of subsystems are somewhat less clear. There are several such groups possible, antisubmarine weapons and sonars, antisurface/anti-air weapons and radars, E.W. and jammers and decoys. A case could be presented for making these groups autonomous, however, there is also considerable interchange of information among these systems. The technical aspects of system integration and reliability have a bearing on this decision and it will be considered in more detail later.

Finally, system reliability must be considered. Controlled degradation of performance under failure is one of the greatest potential advantages of the ring structured distributed computing system. However, previous work by Farber and Harris has not had to contend with one of the most obvious failure modes in the military environment, that of battle damage. The ability of the system to continue to operate under these conditions is of utmost importance.

III. SYSTEM DESIGN

A. AREAS REQUIRING CONSIDERATION

The requirements of a warship's computing system impose several restraints not allowed for in the systems developed by Farber and Harris. These are:

1. bus speed,
2. real time processing,
3. multiple addressees, and
4. battle damage.

The effects of each of these on systems design will now be individually discussed.

1. Bus Speed

In a warship's computing system of the nature that has been discussed, it is critical that the bus be able to support the data rate required with little or no delays. In a system such as designed by Farber, if the bus should become overloaded for a period of time the result is mainly inconvenience to the user. On a warship such an overload could be fatal. The bus data rate is, therefore, an important factor in system design.

The bus data rate is determined by the node design and the propagation characteristics of the bus itself. It is these areas that will now be discussed. A node consists of a repeater and a ring interface. The bus consists of the repeaters and their interconnecting wiring.

While the possible media for connecting the nodes together are numerous, from a single wire with ground return to optical fibres, only three are worth considering due to problems of electromagnetic interference in the shipboard environment. These are coaxial cable, shielded twisted wire pair, and optical fibres. All three are capable of transmitting at five to ten megabits per second over the cable lengths anticipated on board ship. Optical fibres are capable of much higher frequencies.

An independent problem that arises when connecting several computers together electrically is establishing a common electrical ground. To avoid this common ground problem when using shielded electrical cables, optical isolators can be inserted into the bus at each node. The use of optical fibres, of course, dispenses with this problem altogether. Since cost, availability, and convenience of optical fibres are rapidly approaching those of coaxial cable, they would appear to be the best choice for the bus.

While the ring operating at the University of California at Irvine operates at 2.25 GHz, it uses, as mentioned, a hardwired interface. The ring interface proposed by Harris [1] using a microprocessor was designed to operate at 112 kilobits per second bus data rate. This was determined by the memory cycle time (1.1 microsecond) of the erasable PROM in the ring interface microprocessor, a requirement for the execution of four microprocessor instructions between bits, and a two for one encoding ratio of transmitted bits to data bits. The remainder of the circuitry, being transistor transistor logic (TTL) modules, is capable of sustaining five to ten megabits per second with no difficulty. Currently non-erasable PROM's are available with access times of the order of 50 nanoseconds. This would allow the bus data rate to be increased to 2.5 megabits per second.

The requirements in table 3, for the system being considered, result in an average bus data rate of 400,000 bits per second. The most significant load (more than 90% of the total) is that required for displays. Even allowing for an error in the estimate of the average data rate of a factor of three, a 2.5 MHz data bus still has sufficient capacity to handle peak loads of two times the average. If this is not capable of handling the communications requirements, PROMs utilizing emitter coupled logic are available with 20 ns cycle times raising the possible bus rate to over six MHz.

The other side of the interface must now be considered, the interface between the node and the processor. For purposes of discussion the characteristics of the AN/UYK-20 minicomputer will be used, as it is a militarized minicomputer in common use. This computer uses a 16 bit I/O word. Therefore, at 2.5 million bits per second bus rate with 16 bit buffering in the node the I/O controller would require access to memory at least once every 6.4 microseconds or about every eighth memory cycle. The computer can thus handle this date rate continuously with about a 12.5% loss of processing time.

If the bus rate were raised to five or six MHz the loss of processing time would increase to 25 or 30 per cent for continuous sending and receiving. This still may be acceptable if, in fact, a lower proportion of the time is spent in communicating. With 10 to 15 processors any one should be communicating with the bus only 15 to 25 per cent of the time on the average. Thus, direct communications with the ring interface from a five MHz bus with only one 16 bit serial to parallel conversion would still be adequate.

If bus comunication requires too much processor time, there are several possible courses of action. First the AN/UYK-20 is capable of 32 bit parallel communication. This would halve the time required to service bus interrupts but the buffer size in the ring interface would have to be increased. Second, a direct memory access capability is available in the AN/UYK-20 which would allow the ring interface to access memory by a second port without locking out the central processor. However, this would require memory interface logic in the ring interface. Finally a first in first out (FIFO) buffer in the ring interface would allow more flexibility in the rate of passing data between the ring interface and the host processor, i.e. the processor could be interrupted once and a block of several words could be accepted or transmitted on consecutive memory cycles. The advantage gained by this technique would be strongly dependent on the nature of the processing being carried out as well as the relationship between maximum message length on the bus and the size of the FIFO buffer.

So far no mention has been made of disseminating raw data. The requirement to display this data is present in approximately 16 displays (see table 1). It should be apparent from the discussion of the requirements that a dedicated bus is required for this information. Since a single display would only be required to display the raw output from one sensor at a time, one obvious configuration for this bus would be a "star" pattern. All sensors would feed into a central switching unit which would then distribute the data to the various displays as required. As this raw data system is independent of the distributed computing system and would be required no matter what type of computing system were used, there will be no further discussion of this requirement.

2. Real Time Processing

The transmission of messages for real-time control purposes poses another problem. If such a message were held overly long in a queue waiting for the ring interface to gain control havoc could result. Therefore, there must be a method whereby a real-time message can be forced onto the bus.

One possible method could be to have a priority interrupt token. A process requiring to transmit a real-time message could interrupt the ring, transmit an interrupt token to warn the rest of the ring and the sender of the interrupted message that it had pre-empted the ring, and then transmit the priority message in the normal manner. The simplest follow up would be for the interface originating the interrupted message to retransmit it at the next opportunity, the same as if it had detected a faulty transmission on an uninterrupted message. A possible problem arises from this system, however, in that control has jumped over the nodes between the interrupted sender and the interrupting node. Thus, instead of being guaranteed a chance to send a message at least once in a time period equal to the product ot the number of nodes and the longest message time, a node could be locked out. To preclude this the interrupting interface, rather than placing a control token on the ring at the end of its message, could place a special marker onto the ring which when recieved by the interrupted node would signify that it could reassert control and retransmit its message.

Another, but more complicated way would have the interrupting ring interface store the remainder of the interrupted message and retransmit it on completion of the priority message. This would have the least detrimental effect on ring communications but would require either a FIFO buffer in the ring interface or full duplex operation between the ring interface and its host processor. A variation on this would be to impose a fixed and fairly short length on priority messages. A serial-in serial-out shift register would then be used to delay the interrupted message the appropriate number of bits. This is perhaps the most pratical of the methods discussed. For example a 48 bit delay would allow three twelve-bit control words plus 12 bits for interrupt token and addressing. The end of the priority message could then be found by counting bits rather than needing another token.

3. Multiple Addressees

One of the basic tenets of the ring structured distributed computing system is that interprocessor communications are addressed to processes, not processors. In the ship's general purpose distributed computing system there are a number of cases where the same data is required by several processes, for example: ship's course, roll and pitch are required for processing of sensor data, for gun control calculations, for manoeuvring etc.; also, an update for synthetic video may be used by several displays. To minimize bus loading problems, it is desirable that one message be sufficient to transfer such data to all users of it. There are several possible ways of accomplishing this. One would be to set up a specific receiving process for each of these various classes of data. However, this would mean that a process which requires this information must ensure that the appropriate receiving process were resident in the same processor. This would violate the requirement for independance of processes and physical locations.

A second possibility would be to structure the process names to allow qualifiers. A general message could then be sent to all processes with the same qualifier. Problems arise here if the number of qualifiers is large, then complicated decoders and/or long process names would be needed.

Perhaps the simplest solution would be to allow a process to have several names. Some of these would be the same as names of other processes requiring the same information. Host software would only have to ensure that all resident users of the same name were referred to the same input buffer for the common data.

4. Fault Tolerance Under Battle Damage

A situation that Farber's distributed computing system cannot cope with is that of a failure in the ring itself. While the failure of a twisted wire pair is so improbable as to be inconsequential in a non-combat environment, the probability of the shipboard ring being severed by battle damage is very real and cannot be ignored. The solution requires three separate actions:

1. determination that the ring has been severed,

2. localization of the break,

3. repair or bypassing of the break.

The severing of the ring will be immediately apparent to the node immediately beyond the break. Using the same encoding scheme as Harris [1] ensures that the input cannot remain in the same state for more than two clock pulses. The only exception is for a token, at which time the state persists for three clock pulses. Detection of four or more identical bits in a row would indicate a break. If the detecting node then immediately started to transmit, no other node would detect the error and the first two steps of the solution would have been accomplished. What happens next will depend on the hardware configuration.

It is obvious that some sort of path redundancy must be built into the ring to allow for this situation. The simplest would be to have two or three or more complete rings. For reasons to be discussed later they should connect the nodes in different sequences. All nodes would listen to all incoming connections but only transmit on one output connection, all others being held in the same state. If a break were detected the detecting node could send on the broken ring a priority message which would cause each node to switch to the next back up ring. Simultaneous breaks would be handled by this system since each node immediatly downstream of a break would initiate such a message and it would be passed along to all nodes until the next break was reached. The time-out system would then reinitialize the ring. If the new ring were not established within a specified longer time limit an automatic switch to the next ring could be incorporated.

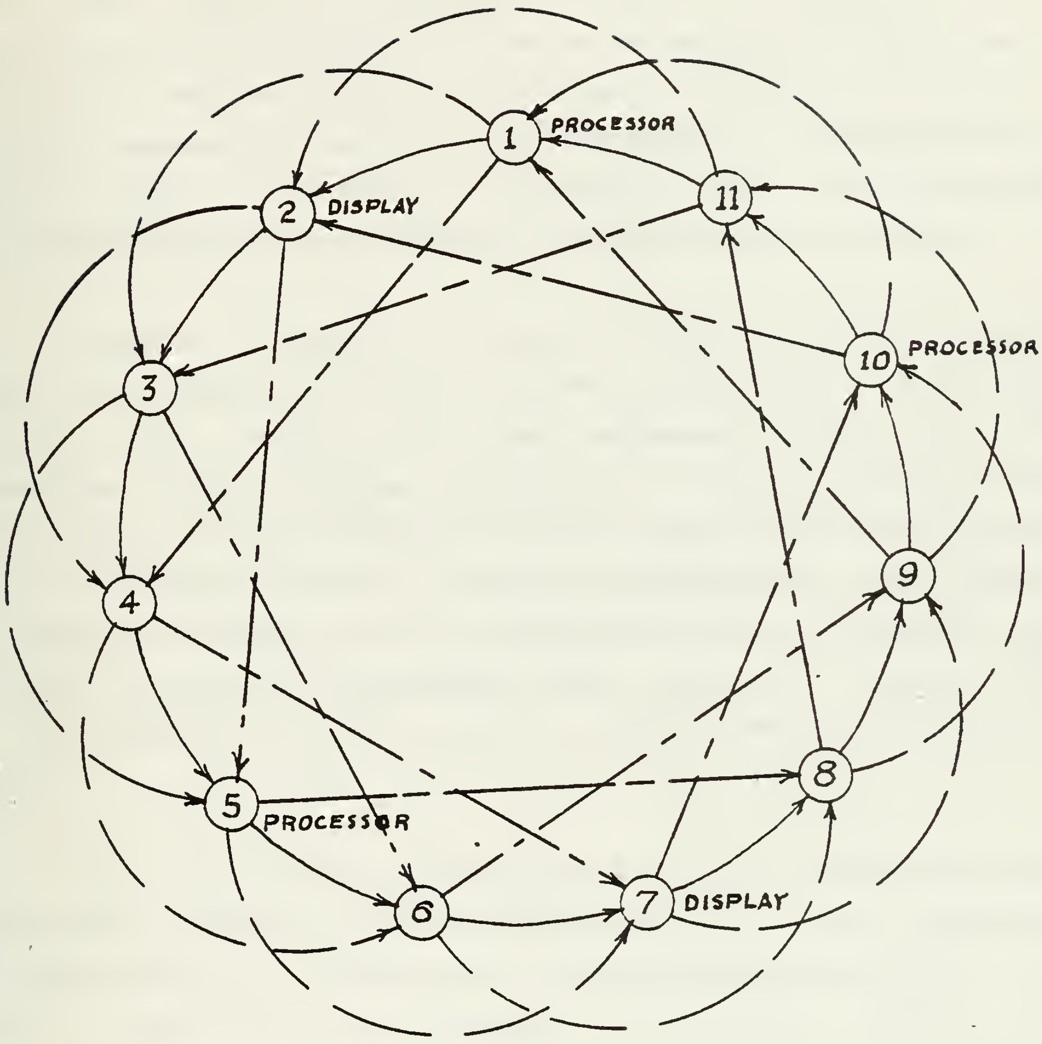

Figure 1 - A SIMPLE RING

The main weakness in this system is the situation where a node is disabled. This would interrupt all rings since they all pass through all nodes. Furthermore, if all rings connected the nodes in the same sequence, the loss of a node is catastrophic. If the nodes are connected in different sequences, ways can be devised to use segments of various rings to bypass a defective node. The following paragraphs propose one method of solving this problem.

Figure 1 shows a seven node ring with each node having three input and three output connections. The circle connects the nodes in one sequence (ring 1). The connections shown outside the circle (ring 2) skip every other node on the circle, and the connections shown inside (ring 3) skip two nodes. For this arrangement, as long as the number of nodes is not a multiple of two or three, rings 1, 2, and 3 will be independent and complete. Dummy nodes can be inserted into the ring to ensure that this condition is met.

Now consider what happens if ring 1 is in use and a break occurs between nodes three and four. All interfaces are listening to all inputs but transmitting on ring 1. All inputs to node 4 are now constant and it will realize that ring 1 has been broken. One of two things will occur at node 4 at this point. If node 4 has as a host a general purpose processor, it will alert the processor and send out on the ring a ring broken (RBR) token and a message saying to stand by for instructions. If it has a less intelligent host (a gun controller or a mass storage device perhaps), it will transmit an RBR token and its node number. In this case the first node having an appropriate host will alert its host and change the RBR message to "wait for instructions". Once an RBR token has been received a node would disregard a constant input until after the ring had been reestablished. This is necessary to avoid having messages arriving at a node on two different inputs at one time.

The situation is now that the first general purpose processor downstream of the break knows where the break is. Moreover, since the order of nodes on the working ring would be stored by all processors each time the ring is established or restructured, the processor would be in a position to control the restructuring of the ring.

Obviously, the first thing to try is a switch to an alternate ring. In the example, node 5 would take control and attempt to switch to ring 2. It would transmit this command on ring 2 and each node, as it received the command, would retransmit it on the new ring. When the message returned to node 5 it would know reconfiguration was complete and reinitialize the ring.

If ring 2 were also broken, the message sent out by node 5 would never return to it. After an appropriate time node 5 would transmit a command on ring 3 to switch to ring 3 and wait for this comand to return. If this does not occur then it is apparent that all 3 rings are broken.

The most probable cause of an interruption of all three rings is that one node, rather than three independent ring segments, has been destroyed. In the example the logical conclusion is that node 3 is at fault. A command could be sent out by node 5, on ring 1, and addressed to node 2, telling it alone to switch to ring 2. This would bypass the presumed defective node 3. Again the return of the message to node 5 would indicate that the ring had been reestablished. Reinitializing the ring in this case would be more complex, as a node and its processes have been lost.

Failure of this last message to return in a reasonable time could initiate further attempts to reconfigure the ring. Perhaps node 1 could switch to ring 3 bypassing both nodes 2 and 3. Eventually the processor controlling the reconfiguration will exhaust all programmed possibilities. It would then send out a message identifying the known servicable nodes, in the example only node 4 and itself. Any display downstream would receive this message and display it. Any processor downstream would add the numbers of the intervening nodes. Thus the last display before the break would have as complete a specification as possible of the contiguous segment of the ring. In the example ring the display at node 7 would display that nodes 4 and 5 were serviceable and the display at node 2 would display that nodes through 11 and 1 were serviceable. Human repair of the ring would now be required.

It would be possible, during the attempted reconfiguration and after reconfiguration had failed, for the controlling processor to put control tokens on the ring at regular time intervals allowing functioning processes in the contiguous portion of the ring to time-share the bus in a degraded mode in which a node would not expect to receive its own message back. Priority real-time messages could be sent as usual and, of course, reconfiguration control messages would be sent as priorities. Thus one-way communication with processes downstream and before the break would be possible. Whether this mode would have any real value is debatable and would depend on the physical location of processes at the time of the break.

The actual number and routing of internodal connections would depend on the degree of survivability desired, the physical location of the various nodes etc. It is obviously very dependent on the particular installation and could provide a major area for operational analysis to determine optimum values of these parameters. However, only the maximum possible number of interconnections per node would affect the software and once this was decided the actual routing would not affect the system. Thus a standard software package could handle ring reconfiguration.

B. RESULTING DESIGN CONCEPTS

Appendix I contains a block diagram of a ring interface and repeater adapted from those proposed by Harris in Ref. 1. The main changes are:

1. the addition to the repeater of n continuously monitored inputs and n selectable outputs,

2. the addition of a priority interrupt (pri) and a ring break detected (RBR) token,

3. the inclusion of an m bit delay circuit for interrupted message handling, and

4. the expansion of the input and output buffers to 16 bits.

These changes would allow a ring structured distributed computing system to operate in the environment of a small warship as described in the previous sections. However, there are several other areas that should be considered. One is the requirement for mass storage.

On starting up the ring or upon restructuring the ring after battle damage, the system must be reestablished and available resources distributed as effectively as possible. This requirement plus the need for any processor to have available a large number of processes which it might be required to implement necessitates the existence of copies of these processes available for loading by the processors. These programmes might be stored in mass storage devices attached directly to the ring and thus be accessible by all the processors and provide shared bulk storage as well. However, for survivability there would have to be several copies on several separate devices distributed around the ring. There is, then, some justification for each processor to have its own dedicated bulk storage. This would ensure access to bulk storage no matter what has happened to the ring and would reduce bus traffic during reconfiguration. When it is considered that the most probable time that reconfiguration would be necessary is also the time of greatest system activity, i.e. during action, the latter consideration could be very important.

Another area for consideration is the redundancy and autonomy built into the system. Ideally it would be desirable for all processors to have access to all inputs. For example, all processors may be capable of performing the radar auto-tracking function. However, this would require a dedicated bus to each of the radars. It would be impractical to provide this to all processors. Among other things, there are not enough input ports to a processor to handle all the inputs required. It may be impractical in some cases, possibly for reasons of physical location, to provide a particular input to more than one processor. Thus, there are going to be processors dedicated to a particular task. The greater the number of inputs that can be made available to more than one processor the greater the flexibility there can be for reconfiguration and hence the greater the survivability of the system.

The existence of such dedicated devices in conjunction with the modes of degraded operation have some useful consequences. If, for example, a display with access to a sensor, a processor, and the appropriate weapon were included in that order in the ring, as long as a break in the ring did not occur between them, the weapon could still be controlled and fired. These considerations should be taken into account when designing bus routings.

IV. CONCLUSIONS AND RECOMMENDATIONS

The concept of using a ring structured distributed computing system for a general shipboard computing system appears to be feasible. While maintaining the advantage of flexibility in size and configuration, the system can be extended to provide the required bus speed, handle real-time processes, and cope with the additional hazards of the warship environment. The additional software overhead to accomplish this does not appear to be too great, however, a detailed analysis would be required to verify this assumption. There is some increase in the complexity of the node, but it is still well within the capability of the microprocessor used by Harris.

While the system appears feasible there are still many areas that require greater study. Neither the software nor the communications overhead of the system have been determined. The bus data rates used have been averages. Thus a detailed look at the distribution in frequency and length of interprocessor messages is required and possibly a simulation needs to be developed to determine the capability of the system to handle peak loads.

Finally, the costs and benefits of the ring structured distributed computing system must be compared with other forms of the distributed computing system, particularly the bidirectional multiply-connected data bus, to determine the overall relative value.

APPENDIX A

RING INTERFACE BLOCK DIAGRAMS

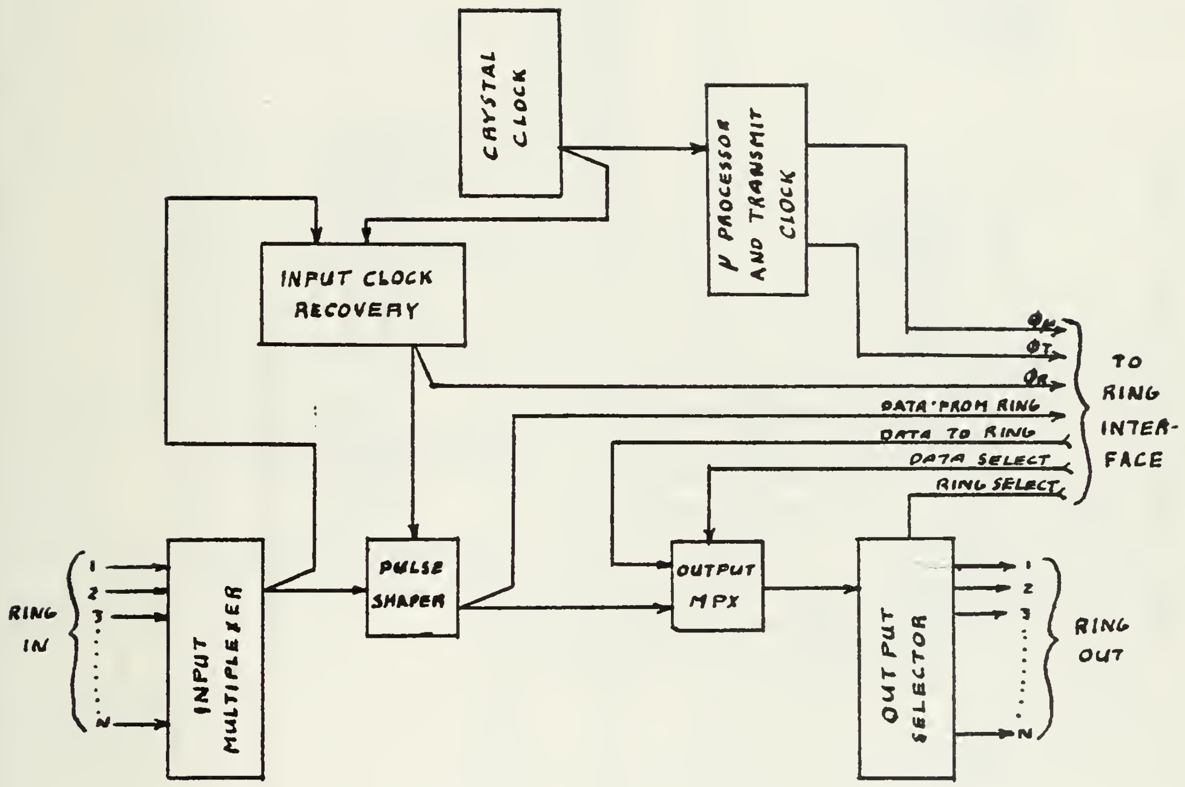

Figure 2 - REPEATER BLCCK CIAGRAM

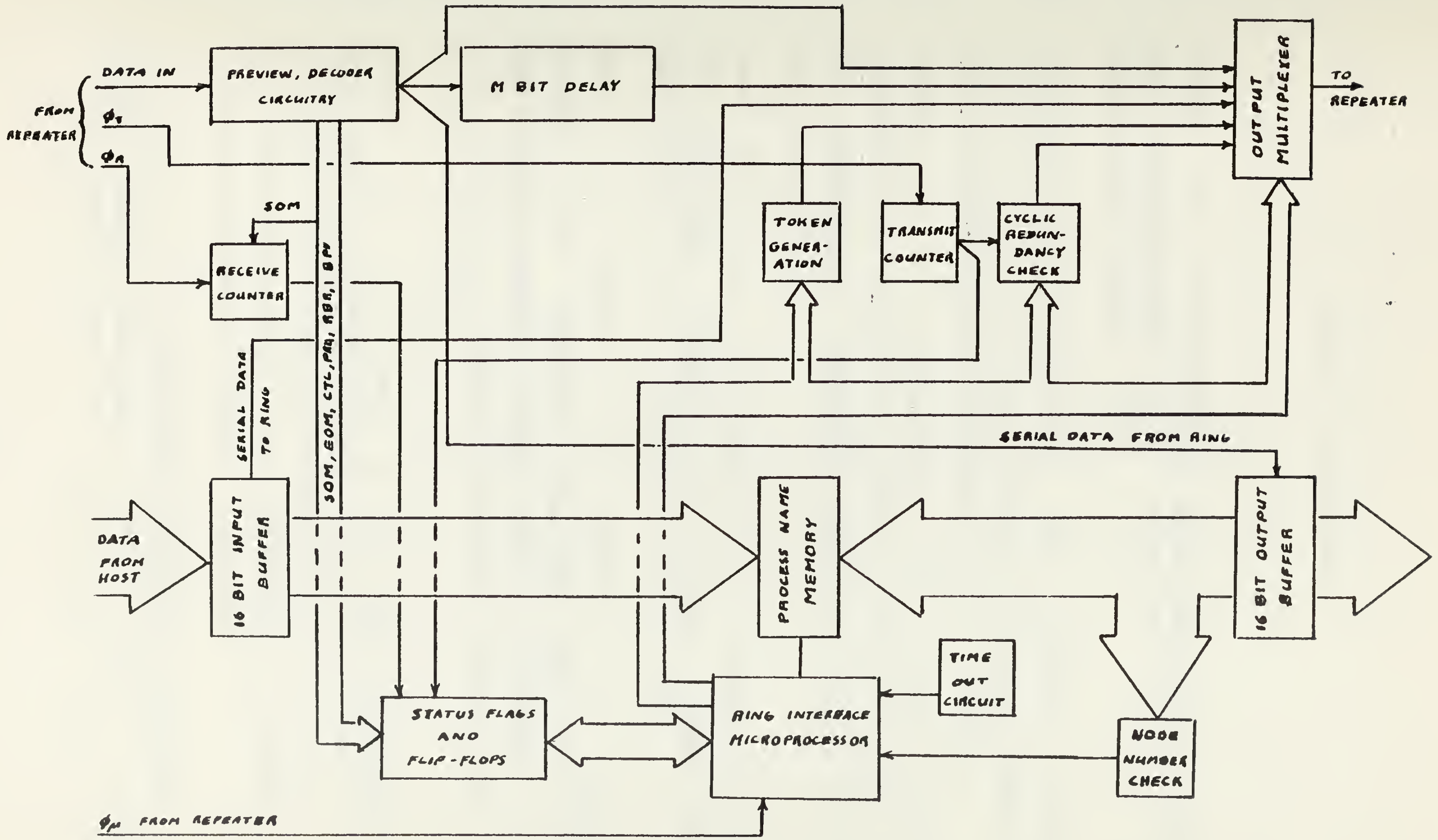

Figure 3 - RING INTERFACE BLOCK DIAGRAM

LIST OF REFERENCES

1. Harris, M. J., _A Prototype Ring Interface for the NPS Data Communication Ring_, Master's Thesis, Naval Postgraduate School, 1974.

2. Department of Information and Computer Science, University of California, Irvine, Technical Report 26, _Ring Communications Protocols_, by D. C. Loomis.

3. Department of Information and Computer Science, University of California, Irvine, Technical Report 66, _The Distributed Computing Operating System_, by L. A. Rowe, June 1975.

4. Naval Electronics Laboratory Center, Technical Document 318, _Small Ship Command and Control System, System Description_, by E. J. Pasahow, 3 April 1974.

5. Rowe, L. A., Hopwood, M. D., Farber, D. J., _Software Methods for Achieving Fail-soft Behavior in the Distributed Computing System_, paper presented at the IEEE Symposium on Computer Software Reliability, April 30, May 1-2, 1973.

6. Farber, D. J., Larsen, K. C., _The Structure of a Distributed Computing System_, paper presented at the symposium on Computer-communication Network and Teletraffic sponsored by the Polytechnic Institute of Brooklyn, Microwave Research Institute.

7. Carruthers, J., _An Integrated Shipboard Processing and Display System (Proposed)_, SHINPADS Paper 6, contained in 3120-280/c3-3(DMCS 7) dated March 1976, Ottawa,

Canada.

8. NATC Industrial Advisory Group SG6/16, Data Transfer Analysis, March 1974.

9. SHINPADS Minutes 4, Dec. 1975, contained in National Defence Headquarters letter 3120-280/C3-3(DMCS 7) dated 10 Dec. 1975, Ottawa, Canada.

INITIAL DISTRIBUTION LIST

1. Defense Documentation Center 2
   Cameron Station
   Alexandria, Virginia 22314

2. Library, Code 0212 2
   Naval Postgraduate School
   Monterey, California 93940

3. Department Chairman, Code 62 2
   Department of Electrical Engineering
   Naval Postgraduate School
   Monterey, California 93940

4. LTJG G. M. Raetz, Code 52Rr 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, California 93940

5. Asst. Professor V. M. Powers, Code 52Pw 1
   Computer Science Department
   Naval Postgraduate School
   Monterey, California 93940

6. Director, Maritime Combat Systems 2
   National Defence Headquarters
   Ottawa, Ontario
   CANADA K1A 0K2

7. LCDR J. Q. Jackson, Canadian Forces 2
   5731 Atkins Street
   Ottawa, Ontario; CANADA