Intercomputer Communication Architecture for a Mixed Redundancy Distributed System

Jaynarayan H. Lala* and Stuart J. Adams†
The Charles Stark Draper Laboratory, Inc., Cambridge, Massachusetts

Advanced aerospace vehicles now on the drawing boards are likely to use distributed information processing systems for the numerous planning, control, and monitoring functions. The Advanced Information Processing System (AIPS) is one such architecture that has been designed to serve as the core avionics system for a broad range of aerospace vehicles. One of the key issues in the design of distributed architectures is the communication between processing sites. The intercomputer communication design becomes a crucial and challenging aspect of the system architecture in the light of the mixed redundancy requirement. Another fundamental requirement is that a single failure must be tolerated without affecting the operation of any triply redundant processing sites or communication between triplex processing sites. The AIPS intercomputer communication mechanism has been designed to meet these requirements. This paper describes the AIPS intercomputer communication architecture, the intercomputer arbitration protocol, and some simulation results for arbitration in the presence of faults.

I. Introduction

SIGNIFICANT progress has been made over the last few years in the use of digital computers to enhance the performance of aerospace vehicles. A common characteristic of these electronic systems has been their centralized architecture. Although these centralized systems have served their applications well, there are a number of advanced aerospace vehicles that can benefit significantly from distributed computer systems. These include the Space Station, the Aeroassisted Orbital Transfer Vehicle, the Entry Research Vehicle, the National Aerospace Plane, and advanced fighter aircraft for the U.S. Air Force and Navy.

Fault-tolerant distributed systems can have characteristics superior to those of centralized systems and are better suited to highly integrated electronic information systems of future vehicles. These attributes include function integration, parallel computation, graceful performance growth, selective technology upgrade, appropriate levels of function reliability, graceful degradation of system capabilities in the presence of faults or damage, and efficient hardware resource utilization. A system with these attributes, called the Advanced Information Processing System (AIPS),[1,2] has been developed by the Charles Stark Draper Laboratory under the sponsorship of NASA.

For a distributed information processing system to be cost effective, it is necessary to match the reliability level, and hence the redundancy level, of a processing site to the reliability requirements of the functions assigned to that site. Thus, critical functions with real-time response requirements, such as aircraft flight control or spacecraft attitude control, must be hosted in at least a triple redundant computer to provide dynamic error masking. Less critical functions, such as flight management or trajectory optimization, may be assigned to duplex processing sites, while noncritical functions, such as an engine health trend monitor, may be executed on a simplex processor. Although such an architecture makes efficient use of hardware resources, it does imply that the system support mixed or graded redundancy. Federated systems on contemporary commercial aircraft support mixed redundancy. However, they do not possess many other attractive features of distributed systems outlined earlier because of the rigidity of the architecture that results from a fixed assignment of functions to processing sites and very little sharing of information between processing sites. To provide these additional features, it is necessary that functions located at different processing sites be able to communicate with each other at high speed and with high integrity.

The design of such an intercomputer "bus" poses a number of challenges that are made all the more difficult when one considers the graded redundancy requirement. This paper discusses how the architecture of the AIPS intercomputer bus meets this challenge and specifically the contention resolution problem in a graded redundancy system such as AIPS. The overall AIPS architecture has been described previously in the literature,[1,2] and a detailed architectural discussion is outside the scope of this paper. However, to make this paper stand alone, Secs. II and III briefly describe the system architecture and the intercomputer communication architecture, respectively. Section IV then goes into the details of resolving contention between processing sites for access to the intercomputer network, a failure modes and effects analysis of the contention resolution mechanism, and some results from a simulation of the arbitration mechanism in the presence of faults.

II. Advanced Information Processing System (AIPS) Architecture Overview

The Advanced Information Processing System is designed to provide a fault- and damage-tolerant data processing architecture that can serve as the core avionics system for a broad range of aerospace vehicles for which NASA has direct or supporting research and development responsibilities. AIPS is a multicomputer architecture composed of hardware "building blocks." These are fault-tolerant, general-purpose processors,[3] fault- and damage-tolerant intercomputer and input/output networks, a fault-tolerant mass memory, and a fault-tolerant power distribution system. The system software is another AIPS building block. It provides the traditional services necessary in a real-time computer such as task scheduling and dispatching, communication with sensors and actuators, and so forth. The software also supplies those services necessary in a distributed system such as interfunction communication across processing sites, management of local and distributed redundancy, management of networks, and migration of functions between processing sites.

AIPS consists of a number of computers located at processing sites that may be physically dispersed throughout the

Presented as Paper 87-2607 at the AIAA Guidance, Navigation, and Control Conference, Monterey, CA, Aug. 17-19, 1987; received Sept. 21, 1987; revision received April 29, 1988. Copyright © 1987 American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Division Leader, Fault Tolerant Systems Division. Associate Fellow AIAA.

†Staff Engineer. Member AIAA.

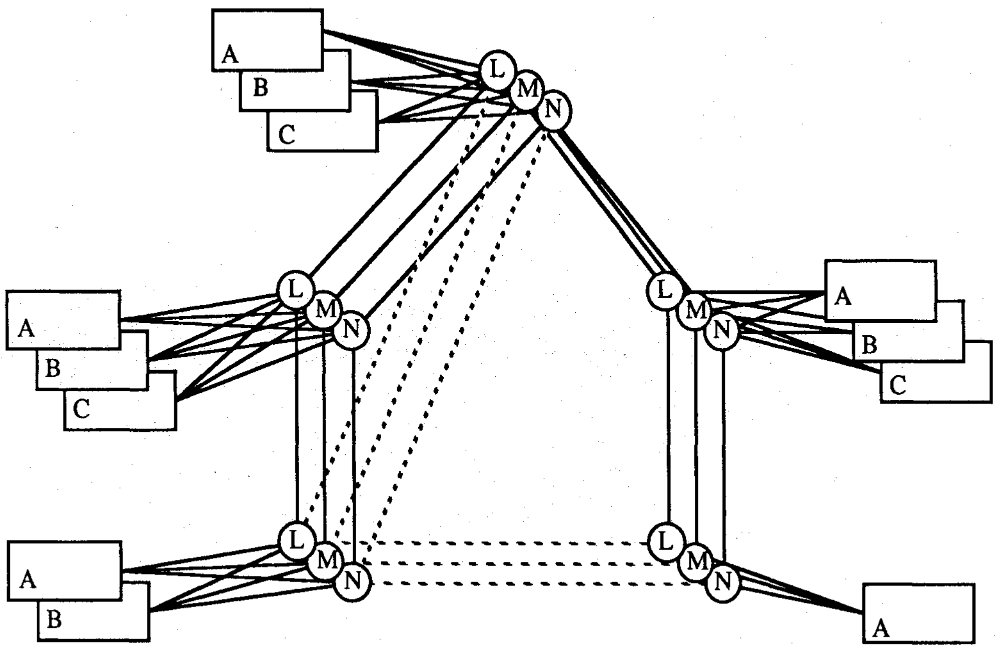

Fig. 1 AIPS: a typical mixed redundancy configuration.

vehicle. These processing sites are linked together by a reliable and damage-tolerant data communication bus called the intercomputer (IC) bus. A computer at a given processing site may have access to varying numbers and types of input/output (I/O) buses that are separate from the IC bus. The I/O buses may be global, regional, or local in nature. Input/output devices on the global I/O bus are available to all, or at least a majority, of the AIPS computers. Regional buses connect I/O devices in a given region to the processing sites located in their vicinity. Local buses connect a computer to the I/O devices dedicated to that computer. Additionally, I/O devices may be connected directly to the internal bus of a processor and accessed as though the I/O devices reside in the computer memory (memory mapped I/O). Both the I/O buses and the IC bus are time division multiple-access contention buses. There may be a system mass memory that is directly accessible from all computers in the system on a dedicated multiplex mass memory (MM) bus.

Computers at various AIPS processing sites are general-purpose computers (GPC) of varying capabilities in terms of processing throughput, memory, reliability, fault tolerance, and damage tolerance. Throughput may range from that of a single microprocessor to that of a large multiprocessor or a parallel processor. Memory size will be determined by application requirements. An extremely wide range of reliability, as measured by the probability of failure due to random faults, is provided by the AIPS building blocks. It ranges from approximately 10^{-4}/h for a simplex processor to 10^{-10}/h for a multiprocessor that uses parallel hybrid redundancy.

For those functions requiring fault masking, a triplex level of redundancy is provided. For low criticality functions or noncritical functions, the GPC's may be duplex or simplex. Parallel hybrid redundancy in a multiprocessor or a parallel processor is used for extremely high levels of fault tolerance and/or for longevity (long mission durations). Processing channels of a redundant GPC at a processing site are tied together by custom hardware to provide byzantine resilient data exchange, comparison, voting, error masking, and channel synchronization. Redundant processor channels execute identical code in microframe synchronization. GPC's also can be made damage-tolerant by physically dispersing redundant GPC elements and providing secure and damage-tolerant communications between these elements. Within AIPS, computers of varying levels of fault tolerance can coexist such that less reliable computers do not pose a danger to higher-reliability computers.

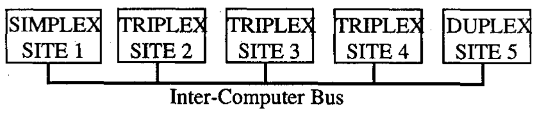

Fig. 2 The applications programmers view of the AIPS configuration of Fig. 1.

Each processing site has the ability to operate autonomously, particularly for the performance of critical functions. However, the AIPS concept allows the coordinated use of the entire information processing system to provide attributes superior to the more federated systems that are typical for current aircraft. All the AIPS general-purpose computers are designed to interface with the intercomputer communication medium such that high-speed communication between functions located on different processing sites can take place reliably and efficiently. The system is managed by a system manager resident in one of the GPC's. The system manager allocates functions to individual processing sites, performs system-level redundancy management and reconfiguration, and maintains knowledge of the system state for distribution to the component elements. The system management functions may be reassigned to an alternate processing site in-flight. Redundancy management, task scheduling, and other local services at individual processing sites are handled by local operating systems. System software, distributed across all processing sites, provides such services as intertask communications to applications programs.

Figure 1 shows a typical mixed redundancy configuration of AIPS. It consists of three triply redundant general-purpose computers, one duplex GPC, and one simplex GPC. The five processing sites shown are interconnected by a three-layer circuit-switched network. The three layers of the network are labeled L, M, and N. The active links in the network are shown by solid lines, and the spare links are indicated by dotted lines. Figure 2 shows the virtual AIPS architecture seen by the applications programmer. Note that the hardware redundancy is totally transparent to the user.

III. Intercomputer Network Architecture Overview

For communication between GPC's and between a GPC and I/O devices, a damage and fault-tolerant network is employed. The network consists of a number of full duplex links interconnected by circuit-switched nodes. In steady state, the circuit-switched nodes route information along a fixed

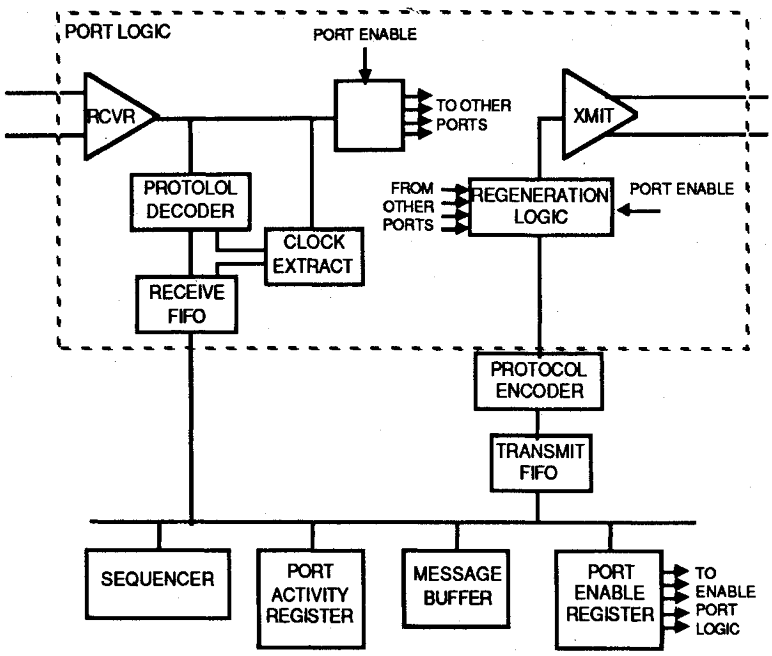

Fig. 3 Node port.

communication path or "virtual bus" within the network without the delays associated with packet-switched networks. Once the virtual bus is set up within the network, the protocols and operation of the network are similar to typical multiplex buses. Every transmission by any subscriber on a node is heard by all the subscribers on all the nodes just as if they were all linked together by a linear bus.

Although the network performs exactly as a bus, it is far more reliable and damage-tolerant than a linear bus. A single fault or limited damage can disable only a small fraction of the virtual bus, typically a node or a link connecting two nodes. Such an event does not disable the network, as would be the case for a linear bus. The network is able to tolerate such faults due to the richness of interconnections between nodes. By reconfiguring the network around the faulty element, a new virtual bus is constructed. Except for such reconfigurations, the structure of the virtual bus remains static.

The nodes are sufficiently intelligent to recognize reconfiguration commands from the network manager, which is resident in one of the GPC's. The network manager performs the necessary diagnostics to identify the failed element and can change the bus topology by sending appropriate reconfiguration commands to the affected nodes.

The network manager is part of the system software. There is one network manager for each of the three intercomputer networks. The three layers of the network are used to provide dynamic fault masking capability rather than higher bandwidth as compared to a single network. The three network managers operate independently of each other. They all may be resident in the same or different GPC's. The only requirements for a GPC to host a given network manager are that it have an operating physical connection to the network layer it is managing and that it be at least triply redundant to provide adequate reliability for the network management function. Furthermore, the network manager function can be dynamically migrated to an alternate GPC under system manager control if the current GPC ceases to meet these requirements.

The operation of the node is as follows. The internal configuration of the node is shown in Fig. 3. The circuits inside the dotted lines are repeated for each port. The common control circuits in a node monitor messages coming in on all ports whether that port is enabled or not. This procedure is necessary for the initial growth of the network. It is also necessary to monitor all ports so that the network manager will be able to respond to certain kinds of failures where the established paths have been disrupted by a malicious failure. The control decodes the message to determine if it is a valid message and if it is intended for that node. If so, the node responds to the message. Messages sent to nodes include requests for status and reconfiguration commands. The network manager requests status as an input to its network monitoring task. The reconfiguration messages establish or change the port enable status.

When a port is enabled, all messages received on that port are rebroadcast from other enabled ports after the waveform is regenerated. The total delay through the node is approximately one half of a clock period. There are several advantages to this arrangement. First, the bus can "Y," which cannot be done on most other buses. Second, the regeneration of the wave form makes the system less sensitive to the number of terminals and the length of the bus. Finally, all communication links are point-to-point, which is highly compatible with fiber optic technology.

Damage caused by weapons or electrical shorts, overheating, or localized fire would affect only subscribers in the damaged portion of the vehicle. The rest of the network, and the subscribers on it, can continue to operate normally. If the sensors and effectors are themselves physically dispersed for damage tolerance, and the damage event does not affect the inherent capability of the vehicle to continue to fly, then the digital system would continue to function in a normal manner or in some degraded mode as determined by sensor/effector availability.

Fault isolation is much easier in the network than in multiplex buses. For example, a remote terminal transmitting out of turn, a rather common failure mode that will totally disable a linear bus, can be easily isolated in the network through a systematic search where one terminal is disabled at a time. This, in fact, is a standard algorithm for isolating faults in the network. Furthermore, for networks of moderate size, up to 50 nodes, a total reconfiguration can be accomplished in a few milliseconds.

The network can be expanded very easily by adding more nodes linked to spare ports in existing nodes. In fact, nodes and subscribers to the new nodes (I/O devices or GPC's) can even be added without shutting down the existing network. In bus systems, power to buses must be turned off before new subscribers or remote terminals can be added. Finally, there are no topological constraints, as are encountered with linear or ring buses. In fact, these are simply subsets of the fault-tolerant network architecture.

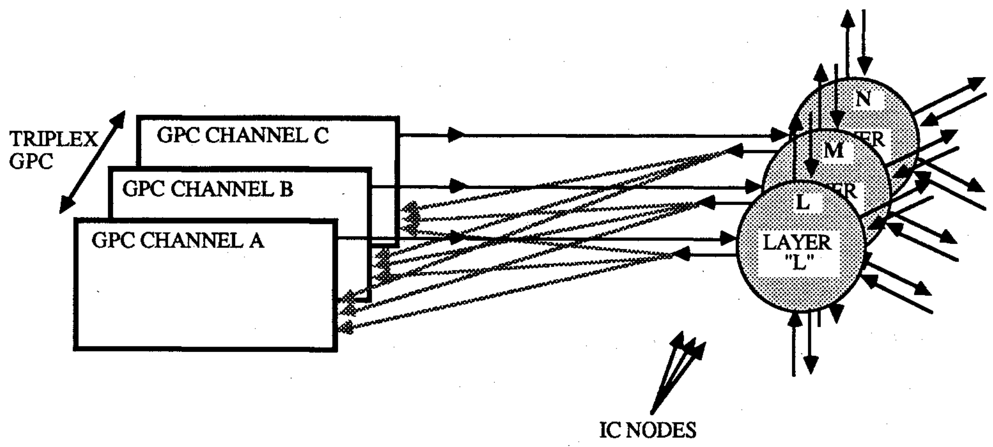

Fig. 4 GPC-IC network connections.

To maintain the fault-tolerance requirements, each GPC channel receives data from all three intercomputer network layers but can physically transmit on only one layer, as shown in Fig. 4. A GPC does not hear its own transmissions on the network but does hear the transmissions of all other GPC's. Once a path has been grown through the network, it appears as a serial bus using the HDLC data transmission format. The IC nodes perform the "OR" of incoming signals such that if any site transmits a "1," all sites hear a "1." Thus, the IC network forms a wired-OR HDLC serial bus interconnecting all sites.

All three layers of the IC network are used together when transmitting and receiving data, as explained previously. Since all channels of a triplex site are executing the same code synchronously, all three channels (each channel transmitting on a different layer) transmit identical messages. Thus, within some skew, the redundant layers of the network contain the same message. This allows the receiving site to vote the three layers, masking any failure. Although always receiving on all three layers, duplex sites can transmit on only two of the three layers of the network and simplex sites on only one of the three layers.

IV. Intercomputer Network Contention

This section describes the bus contention protocol design used to arbitrate access to the intercomputer network by the GPC's. The issue of reliable bus contention is critical in a mixed redundancy distributed system such as AIPS.[4]

The intercomputer circuit-switched nodal network is triplicated in the AIPS proof-of-concept system. As stated earlier, the three layers of the network are used to provide redundancy rather than higher bandwidth as compared to a single network layer. In normal operation, the three layers carry identical information. By making redundancy management and network contention transparent to application software, a very simple and easy to use virtual architecture of the triplex network, that of a highly reliable simplex bus, is presented to the applications programmer. In fact, most aspects of the network contention are hidden from the system software as well.

For access arbitration purposes, the triplex network is treated as a single entity. GPC's, regardless of their redundancy level, compete for all three layers of the network. At the end of the contention sequence, one and only one GPC may have access to all three layers of the network. Thus, if a duplex GPC wins contention, it is given exclusive use of all three network layers, even though it can broadcast on only two of the three layers. No effort is made to maximize the network bandwidth by providing simultaneous access to a duplex GPC on two layers and a simplex GPC on the third layer, for example.

A. Intercomputer Contention Arbitration Requirements

The bus arbitration scheme must meet certain requirements for the type of applications for which the AIPS architecture is intended. Communication between critical functions, which are resident in triplex GPC's, must not be interrupted or corrupted by lower criticality functions resident in duplex and simplex GPC's. Triplex GPC's should be given access priority over all others. Similarly, duplex GPC's should have priority over simplex GPC's.

In a distributed system such as AIPS, the contention resolution must be fair and equitable to all sites of like redundancy level. Over a period of time, for example, all triplex GPC's should have an equal chance of getting network access. Similarly, all duplexes should be served equally well by the network, as should all simplexes. However, the arbitration scheme also should be flexible enough so that a low criticality function that may be assigned to run on a triplex GPC should not hog the network. A triplex GPC should be able to contend as a duplex or a simplex, if the function requesting the communication is of appropriate low priority.

No single-point failure should result in a communication disruption between two triplex computers. The arbitration logic must be able to resolve bus contention in a reliably robust manner, even in the presence of an arbitrary fault. In other words, a malicious failure in a simplex GPC or in one channel of a redundant GPC should not be able to disrupt traffic on more than one layer. Furthermore, in keeping with the spirit of the distributed nature of the AIPS architecture, the arbitration authority must not be centralized. It should be distributed throughout the system, and all processing sites wishing to access the network at any given time should arrive at a consensus about the sole winner cooperatively but independently, in a fashion that preserves network integrity in the face of failures or damage.

In addition, redundant channels within a GPC must come to a consensus as to whether or not they are contending for the bus, and at the end of the contention sequence whether or not they have won access to the bus. The bus contention protocol does not occur in the GPC processors themselves but in the dedicated intercomputer interface sequencer (ICIS) electronics, which are appended to the GPC processors.

B. Bus Contention in the Presence of Faults

The IC arbitration protocol is an evolution of the Laning poll, a protocol previously used to resolve bus contention internally in the fault-tolerant multiprocessor (FTMP).[5]

1. *The Laning Poll*

The Laning poll is a bit serial algorithm for prioritized contention of serial buses. The Laning poll assumes that multiple sites can transmit on the same bus or serial line simultaneously, each site then receiving the "OR" of all bus transmissions. That is, if any site transmits a "1," all sites will hear a "1." Each site contending for the bus has its own unique binary priority vector $P(v1,v2...,vn)$ ordered from most significant to least significant bit. A higher number signifies a higher priority. The Laning poll algorithm guarantees that the site with the highest priority will win the bus. The poll consists of sequentially transmitting each priority bit (from most significant to least significant) on the bus. Each site behaves according to the following algorithm during the polling period:

```
For all $i$ while still contending do:
    Transmit $P_i$ on the serial bus

    If $P_i = 1$ and Received value $= 1$ then continue
    If $P_i = 1$ and Received value $= 0$ then win bus
    If $P_i = 0$ and Received value $= 1$ then quit
    If $P_i = 0$ and Received value $= 0$ then continue
```

This algorithm implies that all sites on the bus are synchronized in some manner so that all sites are transmitting their ith priority bit simultaneously. A start bit precedes the polling sequence to ensure synchronization of the polling sequence across all sites.

2. *The AIPS Contention Protocol*

The AIPS contention protocol uses a modified form of the Laning poll. It consists of two parts, the redundancy contention sequence (RCS) and the priority contention sequence (PCS). The RCS consists of 3 bits: S, T, and D (denoting start, triplex, and duplex, respectively) and the PCS consists of three GPC priority bits followed by six GPC ID bits. The objective of the redundancy contention sequence is to resolve contention between the different levels of redundant elements contending for the bus (i.e. triplex, duplex, simplex). At the end of the RCS, all nonfailed GPC's still contending should be of the same redundancy level. The priority contention sequence resolves contention among nonfailed GPC's of the same redundancy level according to the priority and the ID bits. The redundancy contention sequence is discussed next.

3. *The Start Bit*

The bus contention begins with the "S" (start) bit. A GPC may initiate a contention sequence if, 1) the network has been idle for $> 256\ \mu s$ or 2) the network has been busy for > 100 ms since the last contention. The first condition is simply the nominal condition under which a GPC initiates a poll. However, certain other conditions, such as a babbling failure of a GPC, are also valid conditions for starting a poll, as exemplified by the second condition. To initiate a contention poll, a GPC transmits a "1" on all layers to which it is connected. Other GPC's may join in the poll sequence during the start bit by transmitting their own S bits. All channels OR the S bits on all three network layers. The S bit serves to synchronize all contending sites at the beginning of the poll sequence.

4. *The Triplex Bit*

All triplex GPC's contending for the network transmit "1"s on all three network layers during the T (triplex) bit. All GPC's contending for the network vote the T bit on the three layers. If the voted result is a "1," indicating the presence of a triplex in the contention sequence, all GPC's not configured as a triplex drop out of the contention sequence. All contending GPC's configured as triplexes skip the next RCS bit and proceed directly to the priority contention sequence. If there is only one triplex in the contention, it should obtain a voted result of "0" and declare itself the winner. It does not go through the rest of the contention sequence. A triplex not contending as a triplex for low functional priority reasons does not transmit during this bit. If duplexes or simplexes obtain a voted result of "0" during the T bit, they conclude that no triplexes are contending and they go on to the D bit.

5. *The Duplex Bit*

The D (duplex) bit is used to resolve contention between duplex and simplex GPC's. During the D bit poll, a GPC configured as a duplex transmits a "1" on the two layers to which it is connected. All duplex channels contending for the network vote the three layers. All simplex channels contending OR the three network layers. If in either case the result is a "1," indicating the presence of one or more duplexes, all GPC's not configured as duplexes drop out of contention. All GPC's configured as duplexes proceed to the priority contention sequence. If there is only one duplex in the contention, it should obtain a voted result of "0" and declare itself the winner. It does not go through the rest of the contention sequence. If simplexes obtain a result of "0," they proceed to the priority contention sequence.

6. *The Priority Sequence Bits*

The priority sequence consists of three priority bits and six GPC ID bits. The voted or ORed result of each of these poll bits, as described next, is treated in the traditional Laning poll manner.

Triplex contention: Each channel of a triplex GPC votes the three network layers.

Duplex contention: Each channel of duplex GPC votes the three network layers.

Simplex contention: Each simplex GPC channel OR's the three network layers.

7. *Comparison with Other Contention Schemes*

The AIPS intercomputer contention resolution scheme, the modified Laning poll, differs in several important respects from other contention resolution schemes commonly in use now. The AIPS contention scheme is deterministic in nature. By comparison, the Carrier Sense Multiple Access/Collision Detect (CSMA/CD) scheme used in the popular Ethernet is statistical in its contention resolution performance. Given a set of contending GPC's, the AIPS scheme will always result in the same winner. The winner of the bus will be determined in a predetermined maximum length of time because of the deterministic nature of the modified Laning poll algorithm. This can not be said of the CSMA/CD. In fact, as the bus load increases in the CSMA/CD scheme, a limiting point is reached (around 20% of the bus bandwidth) where the bandwidth used to transmit data by the winners starts to *decline*. This is because more and more bus time is spent in collisions. In the modified Laning poll, the useful bus bandwidth continues to increase with demand until the bus is saturated. With a further increase in demand, the useful bandwidth stays the same and does not decline.

Another popular contention resolution scheme is the token-passing strategy where a token is circulated on the bus (typically a ring topology). A potential bus user waits for the token to reach its bus interface unit (BIU), at which point it does not retransmit the token on the bus, thereby assuring access to the bus. Although simple in concept and seemingly a fair way of arbitrating bus access equally among many users, the token-passing scheme suffers from single-point failures. Many variations have been proposed to contend with lost tokens and extra tokens on the bus. However, none can cope with all the failures modes of a GPC or a BIU in corrupting the tokens in a myriad of ways. By comparison, the AIPS scheme does not suffer from any single-point failures. In fact, it has specifically been designed to be tolerant of *all* arbitrarily malicious single-point failures, including Byzantine failures.

C. Failure Modes and Effects Analysis (FMEA) of the Contention Protocol

1. *Start Bit FMEA*

The S bit acts as a start bit, indicating the initiation of a poll. All GPC's continue after the start bit, so this is not a problem. A GPC might drop out during the start bit, but this should not happen in the fault-free case. A working ICIS will not think he won during the S bit. A bad network layer can make the bus look busy all the time but eventually, after 100 ms, others can try to contend anyway. This could lead to 100 ms delays between all contention sequences.

2. *Triplex Bit FMEA*

During the T bit, any single-point failure will be dynamically masked. Any single-point failure can manifest itself as bad data in one or all channels on a single layer. It may also manifest itself in reception of bad data on all three layers in a single channel. The term "bad data" is interpreted to mean a "1," "0," abort, or noise.

Triplexes—Any such failure will be dynamically masked, each channel of the triplex voting all three layers.

Duplex—A failure may cause a duplex erroneously to lose the contention, but cannot cause the duplex to continue the contention erroneously.

Simplex—A failure may cause a simplex to erroneously lose or win the contention, but the failure cannot propagate to another layer of the network.

3. *Duplex Bit FMEA*

Any single-point failure can manifest itself as bad data in one or all channels on a single layer. It may also manifest itself in reception of bad data on all three layers in a single channel. An error in the previous T bit sequence may have caused a single channel of a triplex to still be participating in the poll sequence. This will at most manifest itself as an error in all channels of a single layer. There may be duplexes and simplexes continuing erroneously in the presence of failures at the end of this bit. A duplex voting a "0" cannot assume that he has won the poll, since another duplex could still be contending in the presence of failures.

4. *Priority Bit FMEA*

Triplexes—Any such failure will be dynamically masked, each channel of the triplex voting all three layers, and then voting the results of their vote (i.e., continue/lose contention status).

Duplex—A failure may cause a duplex to erroneously lose the contention, but cannot cause the duplex to continue the contention erroneously.

Simplex—A failure may cause a simplex to lose or win the contention erroneously, but the failure cannot propagate to another layer of the network.

5. *Summary of Failure Modes*

BIT S: one and only one erroneous channel may start/continue/dropout erroneously.

BIT T: one and only one erroneous channel may continue to contend.

BIT D: no duplexes will continue erroneously; erroneous continue of simplexes on L, M or N; erroneous dropout of simplexes on L, M, or N; erroneous win contention of simplexes on L, M, or N.

End of STD bit sequence failure modes: The following are the failure modes exhibited at the end of the STD bit poll sequence: duplexes on LM, LN, or MN and simplexes on L, M, or N still contending; triplexes on LMN and simplexes on L, M, or N still contending.

D. Fault Insertion Examples

1. *Successful Completion in the Presence of a Fault*

The following represents an example of a simulated fault scenario. There are two triplexes and a duplex contending for the network. Triplex A has the priority vector (1,1), triplex B, (0,1), and duplex C (1,0). Triplexes A and B are connected to all three layers, L, M and N, and duplex C is connected to layers M and N.

Triplex A	Triplex B	Duplex C
Priority: 11	Priority 01	Priority: 10
Layers: LMN	Layers: LMN	Layer: MN

Fault: All channels at all sites always receive a "1" from layer L. (Fault in IC node layer L transmitter.)

Triplex A	**Triplex B**	**Duplex C**
S BIT		
Xmit "1" on LMN	Xmit "1" on LMN	Xmit "1" on MN
Rcv L = 1, M = 1, N = 1	Rcv L = 1, M = 1, N = 1	Rcv L = 1, M = 1, N = 1
Start contending	Start contending	Start contending
T BIT		
Xmit "1" on LMN	Xmit "1" on LMN	Xmit "0" on MN
Rcv L = 1, M = 1, N = 1	Rcv L = 1, M = 1, N = 1	Rcv L = 1, M = 1, N = 1
Voted LMN = "1"	Voted LMN = "1"	Voted LMN = "1"
Continue	Continue	Drop out
(skip to priority sequence bits)		
PBIT1		
Xmit "1" on LMN	Xmit "0" on LMN	
Rcv L = 1, M = 0, N = 0	Rcv L = 1, M = 1, N = 1	
Voted LMN = "0"	Voted LMN = "1"	
Win bus	Drop out	

As expected, triplex A, having the higher-priority vector, wins the contention after the first priority bit.

2. *Unsuccessful Completion in the Presence of a Fault*

The following represents an example of a simulated fault scenario in which a duplex erroneously wins the bus. There are two duplexes and a simplex contending for the network. Duplex A has the priority vector (1,1), duplex B (0,1), and simplex C (1,0). Duplexes A and B are connected to layers M and N, and simplex C is connected to layer L.

Duplex A	Duplex B	Simplex C
Priority: 11	Priority: 01	Priority: 10
Layers: MN	Layers: MN	Layer: N

Fault: Duplex B and simplex C always receive a "0" from layer N. (Fault in IC node layer N transmitter.)

Duplex A	Duplex B	Simplex C
S BIT		
Xmit "1" on MN	Xmit "1" on MN	Xmit "1" on L
Rcv L = 1, M = 1, N = 1	Rcv L = 1, M = 1, N = 0	Rcv L = 0, M = 1, N = 0
Start contending	Start contending	Start contending
T BIT		
Xmit "0" on MN	Xmit "0" on MN	Xmit "0" on N
Rcv L = 0, M = 0, N = 0	Rcv L = 0, M = 0, N = 0	Rcv L = 0, M = 0, N = 0
Voted LMN = "0"	Voted LMN = "0"	Voted LMN = "0"
Continue	Continue	Continue
D BIT		
Xmit "1" on MN	Xmit "1" on MN	Xmit "0" on N
Rcv L = 0, M = 1, N = 1	Rcv L = 0, M = 1, N = 0	Rcv L = 0, M = 1, N = 0
Voted LMN = "1"	Voted LMN = "0"	ORed LMN = "1"
Continue	Win contention	Drop out
(skip to priority sequence bits)		
PBIT1		
Xmit "1" on MN		
Rcv L = 0, M = 0, N = 0		
Voted LMN = "0"		
Win bus		

This example shows that although we can guarantee fault-masking for contention and communication between triplexes, we cannot guarantee this for sites of lower redundancy. Since the interconnectivity of duplex and simplex sites does not satisfy the rigorous theoretical requirements for Byzantine resilience, we can make few assumptions about their ability to communicate under faulty conditions. However, since we are guaranteed communication between triplexes and since the network manager always runs on a triplex site, communication should be able to be restored after quick diagnosis and reconfiguration by the network manager.

E. Simulation of the Contention Protocol

The AIPS contention protocol has been simulated on the AIPS symbolic simulator.[6] The symbolic structure of the simulator holds many advantages over traditional simulations. The basic building blocks and the interconnection methodology of the AIPS system have been predefined in the simulator. Using the predefined symbolic structure of AIPS, the user may change, add to, or respecify the definition or interconnectivity of AIPS components. The user is always working directly with AIPS objects, allowing for very rapid prototyping of arbitrary AIPS configurations and testing of algorithms and variations on the system. The stringent fundamental requirements of the AIPS system require the validation and understanding of the system under both fault-free and single-failure conditions.

The AIPS simulation is being done on a Symbolics 3600 LISP machine. The key behind the elegance of the LISP machine is its object-oriented programming environment. The Symbolics LISP dialect, ZetaLisp, implements object-oriented programming through the flavor system to create and manipulate objects. Communication between objects is accomplished through message-passing paradigms. The ability to create objects and build these objects into a structure representative of the actual system to be simulated allows for a very high-resolution structured simulation. Objects are defined to represent computers and communication nodes in the network. Each of these is further composed of objects such as CPU, I/O processor, and network interface. The interactive user interface allows single stepping and dynamic inspection of any object in the system. The user explicitly sees the effects of failures on the system.

To allow for automatic insertion of faults, each object in the system is responsible for its own faulty behavior. An object can be set to exhibit a particular failure mode. Thus, we keep a list of fault insertion points and possible modes of failure for each object and also a list of different operational modes of the system. This allows us to achieve automatic fault insertion by only defining faulty behavior for each object type. The simulation sequentially inserts every possible fault in every possible operational mode, recording results for later analysis.

1. *Object-Oriented Simulation Methodology*

The first step in building an object-oriented simulation is to define the objects to be used in the simulation. In ZetaLisp one defines objects through the flavor system. This system allows one to define a type of object and a set of generic operations that can be performed on any object of that type.

Five types of basic objects make up the lowest level of AIPS object definitions. They are the generic buffer, the ORing buffer, generic state machine, event queue, and timer queue. The flexibility of these objects is enhanced by the symbolic nature of the LISP programming environment. Lists and arrays of program statements may be constructed and then conditionally evaluated under program control. More complex objects may be built by combining objects to form "macro" objects. Macro objects can be built from other macro objects, yielding larger and larger structures, yet the lowest-level components in any object are of the five basic types.

These five basic types of objects may now be used to build macro objects that correspond directly to any of the AIPS building block components. Macro objects representing AIPS building block components may then be linked to form a particular AIPS configuration for simulation.

For example, the node port in Fig. 3 is modeled with a generic buffer to represent the receiver, protocol decoder, and receive FIFO. An ORing buffer is used to model the regeneration logic and transmitter. A generic state machine object is used to model the sequencer, port activity register, message buffer, and port enable logic. Similarly, the AIPS GPC's are represented by a collection of generic state machine objects to perform the functions of dedicated hardware, an event queue and timer queue object to represent the processor, and generic buffers for the interface to the IC network. IC node and GPC object macros can then be created and configured into a single large object representing the whole AIPS configuration to be simulated.

2. *Automatic Fault Insertion*

The ability to insert faults automatically in the simulation is important because of the system's functional requirements under faulty and fault-free conditions. In a highly complex, distributed system such as AIPS, this becomes laborious unless it is dealt with at the lowest level of definition of the network.

The approach used to provide for fault insertion is to build fault insertion points into each of the five basic object types upon which all simulation structures are built. Each of these five objects has an initialization function which, when an object of that type is created, adds itself to a fault insertion point list. For each fault list member, the entry has a pointer to the object and a list of failure modes supported by that object. The generic and wired-OR buffers support four different failure modes. They are failed-0, failed-1, failed-Byzantine-0, and failed-Byzantine-1.

Failed-0—When a buffer is set to exhibit the failure mode failed-0, the buffer acts in the normal manner of enqueueing and dequeueing messages, but instead of forwarding the dequeued data to its destination buffers, it forwards a zero value. This mode emulates a communication line being stuck at zero.

Failed-1—Similarly, the failed-1 mode forwards a "1" to all destination buffers emulating a stuck at 1 fault.

Failed-Byzantine-0—The failed-Byzantine-0 faults emulate failures in which several of the buffers in the destination list receive valid data and the remaining buffers receive 0's.

Failed-Byzantine-1—The failed-Byzantine-1 faults emulate failures in which several of the buffers in the destination list receive valid data and the remaining buffers receive 1's.

The Byzantine failure modes are particularly insidious and are of special concern in ultrahigh-reliability architectures such as AIPS.

The buffer failures are of the most interest for two reasons. First, communication failures are the predominant types of failures in a large distributed network. Second, many types of failures within a GPC will be manifested in a manner identical to that of a failed buffer. Other objects in the simulation, the generic state machine, the event queue, and the timer queue, exhibit two failure modes: the failed-passive mode, in which the object does nothing, and the failed active mode, in which the object exhibits pseudorandom behavior.

In addition to a list of faults to be simulated, a list of initial conditions is generated. For example, in modeling contention for the IC bus, we generate all possible permutations of sites contending for the bus and insert all possible faults for all possible initial conditions.

Special routines needed to be written to collect results from fault insertion runs. For modeling of the IC bus contention, statistics such as erroneous bus possession and erroneous loss of bus possession are kept for each of the different levels of redundancy. These are determined by looking at the system state after running the simulation of the contention algorithm with a particular fault for some initial condition. The fault injection code determines which site should have won the contention, then queries each site, determining which sites actually won or lost bus possession. The two are compared and result statistics updated if a discrepancy exists.

The following statistics are automatically logged and summarized for each simulation run:

Triplex error loss: A majority of channels (2 of 3, or 3 of 3) of a triplex lost the contention poll erroneously.

Triplex error win: A majority of channels of a triplex won the contention poll erroneously.

Triplex split: A single channel of a triplex GPC either won or lost the contention poll erroneously while the remaining two channels terminated the poll correctly.

Duplex error loss: Both channels of a duplex lost the contention poll erroneously.

Duplex error win: Both channels of a duplex won the contention poll erroneously.

Duplex split: One channel of a duplex won the contention poll and the other channel lost the contention.

Simplex error win: A simplex GPC won erroneously.

Simplex error loss: A simplex GPC lost erroneously.

3. *Results from Automatic Fault Insertion*

Once constructed, the simulation of the contention scheme for the AIPS of Fig. 1 contained 243 fault insertion points. Typical fault insertion points consist of GPC and IC node link transmitters, receivers, and buffers. For example, a fault might be inserted in link transmitter 3 of IC node 5. For each insertion point, there were between two and four failure modes, yielding a total of 816 unique faults that could be inserted into the system. The behavior and results of an access contention poll are also determined by the number, redundancy level, connectivity, and priority of the various sites contending for network access. A fault inserted during a network contention poll combination of a particular combination of sites, priorities, and connectivity is termed a fault scenario. To limit the possible fault scenarios, simulations were performed with fixed AIPS configurations. A particular AIPS system was defined in the simulator with a fixed number of sites and connectivity. Each site also had a fixed priority vector. However, from the fixed configuration, any possible combination of sites could contend, the other sites remaining passive. For the AIPS configuration of Fig. 1, with five sites, there are 31 different permutations of various sets of sites that can contend for the bus. Each fault can be simulated in each permutation, yielding a total of 25,296 fault scenarios.

The column "Run A" of Table 1 summarizes the results of the injection of "1" and "0" hard faults in 198 different points in the proof-of-concept configuration for 31 different permutations of contention configurations for a series of 12,276 faults. The column "Run B" of Table 1 sumarizes the results of the injection of "1," "0," and byzantine faults in 198 different points for 33 different permutations of contention configurations of a 13-site AIPS system (three triplexes, five duplexes, and five simplexes) for a series of 26,136 fault scenarios.

During the automatic insertion runs, a fault by fault detailed list of results was kept as well as the global list of results. The following illustrates one line from the detailed list:

SIMPLEX ERROR LOSS SITE#5 ICIS-STATE: LOST-LAST

Fault Mode: Failed-1 Inserted at Generic Buffer 2 in IC Node 3 Layer L

This indicates that the simplex site 5 erroneously lost the access contention poll. The fault inserted was at generic buffer 2 in IC node 3 on layer L and was exhibiting a stuck at "1" failure mode. This detailed error dump information was kept for each fault scenario that did not terminate correctly. From this information, we can analyze and recreate each fault scenario as well as determine tendencies exhibited by faulty behavior, such as all simplex erroneous losses being accompanied by another erroneous simplex win. The AIPS symbolic simulator provides the ability to step interactively through a particular fault scenario examining objects (buffers, transmitters, receivers, controller, etc.) at each step of the algorithm so that the system engineer can recreate and fully understand the

Table 1 Automatic fault insertion results

	Run A	Run B
Triplex error loss	0	0
Triplex error win	0	0
Triplex split	0	0
Duplex error loss	203	54
Duplex error win	0	0
Duplex split	12	63
Simplex error loss	25	49
Simplex error win	10	52

effect of and reasons behind the behavior of a particular fault scenario.

Analysis of the results from the automatic fault insertion runs depicted in Table 1 shows several tendencies in the error pattern. In particular, no duplex error losses were accompanied by a duplex error win. Also, no duplex splits were accompanied by an error win or error loss, and all simplex error losses were accompanied by an error win.

Note that although many thousand's of fault scenarios were tested, not all possible error possibilities were exhibited. For example, there were no triplex splits and no duplex error wins. The second example of Sec. IV. D demonstrates a scenario that leads to a duplex erroneously winning the contention poll, along with the duplex that should have won the poll, resulting in two duplexes possessing the bus at the same time. This error did not occur because the particular combination of connectivity and site priorities did not arise.

V. Summary and Conclusions

The issue of reliable intercomputer communication in a graded redundancy distributed system is crucial for information processing systems of advanced vehicles. The triply redundant intercomputer network for the Advanced Information Processing System has been discussed here. The AIPS intercomputer network provides a high-speed Byzantine resilient communication service between processing sites that are at least triply redundant. This service is provided even in the presence of arbitary failures of simplex and duplex processing sites that are on the intercomputer network. The intercomputer network contention poll has evolved from the Laning poll that has been used to resolve bus contention between multiple triply redundant sites in systems such as the fault tolerant multiprocessor. The analysis of failure modes and effects and the detailed simulation of the AIPS contention poll has demonstrated a robustness concomitant with the AIPS concept. The bus contention hardware has been designed and fabricated, and testing of the contention hardware and algorithm is to begin shortly.

Acknowledgment

This work was sponsored by NASA Langley Research Center under Contract NAS1-18061.

References

[1]Lala, J. H., "Advanced Information Processing System: Fault Detection and Error Handling," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AIAA, New York, Aug. 1985, pp. 587–596.

[2]Lala, J. H., "Advanced Information Processing System," *Proceedings of the 6th AIAA/IEEE Digital Avionics Systems Conference*, AIAA, New York, 1984, pp. 199–210.

[3]Smith, T. B., "Fault Tolerant Processor Concepts and Operation," Charles Stark Draper Lab., CSDL-P-1727, May 1983.

[4]Lala, J. H. and Adams, S. J., "Inter-Computer Communication Architecture for a Mixed Redundancy Distributed System," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, AIAA, New York, 1987, pp. 1571–1578.

[5]Smith, T. B. and Lala, J. H., "Development and Evaluation of a Fault-Tolerant Multi-Processor (FTMP) Computer," Vol. I, NASA CR-166071, May 1983.

[6]Adams, S. J. and Lala, J. H., "Object Oriented Simulation of a Mixed Redundancy Distributed System," *Proceedings of the 1986 Summer Computer Simulation Conference*, Society for Computer Simulation, 1986, pp. 129–133.