An Age-Old Question for Tomorrow: Who Will Guard the Guardians ?

Algirdas Avižienis

Distinguished Professor Emeritus,
University of California, Los Angeles,
and
Vytautas Magnus University
Kaunas, Lithuania

IFIP WG 10.4,Martinique, January 27, 2012

Major Causes of System Failures

1. Permanent physical failures (changes) of hardware components
2. Interference by external environmental factors: cosmic rays, EM radiation, excessive temperature, etc.
3. Previously undetected design faults (“bugs”, “errata”, etc.) in hardware and software components of a system that cause errors during operation
4. Malicious actions by humans that alter or stop delivery of expected service
5. Unintentional mistakes by human operators or maintenance personnel that lead to loss or undesirable changes of expected service

The Deficiencies of Current Defenses

1. There are unprotected “hard core” elements, especially in the error detection and recovery management hardware and software
2. Hardware and software defenses are interdependent, thus both have to succeed in order to complete recovery
3. There is no built-in support for multi-channel computing with or without design diversity that provides tolerance of design faults

An Example of “hard core” and Interdependence

Pentium and Itanium processors have a Machine Check Architecture (MCA) in which hardware errors are recorded by setting bits in a set of MCA registers that are **not** protected by any form of redundancy or fault tolerance
The operating system then senses the MCA register bits and initiates recovery action

A Contemporary Paradox

Computing systems provide protective infrastructures
for critical infrastructures of modern society:
electrical power, telecommunications, transportation,…
but:
these computing systems do not possess
a protective infrastructure of their own!

My Goal:

To design a *hardware-based fault-tolerant resilience infrastructure* for computing and communication systems, because it is needed as systems progress towards ever higher complexity and speed of operation and are employed in life-critical applications

Why “Resilience” Infrastructure ?

- *Resilience* is the dependability of a system when it is subjected to unforeseen changes of its structure (manifestation of unknown design faults in hardware and software) and/or of its environment (intensive radiation, temperature fluctuations, etc.)
- The goal of the RI is to protect the system when unforeseen changes occur
- The term “robustness” is also used, especially in the terminology of embedded systems, also “survivability”

Why All Hardware and Firmware ?

Because over the past half century hardware and firmware have not been adequately exploited to assure the dependability of computing and communication systems and because all software faults and vulnerabilities are avoided

Desirable Properties of the Resilience Infrastructure

- The RI is generic, i.e., suitable for a variety of "client" systems
- The RI is transparent to the client's software, but communicates with it
- The RI is compatible with and able to support the client's other defenses
- The RI is fully self-protected by fault tolerance, immune to the client's faults and to malicious software

Four Building Blocks of the RI

1. *The Adapter (A-node)*: it provides the interface of the RI to the components (C-nodes) of the "client" system being protected

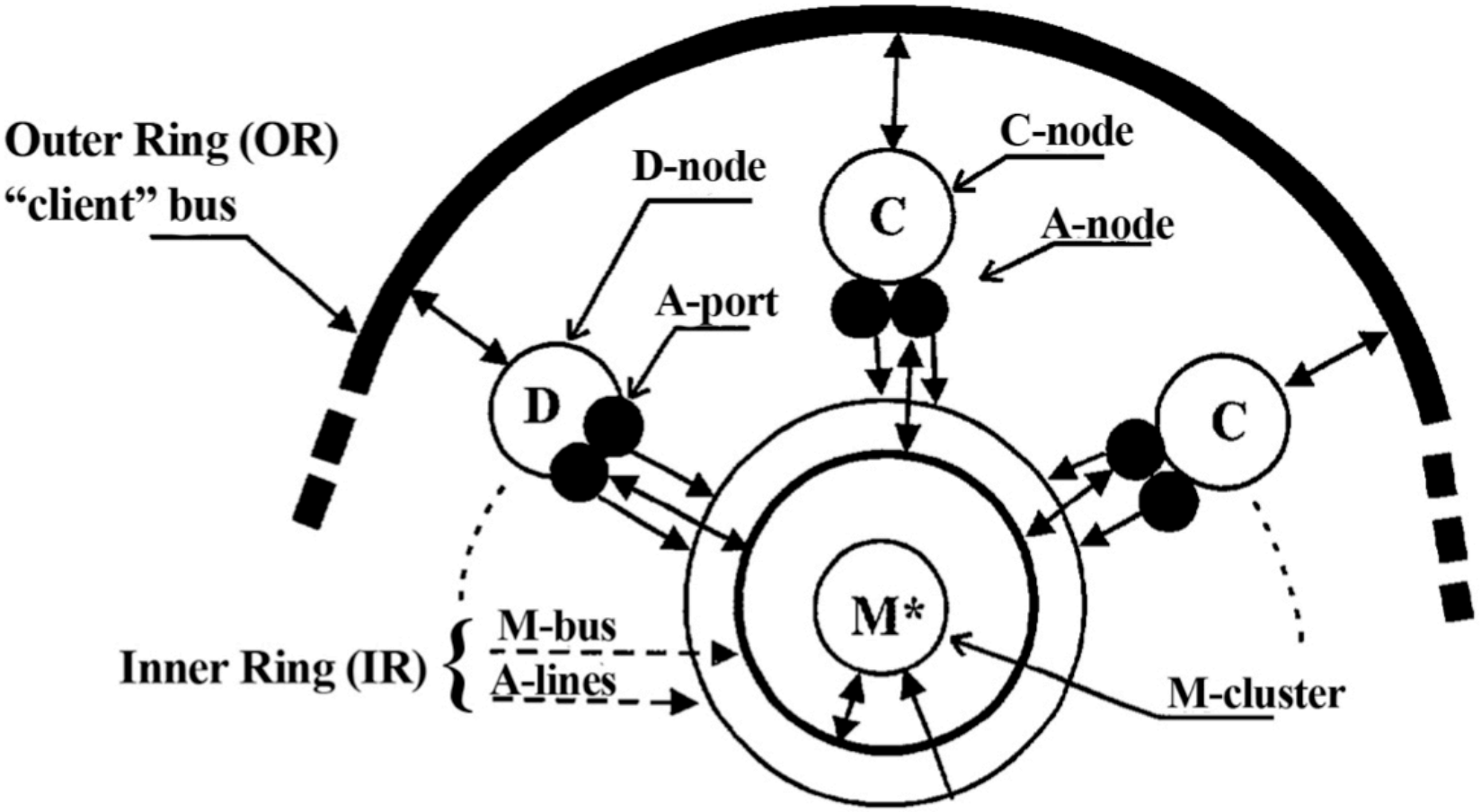
2. *The Monitor (M-node)*: it receives error messages from the C-nodes and issues predetermined responses via the A-node

3. *The Startup, Shutdown and Survival node (S3-node):* it responds to catastrophic events by shutting down the system and protecting critical information needed for startup, also manages M-node self-repair

4. *The Decision node (D-node)*: it provides support for error detection in the "client" system and implements communication between the RI and the "client" system

All four building blocks must be implemented in hardware and firmware of proven dependability and are of relatively low complexity.

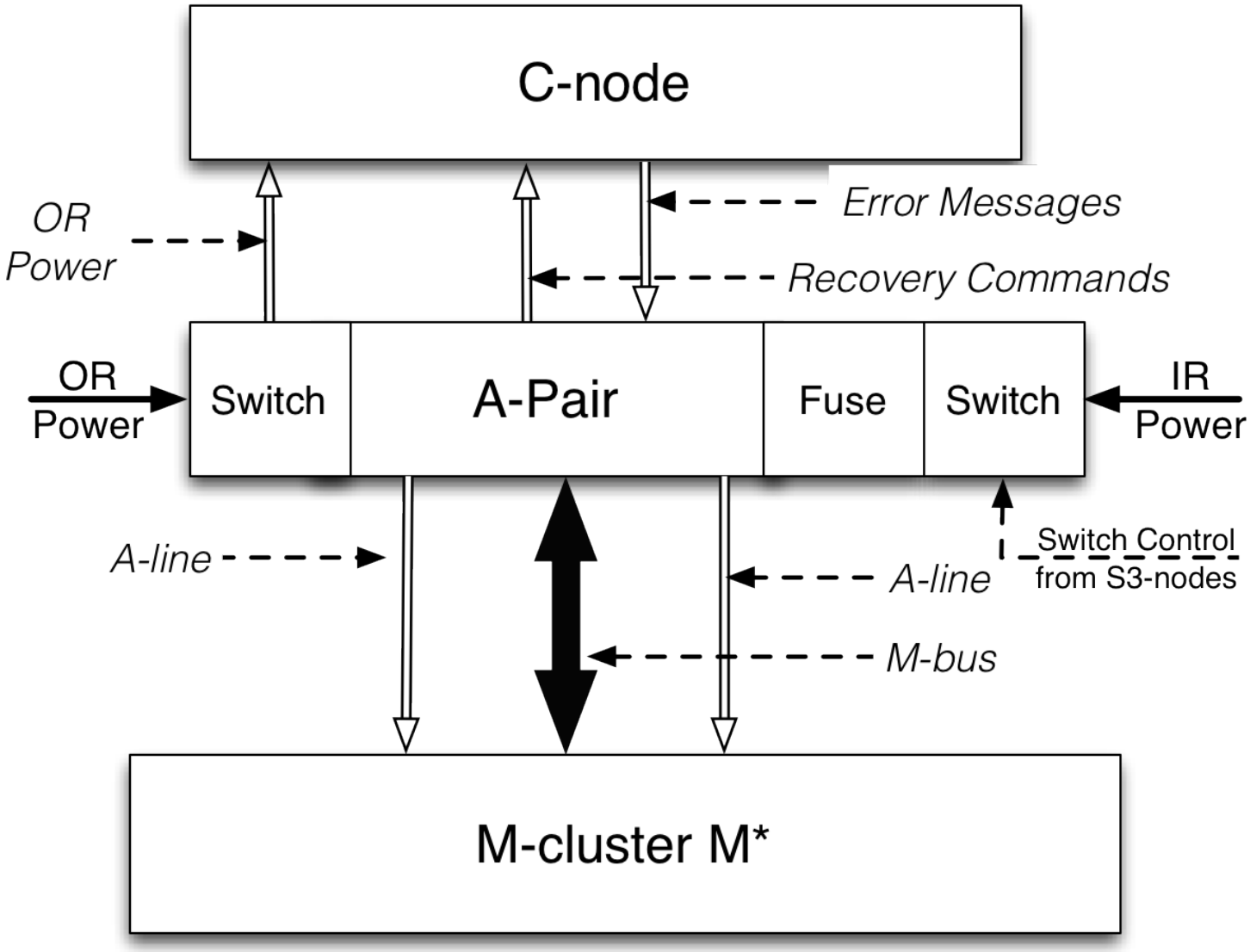
The “Ring” Representation of RI



The Adapter (A) Node

1. Transmits Error messages from C-node to M-cluster
2. Transmits Recovery commands from M-cluster to C-node
3. Controls the Power switch of the C-node according to commands from M-cluster
4. Permanently disconnects its IR power by means of the Fuse upon a command from the M-cluster
5. Uses the A-line to the M-cluster to report its own status and to request access to the M-bus
6. Fault tolerance of the A-node is implemented as a “self-checking pair” of A-nodes
7. Messages from the M-cluster are voted in the A-node

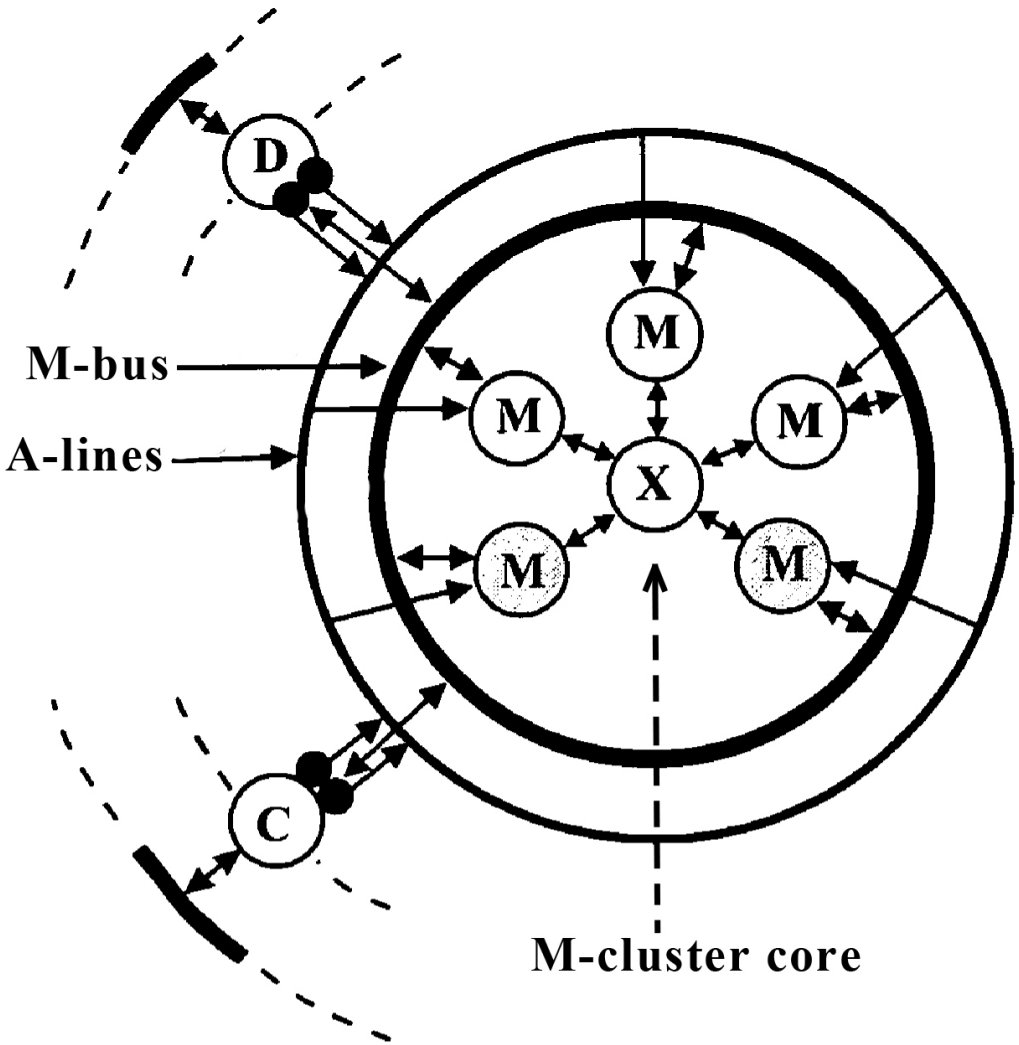
The A-node: RI Interface with C-node



The Monitor (M) Node

1. Is a member of the M-cluster M* of 3 or more M-nodes operating in the TMR (triplication with voting) mode with unpowered spares
2. Receives Status (on A-lines) and Error (on M-bus) messages from all A-nodes
3. Selects appropriate Recovery command from its ROM
4. Sends the Recovery command to the A-node via the M-bus
5. Communicates to other M-nodes by means of the Intra-Cluster (IC) bus
6. ROM data in the M-node consists of:
   (a) Recovery command for every Error message (supplied by the "client" system designers
   (b) Sequences for M-node recovery and replacement in the M-cluster with S3-node assistance
   (c) Recovery sequences for catastrophic events that are initiated by the S3-node array S3*

The Inner Ring IR



Dynamic Data in the M-node Memory

1. Outer Ring ('client") configuration status: list of C-nodes (active, spare, failed).
2. M-cluster status: list of M-nodes (active, spare, failed)
3. System time

Note: the above data is also stored in every S3-node

4. Current activity: buffer storage for Error messages being serviced and service requests from A-nodes that are waiting
5. The log of Inner Ring activity

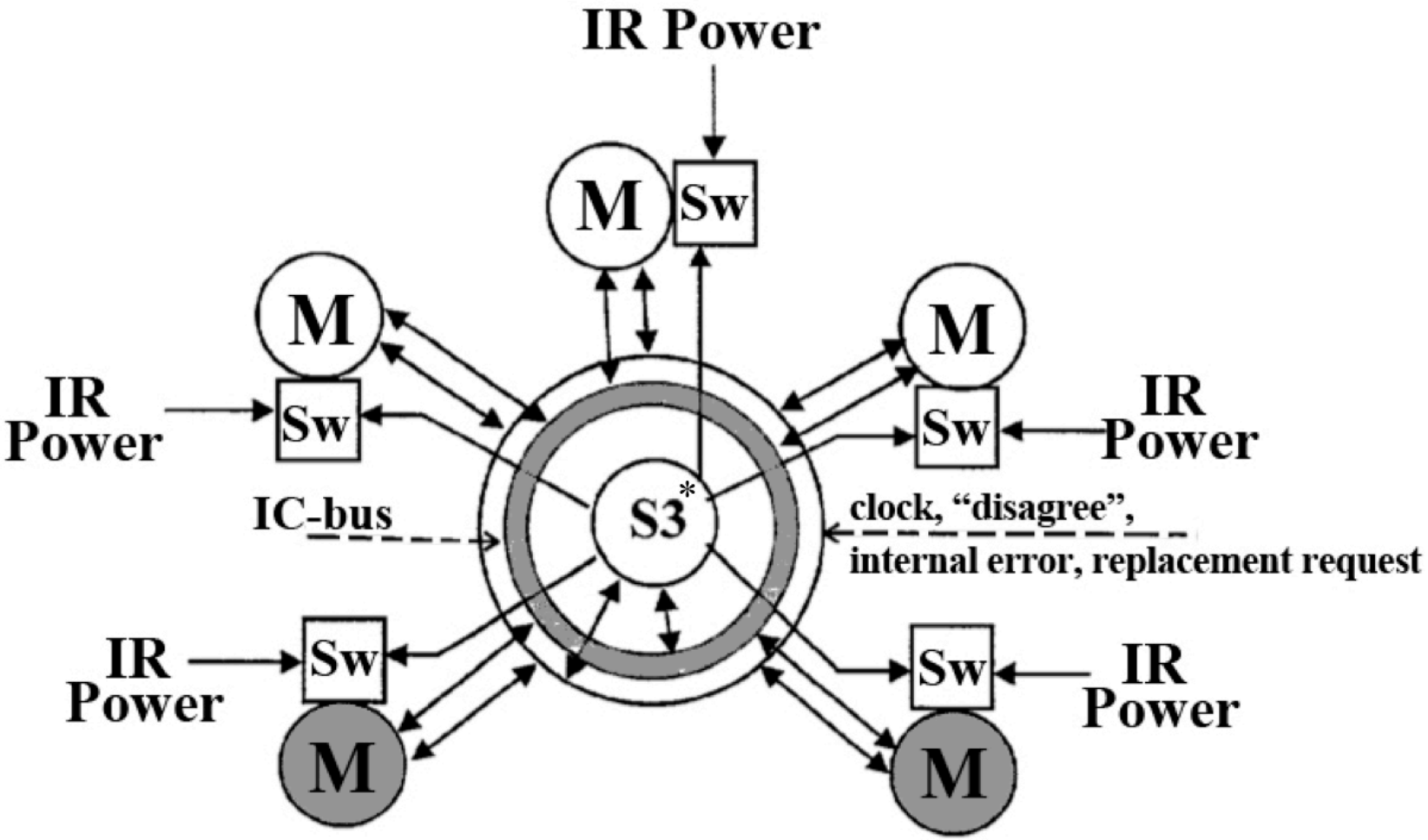
Fault Tolerance of the M-cluster M*

1. M* depends on the S3-node array S3* for spare M-node activation and the removal of power from failed M-nodes
2. Spare M-bus lines and error-coded messages and commands are employed
3. M* degrades to two and one M-node after spares are used up
4. The initial configuration of M* is triplication with voting and unpowered spares

The Startup, Shutdown, and Survival (S3) Node

1. The Array S3* of S3 nodes supports the recovery of the "client" system (C-nodes) and of the entire Resilience Infrastructure RI from catastrophic events: intensive radiation, temporary instability of power supplies, fluctuations of temperature, physical damage to system hardware, etc.
2. S3* controls Power-on and Power-off sequences for RI and the "client"
3. S3* provides fault-tolerant clock signals for the RI
4. S3* keeps System Time and System Configuration in non-volatile, radiation-hard registers
5. S3* controls Power switches of M-nodes and Inner Ring power to A-pairs to support M-cluster recovery
6. One S3-node is a "self-checking pair" that inhibits its output upon disagreement
7. An Array S3* of N S3-nodes will tolerate N-1 failures of individual S3-nodes when their outputs are connected in a logical Or
8. Each S3-node is provided with a backup power source (a battery) in order to tolerate temporary loss or instability of Inner Ring power

The M-Cluster M*

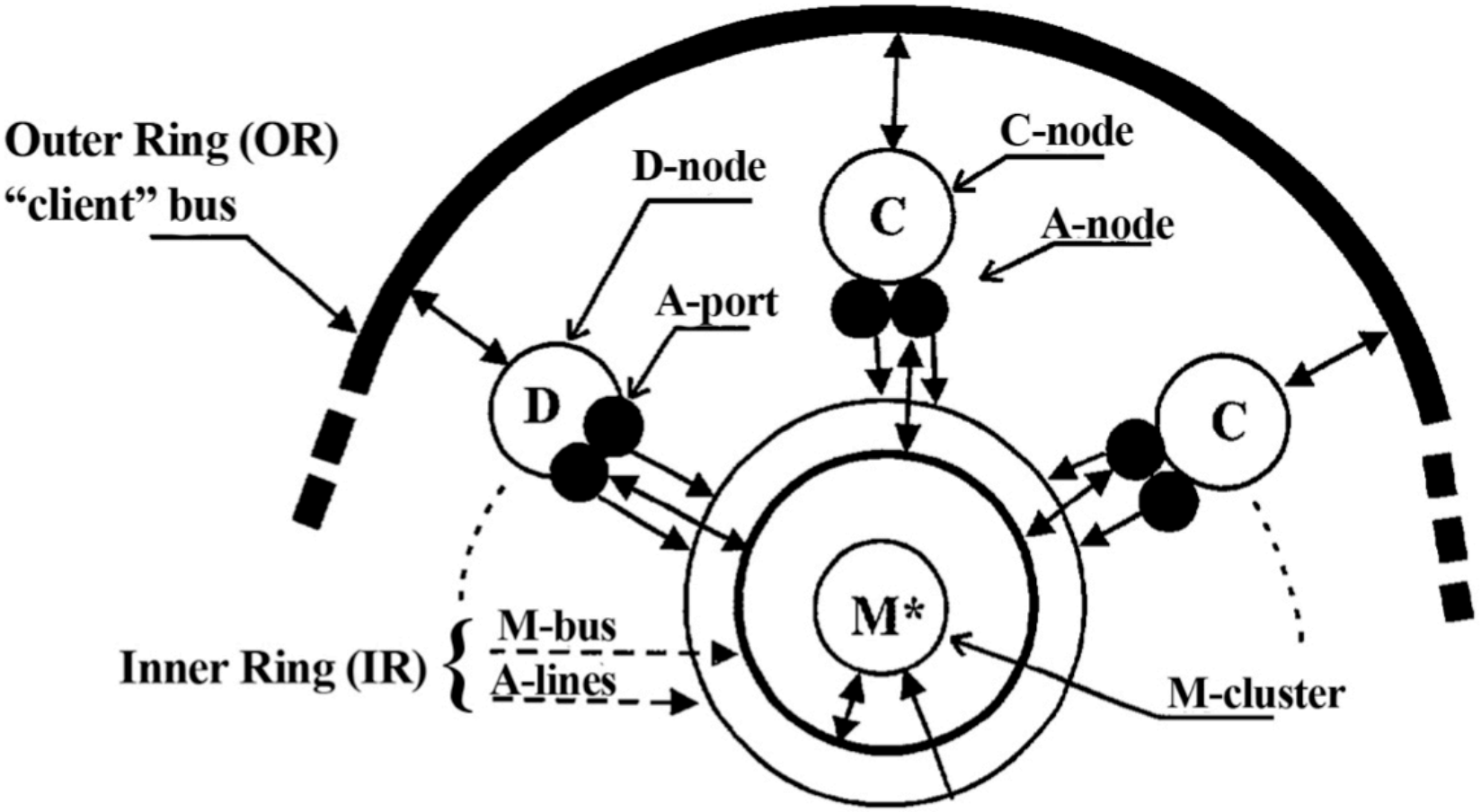


S3 is the S3 node array*

The Decision (D) Node

1. The D-node is a communication link between the Outer Ring C-nodes and the M-cluster
2. The D-node can provide an external voting and comparison service for C-nodes
3.The D-node can provide decision algorithms for N-version software that executes on diverse C-node processors
4. The D-node can log disagreement data on the various decisions
5. The D-node employs only firmware (no software) and is implemented as a self-checking pair for fault tolerance.
6. A-nodes are replaced by built-in A-ports that have all A-node functions.
7. The M-cluster and the S3-node array treat the D-nodes like the C-nodes.

The “Ring” Representation of RI



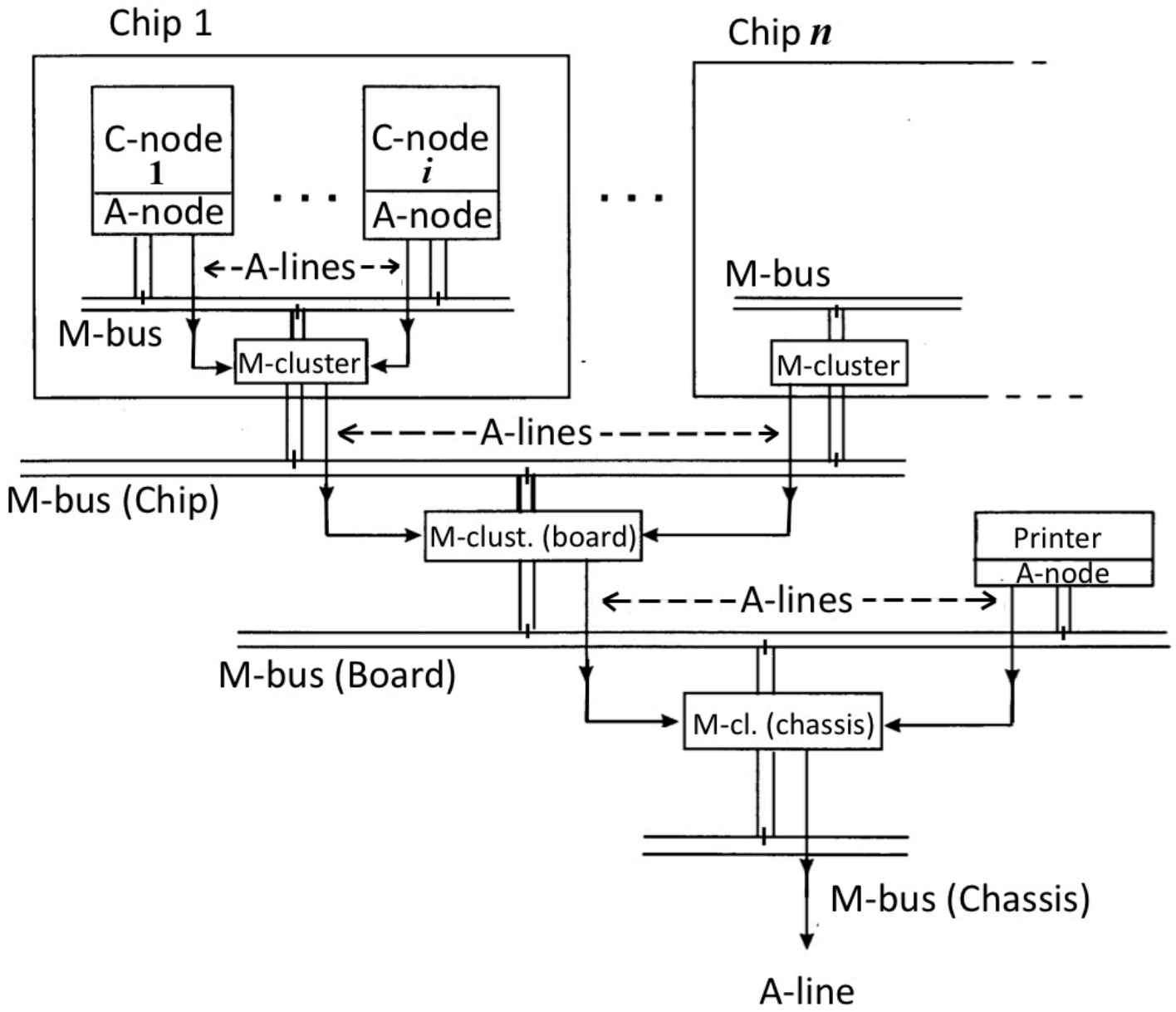
A Hierarchical Structure for the Resilience Infrastructure

One M-cluster is limited to servicing some number N (say 8) of C and D nodes. Then a chip with 64 cores would need 8 M-clusters that have A-node functions, and those M-clusters would be connected to a second-level M-cluster that receives inputs from the 8 first-level M-clusters.

More generally, we can consider a RI structure that has multiple levels. The figure shows a three-level structure with M-clusters located at Chip, Board, and Chassis levels. The structure can be extended to more levels. At the Chassis level peripheral equipment can be connected – the figure shows a printer with its own A-node.

Every M-cluster has its own S3-array, therefore if any connection to a lower level is disrupted, the branch that has been disconnected will function independently until it fails or reestablishes the connection to the rest of the tree. At that time the branch has to accept the critical parameters of that tree. That becomes important if the M-bus and A-lines are wireless links and the C-nodes are mobile.

Hierarchical Structure of RI



Advantages of the Resilience Infrastructure

- Design exercise shows very moderate complexity
- Software is not used at all
- D-node enhances “client” fault tolerance
- “Client” defenses may be simplified, offloading some on RI
- Separation of RI and client allows fault-tolerant design of RI
- Every “client” customizes the RI for its needs
- Hierarchical structure of RI allows protection of large, heterogeneous systems
- The RI guards the guardians, that is, the RI attempts recovery when the defenses of the “client” are defeated and it calls for help

Will the Resilience Infrastructure Be Used ?

- I believe that the RI offers significant advantages, compared to the current implementations of fault tolerance, BIST, software monitors, and other defenses. The absence of software removes the source of many problems, and the complexity of the RI hardware is quite modest.
- However, there is a huge “legacy” problem – system design is a gradual process that cannot readily accommodate a big change – the transition to the inclusion of the resilience infrastructure.
- For the above reason I have identified the human exploration of Mars as a project that is sufficiently far in the future and also life-critical. For this reason the use of the RI could be considered in the initial definition of the system.
- It is important to note that any fault tolerance or other protective means of that future system can be accommodated – the RI is an additional feature that will guard those guardians when a catastrophic event threatens with disaster, although the RI can handle simpler dangers as well

The Human Mission to Mars

- Important studies have been done over the past 20 years:

  D. Landau and N.J. Strange "This Way to Mars", *Scientific American,* 306(6): 58-65 (December 2011).

  "Special report: Sending astronauts to Mars", *Scientific American,* 282(3): 40-63 (March 2000).

- My concern: the survival of the spacecraft for the 1000-day human mission to Mars

- Assume: spacecraft systems are controlled by embedded computers that have state-of-the-art fault tolerance

- Remaining need: Assure resilience (robustness) of the systems:
- provide defenses against unexpected, possibly catastrophic faults and make the defense mechanisms self-protecting

- My solution: Provide a hardware-based, fault-tolerant resilience infrastructure (RI) for all spacecraft systems

References

[1] Avizienis, A.,"Self-Testing and -Repairing Fault Tolerance Infrastructure for Computer Systems",
United States Patent No. US 7,908,520 B2
March 15, 2011

[2] Avizienis, A. "Hierarchical Configurations in Error-Correcting Computer Systems",
United States Patent No. . US 7,861,106 B2
December 28, 2010