An Immune System Paradigm for the Assurance of Dependability of Collaborative Self-organizing Systems

Algirdas Avižienis
Vytautas Magnus University, Kaunas, Lithuania
and
University of California, Los Angeles, USA

Abstract. In collaborative self-organizing computing systems a complex task is performed by relatively simple autonomous agents that act without centralized control. Disruption of a task can be caused by agents that produce harmful outputs due to internal failures or due to maliciously introduced alterations of their functions. The probability of such harmful outputs is minimized by the application of a design principle called ”the immune system paradigm” that provides individual agents with an all-hardware fault tolerance infrastructure. The paradigm and its application are described in this paper.

1 Dependability Issues of Collaborative Self-Organizing Systems

Self-organizing computing systems can be considered to be a class of distributed computing systems. To assure the dependability of conventional distributed systems, fault tolerance techniques are employed [1]. Individual elements of the distributed system are grouped into clusters, and consensus algorithms are implemented by members of the cluster [2], or mutual diagnosis is carried out within the cluster.

Self-organizing systems differ from conventional distributed systems in that their structure is dynamic [3]. Relatively simple autonomous agents act without central control in jointly carrying out a complex task. The dynamic nature of such systems makes the implementation of consensus or mutual diagnosis impractical, since constant membership of the clusters of agents cannot be assured as the system evolves. An agent that suffers an internal fault or external interference may fail and produce harmful outputs that disrupt the task being carried out by the collaborative system. Even more harmful can be maliciously introduced (by intrusion or by malicious software) alterations of the agent’s function that lead to deliberately harmful outputs.

The biological analogy of the fault or interference that affects an agent is an infection that can lead to loss of the agent's functions and also to transmission of the infection to other agents that receive the harmful outputs, possibly causing an epidemic. The biologically inspired solution that I have proposed is the introduction within the agent of a fault tolerance mechanism, called the fault tolerance infrastructure (FTI), that is analogous to the immune system of a human being [4,5]. Every agent has its own FTI and therefore consensus algorithms are no longer necessary to protect the system.

2 A Design Principle: the Immune System Paradigm

My objective is to design the FTI for an autonomous agent that is part of a self-organizing system. I assume that the agent is composed of both hardware and software subsystems and communicates to other agents via wireless links. Then I will employ the following three analogies to derive a design principle called "the immune system paradigm":

(1) the human body is analogous to hardware,
(2) consciousness is analogous to software,
(3) the immune system of the body is analogous to the fault tolerance infrastructure FTI.

In the determination of the properties that the FTI must possess four fundamental attributes of the immune system are especially relevant [6]:

(1) It is a part of the body that functions (i.e. detects and reacts to threats) continuously and autonomously, independently of consciousness.
(2) Its elements (lymph nodes, other lymphoid organs, lymphocytes) are distributed throughout the body, serving all its organs.
(3) It has its own communication links – the network of lymphatic vessels.
(4) Its elements (cells, organs, and vessels) themselves are self-defended, redundant and in several cases diverse.

Now we can identify the properties that the FTI must have in order to justify the immune system analogy. They are as follows:

(1a) The FTI consists of hardware and firmware elements only.
(1b) The FTI is independent of (that is, it requires no support from) any software of the agent, but can communicate with it.
(1c) The FTI supports (provides protected decision algorithms for) multichannel computing by the agent, including diverse hardware and software channels that provide design fault tolerance for the agent's hardware and software.
(2) The FTI is compatible with (i.e., protects) a wide range of the agent's hardware components, including processors, memories, supporting chipsets, discs, power supplies, fans and various peripherals.
(3) Elements of the FTI are distributed throughout the agent's hardware and are interconnected by their own autonomous communication links.
(4) The FTI is fully fault-tolerant itself and requires no external support. It is not susceptible to attacks by intrusion or malicious software and is not affected by natural or design faults of the agent's hardware and software.

(5) An additional essential requirement is that the FTI provides status outputs to those other agents with which it can communicate. The outputs indicate the state of the agent's health: perfect or undergoing recovery action. Upon failure of the agent's function the FTI shuts down all its outputs and issues a permanent status output indicating failure.

The above listed set of design requirements is called the immune system paradigm. It defines an FTI that can be considered to be the agent's immune system that defends its "body" (i.e., hardware) against "infections" caused by internal faults, external interference, intrusions, and attacks by malicious software. The FTI also informs the other agents in its environment of its state of health. Such an FTI is generic, that is, it can serve a variety of agents. Furthermore it is transparent to the agent's software, compatible with other defenses used by the agent, and fully self-protected by fault tolerance.

A different and independently devised analogy of the immune system is the "Artificial Immune System" (AIS) of S. Forrest and S. A. Hofmeyr [7]. Its origins are in computer security research, where the motivating objective was protection against illegal intrusions. The analogy of the body is a local-area broadcast network, and the AIS protects it by detecting connections that are not normally observed on the LAN. Immune responses are not included in the model of the AIS, while they are the essence of the FTI.

3 Architecture of the Fault Tolerance Infrastructure

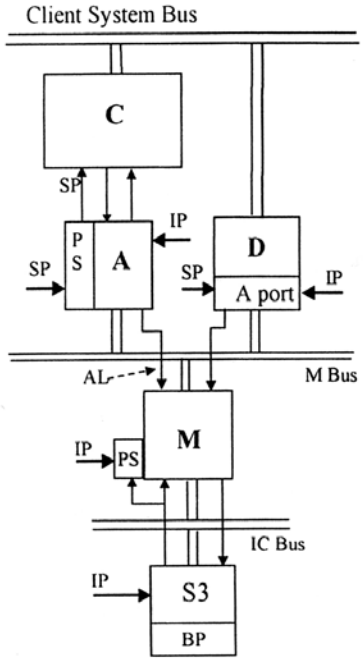
The preceding sections have presented a general discussion of an FTI that serves as the analog of an immune system for the hardware of an agent of a self-organizing system. Such an FTI can be placed on a single hardware component, or it can be used to protect a board with several components, or an entire chassis [5]. To demonstrate that the FTI is a practically implementable and rather simple hardware structure, this and the next section describe an FTI design that was intended to protect a system composed of Intel P6 processors and associated chip sets and was first presented in [5].

The FTI is a system composed of four types of special-purpose controllers called "nodes". The nodes are ASICs (Application-Specific Integrated Circuits) that are controlled by hard-wired sequencers or by read-only microcode. The basic structure of the FTI is shown in Figure 1. The figure does not show the redundant nodes needed for fault tolerance of the FTI itself. The C (Computing) node is a COTS processor or other hardware component of the agent being protected by the FTI. One A (Adapter) node is provided for each C node. All error signal outputs and recovery command inputs of the C node are connected to its A node. Within the FTI, all A nodes are connected to one M (Monitor) node via the M (Monitor) bus. Each A node also has a direct input (the A line) to the M node. The A nodes convey the C node error messages to the M node. They also receive recovery commands from the M node and issue them to C node inputs.

The A line serves to request M node attention for an incoming error message. The M node stores in ROM the responses to error signals from every type of C node

and the sequences for its own recovery. It also stores system configuration and system time data and its own activity records. The M node is connected to the S3 (Startup, Shutdown, Survival) node. The functions of the S3 node are to control power-on and power-off sequences for the entire agent, to generate fault-tolerant clock signals and to provide non-volatile, radiation-hardened storage for system time and configuration. The S3 node has a backup power supply (e.g. a battery) and remains on at all times during the life of the FTI.

The D (Decision) node provides fault-tolerant comparison and voting services for the C nodes, including decision algorithms for N-version software executing on diverse processors (C-nodes). Fast response of the D node is assured by hardware implementation of the decision algorithms. The D node also keeps a log of disagreements in the decisions. The second function of the D node is to serve as a communication link between the software of the C nodes and the M node. C nodes may request configuration and M node activity data or send power control commands. The D node has a built-in A node (the A port) that links it to the M node.Another function of the FTI is to provide fault tolerant power management for the entire agent system, including individual power switches for every C node, as shown in Figure 1. Every node except the S3 has a power switch. The FTI has its own fault-tolerant power supply (IP).



SP: System Power
IP: Infrastructure Power
BP: Backup Power
PS: Power Switch
C: Computing Node
A: Adapter Node
D: Decision Node
M: Monitor Node
S3:Startup,Shutdown, Survival Node
AL: A-Line

Note: Redundant nodes are not shown

Fig. 1. Basic Structure of the FTI

4 Fault Tolerance of the FTI

The partitioning of the FTI is motivated by the need to make it fault-tolerant. The A and D nodes are self-checking pairs, since high error detection coverage is essential, while spare C and D nodes can be provided for recovery under M node control. The M node must be continuously available, therefore triplication and voting (TMR) is needed, with spare M nodes added for longer life.

The S3 nodes manage M node replacement and also shut the agent down in the case of failure or global catastrophic events (temporary power loss, heavy radiation, etc.). They are protected by the use of two or more self-checking pairs with backup power. S3 nodes were separated from M nodes to make the node that must survive catastrophic events as small as possible. The S3 nodes also provide outputs to the agent's environment that indicate the health status of the agent: perfect, undergoing protective action or failed.

The all-hardware implementation of the FTI makes it safe from software bugs and external attacks. The one exception is the power management command from C to M nodes (via the D node) which could be used to shut the system down. Special protection is needed here. Hardware design faults in the FTI nodes could be handled by design diversity of self-checking pairs and of M nodes, although the logic of the nodes is very simple and their complete verification should be possible.

When interconnected, the FTI and the original autonomous agent form a computing system that is protected against most causes of system failure. An example system of this type is called DiSTARS: Diversifiable Self Testing And Repairing System and is discussed in detail in [1]. DiSTARS is the first example of an implementation of the immune system paradigm. Much detail of implementation of the FTI is presented in the U.S. patent application disclosure "Self-Testing and – Repairing Fault Tolerance Infrastructure for Computer Systems" by A. Avižienis, filed June 19, 2001.

5. In Conclusion: Some Challenges

The use of the FTI is likely to be affordable for most agents, since the A, M, D, and S3 nodes have a simple internal structure, as shown in [5] and the above mentioned disclosure. It is more interesting to consider that there are some truly challenging missions that can only be justified if their computing systems with the FTI have very high coverage with respect to design faults and to catastrophic transients due to radiation. Furthermore, extensive sparing and efficient power management can also be provided by the FTI. Given that the MTBF of contemporary processor and memory chips is approaching 1000 years, missions that can be contemplated include the 1000-day manned mission to Mars [8] with the dependability of a 10-hour flight of a commercial airliner. Another fascinating possibility is an unmanned very long life interstellar mission using a fault-tolerant relay chain of modest-cost DiSTARS type spacecraft [9]. Both missions are discussed in [5].

References

1. A.Avižienis, J.-C. Laprie, B. Randell, C. Landwehr. Basic concepts and taxonomy of dependable and secure computing. *IEEE Trans. On Dependable and Secure Computing*, 1(1):11-33, January-March 2004.
2. N.A. Lynch. Distributed algorithms. Morgan Kaufmann, 1996.
3. F. Heylighen, C. Gershenson, The meaning of self-organization in computers. *IEEE Intelligent Systems*, July/August 2003, pp. 72-75.
4. A. Avižienis. Toward systematic design of fault-tolerant systems. *Computer*, 30(4):51-58, April 1997.
5. A.Avižienis, A fault tolerance infrastructure for dependable computing with high performance COTS components. *Proc. of the Int. Conference on Dependable Systems and Networks (DSN 2000)*, New York, June 2000, pages 492-500.
6. G.J.V. Nossal. Life, death and the immune system. *Scientific American*, 269(33)52-62, September 1993.
7. S.A. Hofmeyr, S. Forrest. Immunity by design: An artificial immune system. *Proc. 1999 Genetic and Evolutionary Computation Conference*, pages 1289-1296. Morgan-Kaufmann, 1999.
8. Special report: sending astronauts to Mars. *Scientific American*, 282(3):40-63, March 2000.
9. A. Avižienis. The hundred year spacecraft. *Proceedings of the 1st NASA/DoD Workshop on Evolvable Hardware*, Pasadena, CA, July 1999, pp. 233-239.