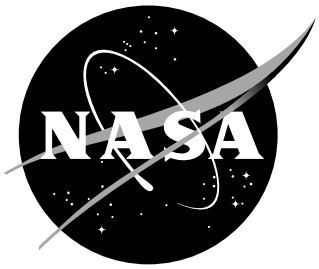
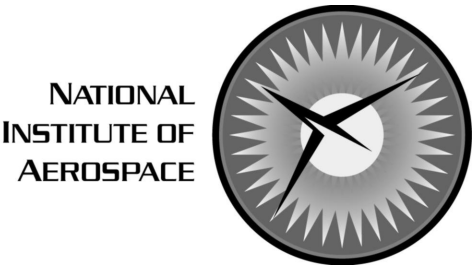
NASA/TM–2004–212432
NIA Report No. 2003–07




# A New On-Line Diagnosis Protocol for the SPIDER Family of Byzantine Fault Tolerant Architectures

*Alfons Geser*
*National Institute of Aerospace, Hampton, Virginia*

*Paul S. Miner*
*Langley Research Center, Hampton, Virginia*

December 2004

The NASA STI Program Office ... in Profile

Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA Scientific and Technical Information (STI) Program Office plays a key part in helping NASA maintain this important role.

The NASA STI Program Office is operated by Langley Research Center, the lead center for NASA's scientific and technical information. The NASA STI Program Office provides access to the NASA STI Database, the largest collection of aeronautical and space science STI in the world. The Program Office is also NASA's institutional mechanism for disseminating the results of its research and development activities. These results are published by NASA in the NASA STI Report Series, which includes the following report types:

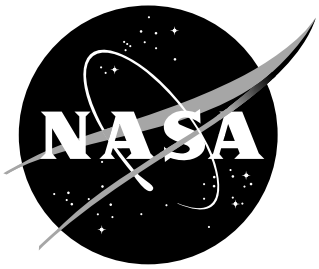
- TECHNICAL PUBLICATION. Reports of completed research or a major significant phase of research that present the results of NASA programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers, but having less stringent limitations on manuscript length and extent of graphic presentations.
- TECHNICAL MEMORANDUM. Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.
- CONTRACTOR REPORT. Scientific and technical findings by NASA-sponsored contractors and grantees.
- CONFERENCE PUBLICATION. Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.
- SPECIAL PUBLICATION. Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.
- TECHNICAL TRANSLATION. English-language translations of foreign scientific and technical material pertinent to NASA's mission.

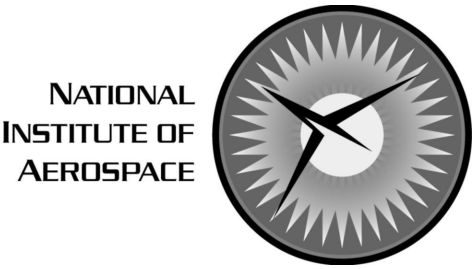
Specialized services that complement the STI Program Office's diverse offerings include creating custom thesauri, building customized databases, organizing and publishing research results ... even providing videos.

For more information about the NASA STI Program Office, see the following:

- Access the NASA STI Program Home Page at ***http://www.sti.nasa.gov***
- E-mail your question via the Internet to help@sti.nasa.gov
- Fax your question to the NASA STI Help Desk at (301) 621–0134
- Phone the NASA STI Help Desk at (301) 621–0390
- Write to:
  NASA STI Help Desk
  NASA Center for AeroSpace Information
  7121 Standard Drive
  Hanover, MD 21076–1320

NASA/TM–2004–212432
NIA Report No. 2003–07




# A New On-Line Diagnosis Protocol for the SPIDER Family of Byzantine Fault Tolerant Architectures

*Alfons Geser*
*National Institute of Aerospace, Hampton, Virginia*

*Paul S. Miner*
*Langley Research Center, Hampton, Virginia*

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23681–2199

December 2004

## Acknowledgment

This work was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-97046 while the first author was in residence at ICASE, NASA Langley Research Center, Hampton, VA 23681-2199, USA.



Available from:

NASA Center for AeroSpace Information (CASI)
7121 Standard Drive
Hanover, MD 21076–1320
(301) 621–0390

National Technical Information Service (NTIS)
5285 Port Royal Road
Springfield, VA 22161–2171
(703) 605–6000

# Abstract

*This paper presents the formal verification of a new protocol for online distributed diagnosis for the SPIDER family of architectures. An instance of the Scalable Processor-Independent Design for Electromagnetic Resilience (SPIDER) architecture consists of a collection of processing elements communicating over a Reliable Optical Bus (ROBUS). The ROBUS is a specialized fault-tolerant device that guarantees Interactive Consistency, Distributed Diagnosis (Group Membership), and Synchronization in the presence of a bounded number of physical faults.*

*Formal verification of the original SPIDER diagnosis protocol provided a detailed understanding that led to the discovery of a significantly more efficient protocol. The original protocol was adapted from the formally verified protocol used in the MAFT architecture. It required $O(N)$ message exchanges per defendant to correctly diagnose failures in a system with $N$ nodes. The new protocol achieves the same diagnostic fidelity, but only requires $O(1)$ exchanges per defendant. This paper presents this new diagnosis protocol and a formal proof of its correctness using PVS.*


# 1 Introduction

The Scalable Processor-Independent Design for Electromagnetic Resilience (SPIDER) is a family of general-purpose fault-tolerant architectures being designed at NASA Langley Research Center to support laboratory investigations into various recovery strategies from transient failures caused by electromagnetic effects. An instance of the SPIDER architecture consists of several Processing Elements (PE) communicating over a Reliable Optical Bus (ROBUS). This logical structure is illustrated in Figure 1. The ROBUS behaves as an ultra-reliable Time Division Multiple

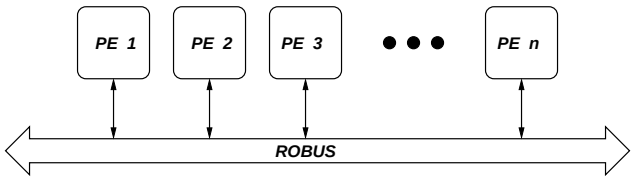



Figure 1: SPIDER architecture

Access (TDMA) broadcast bus. Each PE broadcasts its messages according to a global schedule. This communication model requires that the PEs be synchronized within a known bounded skew. The ROBUS generates periodic synchronization messages to the PEs to maintain this synchrony. The ROBUS provides strong guarantees for communication between the various PEs. The ROBUS ensures that all good PEs that have a good connection will see exactly the same sequence of messages on the bus. The ROBUS also ensures the integrity of the communication schedule. It is impossible for a bad PE to prevent a good PE from correctly broadcasting its

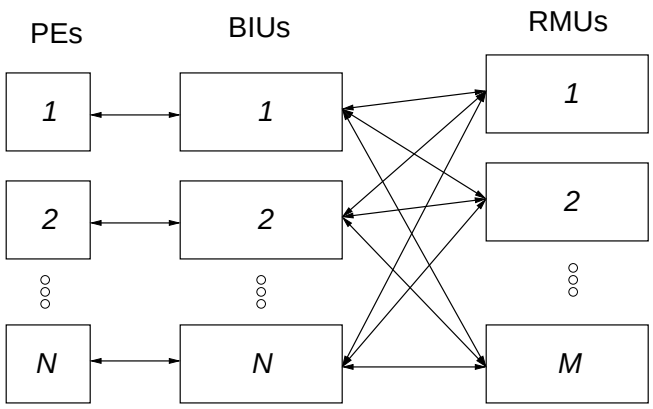



Figure 2: ROBUS topology

messages. Since the ROBUS may suffer internal failures, it periodically reports the health status of its internal components, so that every good PE knows which PEs are able to communicate on the ROBUS.

The ROBUS can only provide these guarantees if enough of its internal Fault Containment Regions (FCR) are good. The topology of the ROBUS, including the attached PEs, is depicted in Figure 2. There are two types of FCRs internal to the ROBUS. The Bus Interface Units (BIU) provide the only interface to the PEs. The Redundancy Management Units (RMU) provide the necessary replication for Byzantine fault tolerance. The RMUs and BIUs may exhibit Byzantine faulty behavior, but the topology of the ROBUS makes that impossible for a PE. The Redundancy Management Units (RMU) provide the necessary replication for Byzantine fault tolerance. The BIUs and RMUs are connected by a complete bipartite graph. There is no direct link between any pair of BIUs or any pair of RMUs.

The SPIDER provides services that are similar to those provided by the Time-Triggered Protocol (TTP) [KG94] and the SAFEbus$^{TM}$ [HD92]. Rushby [Rus03] presents a comparison of safety-critical bus architectures, including SAFEbus, TTP, and SPIDER. One advantage that SPIDER has over these other bus architectures is that it employs a weaker fault assumption. The ROBUS can tolerate several combinations of simultaneous faults, provided its maximum fault assumption is not violated. In contrast, the TTP protocols only guarantee correct operation in the presence of a single undiagnosed fault at a time.

In order to provide the necessary guarantees to the attached PEs, the ROBUS uses three fault-tolerance protocols. Messages between the PEs are sent using an Interactive Consistency (IC) protocol to guarantee agreement. The IC protocol is based on the Draper FTP [Smi84]. Clock synchronization is provided using an adaptation of the Srikanth and Toueg protocol to the ROBUS topology [ST87]. The clock synchronization protocol is also very similar to one presented by Davies and Wakerly [DW78]. The original on-line diagnosis protocol was adapted from MAFT [KWFT88]. The protocol adapted was algorithm DD from Walter, et al. [WLS97]. The conceptual design for the ROBUS is described by Miner, et

al. [MMTP02].

For each of these protocols, we have used the hybrid fault model introduced by Thambidurai and Park [TP88]. This model allows us to tolerate a small number of Byzantine failures and a greater number of benign failures at the same time. The hybrid fault model requires some mechanism to keep track of which sources are eligible to participate in a vote. It requires a local diagnostic mechanism to filter out benign faulty nodes, and a distributed diagnosis protocol to handle more severe failure modes.

We have previously verified both the Interactive Consistency and the original Distributed Diagnosis protocols formally using PVS [GM02]. PVS is a general-purpose theorem proving system developed by SRI International [ORSvH95]. Key aspects of the clock synchronization protocol have also been formally verified using PVS. We have defined a readmission protocol for the ROBUS, but have not yet formally verified it. Readmission requires a slightly different set of fault assumptions for the proofs to complete.

Of the three protocols, the original distributed diagnosis protocol required the most overhead. It involved an interactive consistency exchange originating from each BIU and RMU in the ROBUS. That is, $N + M$ IC exchanges were employed to reach agreement among peers. The new protocol achieves the same effect with less communication than a single IC exchange. Thus the new protocol provides a linear improvement in required bandwidth. In order for the original protocol to ensure agreement, two different sets of eligible voters were required. As a result, additional complexity was introduced to the maximum fault assumption and the resulting proofs. While simplifying the design and formal proof of correctness of the original protocol, we discovered that a much simpler protocol provides the same guarantees. The key observation is that the accusation mechanisms usually provide partial agreement. Furthermore, the new protocol admits a much simpler formal model. For instance we can reuse a basic step, the accusation exchange, together with its properties a few times. The correctness proof of the new protocol in PVS was significantly less complicated than our proof of the original protocol. This allowed us to recognize a few opportunities to strengthen the results.

This report presents this new Diagnosis Protocol, and its formal model in PVS. The PVS model for the new protocol can be found at URL

`http://research.nianet.org/~geser/spider/newdiag.dmp`

The PVS model of the original Diagnosis Protocol can be found at URL

`http://research.nianet.org/~geser/spider/diag.dmp`

The paper is organized as follows: After introducing the basic concepts in Section 2, the two properties are introduced that have to be preserved by the diagnosis: The Maximum Fault Assumption and the Eligible Voters Property (Section 3). An exposition of the new Diagnosis Protocol and the proofs of its properties is given in Section 4. The protocol is extended to handle readmission of previously convicted nodes in Section 5. Finally, a comparison with the original protocol is presented in Section 6.

# 2 Basic Types and Properties

In this section we provide the framework to speak about the SPIDER protocols.

## 2.1 Hybrid Fault Types

Faults may be classified according to the potential consequences they may cause. Our approach is to assume that arbitrary, Byzantine faults may exist, but that they are rare and that less malicious faults come in greater numbers. This is essentially the approach pioneered by Thambidurai and Park [TP88]. We distinguish the following hybrid fault types a node can exhibit:

- A **good** node behaves according to specification.
- A **benign** faulty node only sends messages that are detectably faulty. This includes nodes that have failed silent.
- A **symmetric** faulty node may send arbitrary messages, but each receiver receives the same message.
- An **asymmetric** ("Byzantine") faulty node may send arbitrary messages, including different ones to different receivers.

A node that is not good is called *bad* or *faulty*. The three bad hybrid fault types form a hierarchy in the sense that an asymmetric node has all capabilities of a symmetric node, and a symmetric node may occasionally behave like a benign node.

In view of readmission, we further split good nodes into *trustworthy* nodes and *recovering* nodes. There is no observable difference between the behavior of a recovering node and that of a trustworthy node. The only difference is that we require all trustworthy nodes to be included among the eligible voters held by good nodes. The trustworthy nodes are the good ones that we can count on. Without readmission, all good nodes are trustworthy. With readmission, we may have a node that changes its fault status from *benign* to *good*. The ROBUS needs some time to make the observation that the node is no longer bad. For this purpose we let the node be *recovering* for a specified period before it turns trustworthy. These considerations are not part of the current PVS model.

## 2.2 The Model of Communication

Data received by a node are of type *robus_data*[$T$] where the parameter $T$ denotes the type of data to be communicated. The type *robus_data*[$T$] comprises

- **valid** data of type $T$,
- **receive error**, a token that expresses the fact that an error has been detected during reception,
- **source error**,
- **no majority**.

Note that the absence of an expected message can be detected, so a missing message is modeled by *receive error*. The tokens *source error* and *no majority* are required for the Interactive Consistency protocol [GM02].

Bad behavior of a node is only observable through its communication. This provides us a useful abstraction. We can assign node failure to the communication interface without losing any fidelity. Therefore we may speak about the state of a node as if it were a good node. We may also assume that each node sends correct values, and that there is no error in the receive units. We use this convenient view throughout the presentation. We stipulate that this abstraction is not valid for the reliability analysis.

### 2.3 Node Types and Symmetries

In the ROBUS architecture, BIUs and RMUs are symmetric to each other, up to the detail that BIUs are connected to PEs and RMUs are not. In order to exploit this symmetry, we speak abstractly about two node types, LEFT and RIGHT, where we leave open which are the BIUs and which are the RMUs.

### 2.4 Local Fault Classification

Each good node maintains a view of the health of all nodes. A node *obs* (the *observer*), may classify a node, *def* (the *defendant*), as being

- **trusted**, if *obs* has insufficient evidence to conclude that *def* is faulty;
- **accused**, if *obs* knows that *def* is faulty but is uncertain whether this knowledge is shared by other good observers;
- **declared**, if *obs* knows that *def* is faulty and every good observer of the same kind also declares *def*;
- **convicted**, if *obs* knows that *def* is faulty and every good observer of any kind also convicts *def*.

The local fault classification forms a hierarchy, $trusted < accused < declared$, of increasing knowledge of *obs* about *def*. Sometimes we will need to merge knowledge arriving from two sources; in this case we will get the maximum of the arrived values. For instance, $merge(trusted, declared) = declared$ and $merge(declared, accused) = declared$. The distinction between accused and declared defendants has to be made explicit so as to display the knowledge of agreement. The distinction between declared and convicted defendants is implicit in our model; at the end of the protocol, all declarations are convictions. When we speak about convicted nodes, we usually mean those convicted in the previous frame.

## 3 Invariants of Diagnosis

Diagnosis shall maintain two properties:

1. the presence of enough hardware to assure correct operation (the Maximum Fault Assumption);

2. its own ability to continue (the Eligible Voters Property).

Next we explain what these properties mean, and why we want them preserved.

### 3.1 The Maximum Fault Assumption

The Maximum Fault Assumption (*MFA*) provides an interface between the system reliability assessment and the analysis of the fault-tolerance protocols. The reliability models compute the probability that the *MFA* holds for a specified mission duration. The functional analysis of the protocols is then based on the assumption that the *MFA* holds. In the absence of diagnosis, the *MFA* is independent of the local knowledge of the health status of the system. The SPIDER protocols do not tolerate a simultaneous asymmetric fault of both a BIU and an RMU. Moreover, the protocols require a majority of trustworthy nodes of each kind. The sets of trustworthy, recovering, benign, symmetric, and asymmetric BIUs are denoted by $TB$, $RB$, $BB$, $SB$, $AB$, respectively. The RMUs are similarly denoted by $TR$, $RR$, $BR$, $SR$, $AR$. The Maximum Fault Assumption is defined by:

1. $|TB| > |SB| + |AB|$, and

2. $|TR| > |SR| + |AR|$, and

3. $|AB| = 0$ or $|AR| = 0$.

Call the quadruple $(|TB| - |SB| - |AB|, |TR| - |SR| - |AR|, N - |AB|, M - |AR|)$ the *quality* of the ROBUS. It is obvious that the continuous emergence of faults diminishes the ROBUS quality; finally the *MFA* may become false. The purpose of diagnosis is to counteract this effect, by convicting the faulty nodes at a rate sufficiently faster than the fault arrival rate. The results of the reliability analysis provide a means to determine how fast the diagnosis must be to meet our reliability goal. This is captured in the reliability model described in Miner, et al. [MMTP02]. There is a meta-level argument that diagnosis improves the *MFA*. To model the effect of diagnosis, assume that a node becomes benign when declared. Then, declaring a symmetric or asymmetric node improves ROBUS quality.

The preceding construction cannot be used together with readmission of transiently faulty nodes because a recovering node cannot be represented. Therefore we introduce a variant, the *Dynamic Maximum Fault Assumption*, *DMFA*, that is compatible with readmission. It has an additional parameter, $E_i$, for the set of eligible voters at the node $i$. The *DMFA* is defined by:

1. $|TB| > |SB \cap E_i| + |AB \cap E_i|$ for all trustworthy RMUs $i$, and

2. $|TR| > |SR \cap E_i| + |AR \cap E_i|$ for all trustworthy BIUs $i$, and

3. $|AB \cap E_i| = 0$ for all trustworthy RMUs $i$, or $|AR \cap E_i| = 0$ for all trustworthy BIUs $i$.

The *DMFA* says that a majority of eligible voters of the opposite kind are trustworthy, and that there are no asymmetric eligible voters of both kinds.

### 3.2 The Eligible Voters Property

Each observer maintains its *active sources vector*, an assignment of defendants to fault classifications. Note that information derived from its active sources vector is all that a node broadcasts about the health status of the ROBUS. The following three properties are required for the LEFT active sources vectors:

- **good trusting:** every good node is trusted.
- **symmetric agreement:** any two LEFT observers agree in their assessment of a non-asymmetric defendant.
- **declaration agreement:** the set of declared nodes is the same for any two LEFT observers.

The LEFT nodes are said to satisfy *Correct Active Sources*, $CAS_L$, whenever the above three properties are satisfied.

The set of eligible voters, $E_i$, considered by node $i$ consists of those opposite nodes that $i$ trusts. $E_i$ is derived from information in the active sources vector of node $i$. We say that LEFT nodes satisfy the *Eligible Voters Property*, $EVP_L$, if all good RIGHT nodes are in $E_i$ for all LEFT $i$, and $E_i$ and $E_j$ satisfy symmetric agreement for all LEFT $i, j$. An accusation is called *admissible* if only bad nodes are accused and an accusation against a non-asymmetric defendant is shared by all peers. Merging an admissible accusation into a LEFT active sources vector preserves $CAS_L$. Also $CAS_L$ implies $EVP_L$. Similar statements hold for the RIGHT observers and their properties $CAS_R$ and $EVP_R$.

The conditions good trusting and declaration agreement are clearly motivated by our definitions of accused and declared. We will demonstrate in Example 5 (Section 4.1) that symmetric agreement is an essential premise to correctness for the new diagnosis protocol. In [GM02], we presented an example illustrating why it is essential for the Interactive Consistency protocol. Example 4 (Section 3.3 shows that it is possible to make an inadmissible accusation against a bad node.

### 3.3 Interactive Consistency Protocol

A message broadcast protocol is called reliable if it satisfies the following two properties:

- **validity:** every good node receives the value sent by a good node.
- **agreement:** all good nodes agree in the value sent.

For reliable message passing in the presence of various faults, we use the Interactive Consistency Protocol. It is presented here to illustrate the *Eligible Voters Property* and certain accusation mechanisms. We have previously presented this protocol with a formal proof of its correctness [GM02]. The Interactive Consistency Protocol works as follows:

1. A single BIU, $g$, as per agreed schedule, broadcasts some value, $valid(v)$, to all RMUs.

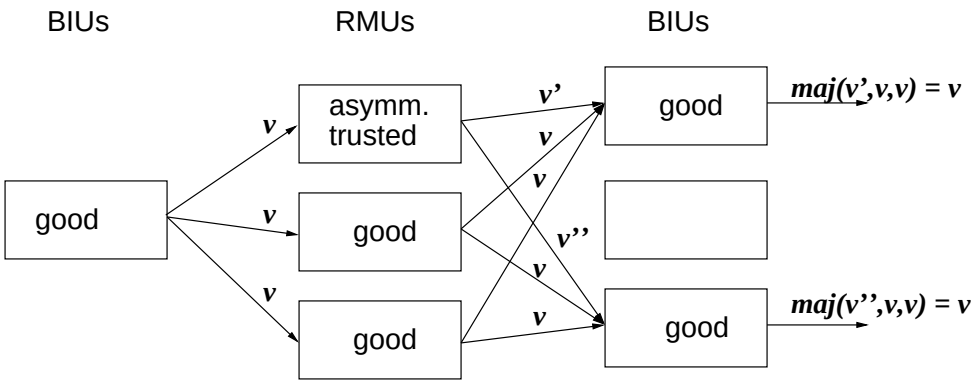



Figure 3: Interactive Consistency with a good source (Example 1).

2. Every RMU relays its received value, $d$, to all BIUs. However, if $d = receive\_error$ then it sends *source_error* to redirect the blame from itself to the originator of the message.

3. Every BIU, $p$, collects the vector of values it received (one value per RMU node). Then it updates the set of eligible voters, $E_p$. Each RMU from which $p$ receives *receive_error* is removed from the set of eligible voters, $E_p$, for the vote in step 4. It may also be accused by $p$.

4. Else if $p$ receives some value $d_{maj}$ from a majority of eligible RMUs then it determines $d_{maj}$ and, if $d_{maj}$ is non-valid, declares $g$. Otherwise $p$ determines *no_majority* and declares $g$.

5. If BIU $g$ was previously convicted by $p$ (not including the recent declarations in step 4) then $p$ forwards *source_error* to its PE. Otherwise $p$ forwards the value determined in step 4 to its PE.

The remainder of this section consists of examples involving the IC protocol. Example 1 describes the behavior of this protocol when the originating BIU is good. Example 2 illustrates the behavior of this protocol when the source BIU is asymmetrically faulty. Example 3 demonstrates that the symmetric agreement property is essential for correct operation of the IC protocol. Finally, example 4 illustrates the possibility of an inadmissible accusation against a symmetric faulty node.

**Example 1.** *Figure 3 illustrates the behavior of the IC protocol when the originating BIU is good. Let $N = M = 3$. Let BIU 2 be the originating node. Since it is good, it sends the same value $v$ to all RMUs. Suppose that RMU 1 is asymmetrically faulty, but is trusted by all good BIUs. It may send $v'$ to BIU 1 and $v''$ to BIU 3. Since RMUs 2 and 3 are good, they correctly relay the value $v$ to the BIUs. Since there is a majority of eligible trustworthy RMUs, all BIUs correctly select $v$ as the value sent by BIU 2. Thus, both agreement and validity are satisfied in this case.*

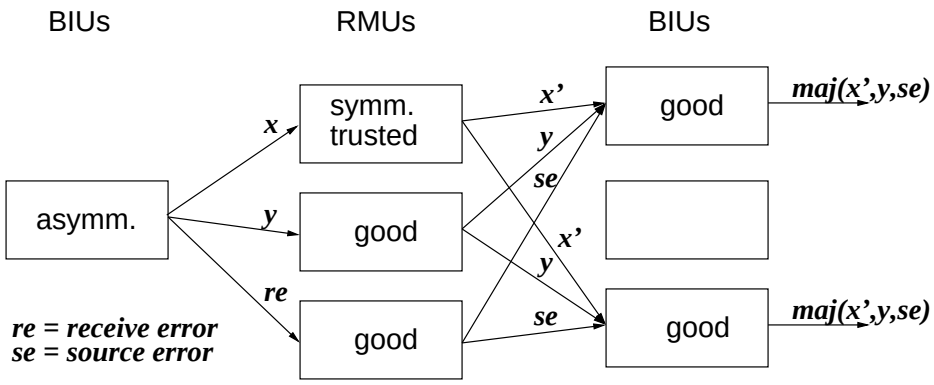



Figure 4: Interactive Consistency with an asymmetric faulty source (Example 2).

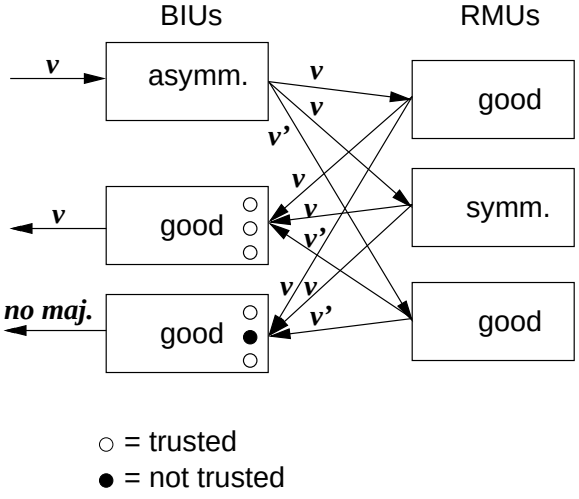



Figure 5: Symmetric agreement is essential (Example 3).

**Example 2.** *Figure 4 illustrates the behavior of the IC protocol when the originating BIU is asymmetrically faulty. Let $N = M = 3$. Let BIU 2 be the originating node. Since it is asymmetrically faulty, it may send different values to each RMU. In this case, it sends $x$ to symmetrically faulty RMU 1, $y$ to good RMU 2, and is detectably faulty to RMU 3. Since RMU 1 is symmetrically faulty, it sends the same value $x'$ to all BIUs. Good RMU 2 forwards $y$ to all BIUs and good RMU 3 sends source_error to all BIUs. Since all BIUs have received the same data vector from the RMUs, $(x', y, source\_error)$, they will all determine the same value for BIU 2.*

**Example 3.** *Figure 5 illustrates the necessity of the symmetric agreement property. Let $N = M = 3$. Suppose that BIU 1 is asymmetric and the static Maximum Fault Assumption holds. Now let BIU 1 send valid$(v)$ to the good RMU 1 and to the symmetric RMU 2, and a different valid$(v')$ to the good RMU 3. Let BIU 2 trust all RMUs, and let BIU 3 trust RMUs 1 and 3 only. So symmetric agreement does not*

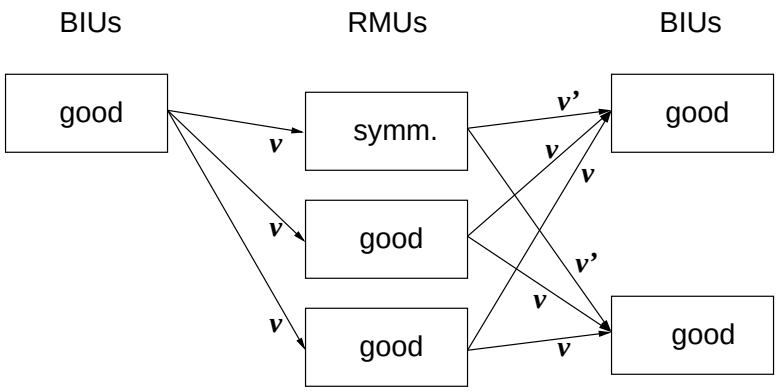



Figure 6: An inadmissible accusation against a bad node (Example 4).

*hold. Then BIU 2 finds a majority value, valid(v), whereas BIU 3 finds no majority. So agreement is violated.*

**Example 4.** *Figure 6 demonstrates a scenario which violates the symmetric agreement clause of admissible accusations. Let $N = M = 3$. Suppose that during an Interactive Consistency exchange the good general BIU 1 broadcasts its value, v, to the RMUs. The good RMUs 2 and 3 forward v to the BIUs. The symmetric RMU 1 forwards a different value, $v'$, to the BIUs. Now BIU 1, who knows which value it had transmitted, can conclude that RMU 1 is bad. Its peer, BIU 3, cannot join this conclusion because it cannot pinpoint which of BIU 1 or RMU 1 is responsible. Symmetric agreement therefore does not hold; BIU 1's accusation is not admissible.*

# 4 An Overview of the New Protocol

Diagnosis can be decomposed into two separate operations. Locally, each node monitors the communication from other nodes and gathers evidence that may form the basis of an accusation. Then the locally generated syndromes are exchanged to reach a consistent global view of the system state.

The goal of diagnosis is to ensure the following properties:

**Correctness** No good node is ever convicted

**Completeness** All faulty nodes are eventually convicted

**Conviction Agreement** All good nodes agree on convictions

In the presence of arbitrary asymmetric failures, it is impossible to guarantee both correctness and completeness [SR87]. For the ROBUS, we have striven to convict as many faulty nodes as possible, while preserving correctness.

The protocols described below apply to a single defendant. It is understood that diagnosis of all defendants may be done in parallel, bandwidth permitting.

### 4.1 Accusation Mechanisms

During all protocols each node records evidence of faulty behavior of other nodes. Some of this evidence may lead to an accusation of a node. Some evidence may even lead to a *declaration* if it is known that agreement holds. There is a design constraint that every accusation mechanism be *admissible*.

We distinguish between direct and indirect observations that lead to accusations. A *direct observation* is a single event that leads to an accusation of the sender. Direct observations include:

- No message was received during the reception window;
- An improperly formatted message is received.

These observations take place during normal operation of all protocols. "Improperly formatted" may also mean that an encoded message does not pass a parity check. The effect of a direct observation is modeled by the token *receive error*.

Indirect observations are a collection of events that only together provide the basis of an accusation. These only occur in the context of some protocol. Disagreement with voted results is the primary mechanism. Below, we enumerate mechanisms for forming indirect accusations and declarations in the context of an Interactive Consistency. Let the diagnosing node be a LEFT node.

1. If a majority of eligible RIGHT nodes offer evidence against a LEFT node; that LEFT node is accused.
2. If, in a consistent source exchange from RIGHT to LEFT (where all good RIGHT are known to agree), a RIGHT node disagrees with the majority; this node is accused.
3. If, in an Interactive Consistency exchange from the LEFT, there is no majority; then the originating node is accused. Furthermore, this indicates that the originating LEFT must be asymmetrically faulty.
4. If a RIGHT relay node disagrees with the majority during Interactive Consistency exchanges from a majority of eligible LEFT sources, this relay is accused.

Both direct and indirect observations result in admissible accusations. All direct observations satisfy symmetric agreement – If the observed node is not asymmetric, then the observers receive the same value, so they make the same observation. By the agreement property, an accusation of the Form 3 is shared by all LEFT observers, so they may issue a declaration. Likewise, an accusation of Form 1 can be turned into a declaration – If the LEFT defendant *def* is asymmetric, then there are no asymmetric RIGHT nodes, so every LEFT observer gets the same data vector. Otherwise, all RIGHT nodes get the same data, so all LEFT observers will arrive at the same voted value. As a typical case, we may declare the general of an Interactive Consistency exchange on the basis that a majority of trusted RIGHT nodes sent source error. Accusations of Forms 2 and 4 also satisfy symmetric agreement – If

there are asymmetric LEFT nodes then no RIGHT node is asymmetric, so each LEFT observer gets the same vector. Otherwise all RIGHT nodes have received the same data. Each LEFT node will arrive at the same majority value. If the RIGHT defendant is not asymmetric then all LEFT will agree that it did not match the majority value.

All of these observation mechanisms are in place in the current SPIDER prototype.

### 4.2 The Accusation Exchange

The accusation exchange is the primitive operation that we use to build our new diagnosis protocol. If a node trusts the defendant *def*, it broadcasts the accusation *working*, otherwise it broadcasts *failed*. Sometimes a node may transmit an accusation which is derived from information other than its trust in the defendant.

The accusation exchange works as follows:

1. Every LEFT node, $l$, as per schedule, transmits its accusation against the defendant *def* to all RIGHT nodes. We denote the value sent by $l$ as $v_l$.

2. Every RIGHT node, $r$, collects the vector of values it received into vote vector $V_r(def)$ (one value per LEFT node).

3. Each RIGHT node, $r$ removes any source that did not send a valid message from its set of eligible voters, $E_r$.

4. For each RIGHT node $r$, if a majority of eligible voters vote for *working*, then the verdict against *def* is *working*. Otherwise, the verdict is *failed*. We denote the result of this step by $vote(E_r, V_r(def))$.

Useful properties of the accusation exchange are summarized in the following three theorems:

**Theorem 4.1 (Validity).** *For good RIGHT $r$, if there is a majority of good LEFTs in the set of eligible voters, $E_r$, then there is a good LEFT $l$ such that*

$$vote(E_r, V_r(def)) = v_l$$

This theorem says that the result of the accusation exchange is consistent with the accusation of some good source. If the result of the vote is *working*, then a majority of the eligible voters voted *working*. Since there is also a majority of good sources, the pigeonhole principle ensures that at least one good source sent *working*. If the result is *failed*, then at least half of the eligible voters voted *failed*. Again, the pigeonhole principle ensures that at least one good source voted *failed*. It is worth noting that this result is only valid when voting a binary type.

**Theorem 4.2 (Agreement Propagation).** *For good RIGHTs $r1, r2$, if there is a majority of good LEFT nodes in both $E_{r1}$ and $E_{r2}$, and all good LEFTs have the same accusation against def, then*

$$vote(E_{r1}, V_{r1}(def)) = vote(E_{r2}, V_{r2}(def))$$

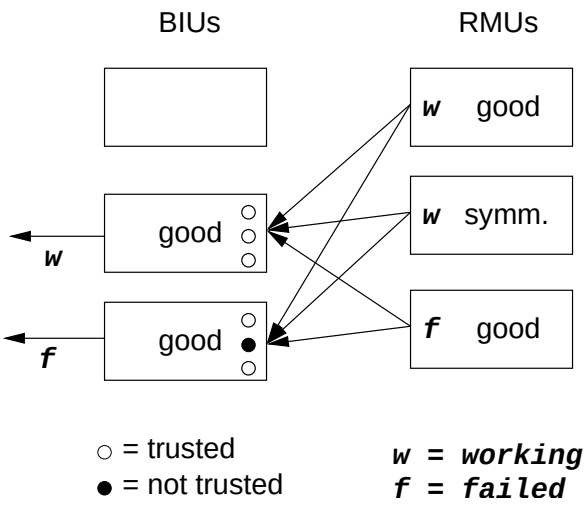



Figure 7: The Symmetric Agreement premise is essential for agreement generation (see Example 5)

This follows as a corollary of Theorem 4.1. We know that the result of the accusation exchange matches the accusation sent by at least one good LEFT. Since all good LEFTs agree concerning their accusation against *def*, all good RIGHTs must agree concerning the verdict against *def*.

**Theorem 4.3 (Agreement Generation).** *For good RIGHTs* $r1, r2$*, if there are no asymmetric LEFT nodes in* $E_{r1}$*, and* $E_{r1} = E_{r2}$*, then*

$$vote(E_{r1}, V_{r1}(def)) = vote(E_{r2}, V_{r2}(def))$$

Since the vote ignores sources that are not in $E_{r1}$, and the vote vectors agree for nodes in $E_{r1}$, the votes must compute the same result.

If we know that the hypotheses of Theorem 4.2 or Theorem 4.3 are satisfied, then the results of the exchange agree and may become declarations.

The validity property provides a mechanism to prove correctness. Since we require that all accusations be admissible, it is impossible for a good node to accuse another good node. The only way for the accusation exchange to return a verdict of *failed* is if there is some good source that votes *failed* for *def*. Agreement propagation and agreement generation give us tools to establish various agreement properties.

Now that we have defined the accusation exchange protocol, we can demonstrate why the symmetric agreement premise is essential for the agreement generation property.

**Example 5.** *Let* $N = M = 3$*. Let the good RMU 3 accuse some (asymmetric) defendant, and let the good RMU 1 and the symmetric RMU 2 defend the defendant (Figure 7). Let BIU 2 trust all RMUs, and let BIU 3 trust RMUs 1 and 3 only. So symmetric agreement does not hold. Then BIU 2 finds a majority in favor of working whereas BIU 3 finds no majority, so its verdict is failed. So agreement is violated.*

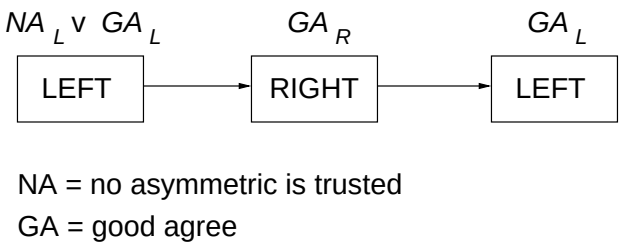



Figure 8: The two-stage diagnosis protocol

### 4.3 Reaching Convictions

For the remainder, we will assume that the defendant, *def*, is a RIGHT node. We wish to set up a pattern of exchange that will ensure correctness and conviction agreement. Initially we are concerned with diagnosing a node, *def*, that has not previously been convicted. Any previously convicted node cannot be in the set of eligible voters for any good node.

The new Diagnosis Protocol operates as follows:

1. The LEFTs broadcast their accusation against *def* using the accusation exchange protocol.

2. The RIGHTs convert the verdict into a declaration and merge this result with any existing local declaration against *def*.

3. If *def* is declared, then the RIGHTs record that *def* is convicted and then broadcast *failed* to the LEFTs using the accusation exchange protocol (from right to left). Otherwise, the RIGHTs broadcast *working*.

4. The LEFTs convert the resulting verdict against *def* into a declaration and use this result to set the new conviction for *def*.

The accusation exchange protocol needs one message exchange where it uses all communication channels in parallel. So the new Diagnosis Protocol needs 2 message exchanges per defendant. Assuming sufficient bandwidth, all defendants may be processed in parallel, and the new Diagnosis Protocol needs only 2 message exchanges in total.

The stages and properties of this protocol are sketched in Figure 8.

**Theorem 4.4 (Correctness).** *If DMFA, EVP, all accusations are admissible, and def is convicted, then def is faulty.*

This result follows from Theorem 4.1. If *def* is convicted, then it must have been declared by a good RIGHT in step 2. It may have been previously declared using one of the indirect mechanisms described in Section 4.1. Or it may have been declared via the accusation exchange in step 1 of the protocol. In either case, the verdict implies that at least one good node accused *def*. By admissibility, a good node cannot accuse a good node, hence *def* must be faulty.

**Theorem 4.5 (Conviction Agreement).** *If DMFA, EVP, and all accusations are admissible, then either all good nodes convict def or no good node convicts def.*

The stages of this proof are illustrated in Figure 8. By clause 3 of *DMFA*, either there are no asymmetric eligible LEFTs or no asymmetric eligible RIGHTs. If there are no asymmetric eligible LEFTs, then by the symmetric agreement clause of *EVP*, the hypothesis of Theorem 4.3 holds, and we have agreement by all good RIGHTs after step 1. Otherwise, there are no asymmetric eligible RIGHTs. Since all accusations are admissible, all good LEFTs agree concerning *def*. The *DMFA* and good trusting clause of *EVP* ensure that a majority of the eligible LEFTs are good, so by Theorem 4.2 we have agreement among all good RIGHTs after step 1. In step 2, the merge preserves declaration agreement. Thus the good RIGHTs always agree concerning *def* prior to step 3. The *DMFA* and good trusting clause of *EVP* ensure that a majority of eligible RIGHTs are good, so Theorem 4.2 guarantees that all good LEFTs will agree. Finally, Theorem 4.1 ensures that the verdict returned after step 3 agrees with some good RIGHT. Thus all good nodes agree on the verdict.

We know from the impossibility result of Shin and Ramanathan [SR87], that we cannot guarantee completeness in the presence of arbitrary asymmetric faults. The merging of declarations in step 2 ensures that the convictions include all existing declarations. We have shown in PVS that:

- all benign faulty nodes are convicted,
- all accused symmetric faulty nodes are convicted, and
- whenever defendant *def* is accused by all members of a subset of good LEFT nodes, $GL' \subseteq GL$, such that for all $i \in GR$, $|GL'| \geq |E_i|/2$, then *def* is convicted.

In other words, any node that is not trusted by a majority of eligible voters is convicted.

# 5 Diagnosis of Previously Convicted Nodes

The previous sections provide strong results concerning the conviction of previously trusted nodes. In order to provide a readmission facility, we also need a mechanism for restoring trust in previously convicted nodes. The readmission procedure requires that a convicted node behave correctly for a brief period before being readmitted to the set of trusted nodes. We would like to use the same diagnosis protocol as above to restore trust. Unfortunately, fresh accusations against a previously convicted node are not guaranteed to satisfy the hypotheses of Theorems 4.2 or 4.3. Section 5.1 presents two such scenarios. In Section 5.2, we present a three-stage exchange that works even in the case of previously convicted defendants.

## 5.1 Counterexamples

The diagnosis protocol introduced in the previous section may fail if the previously convicted defendant is an actively misbehaving asymmetric faulty RIGHT node.

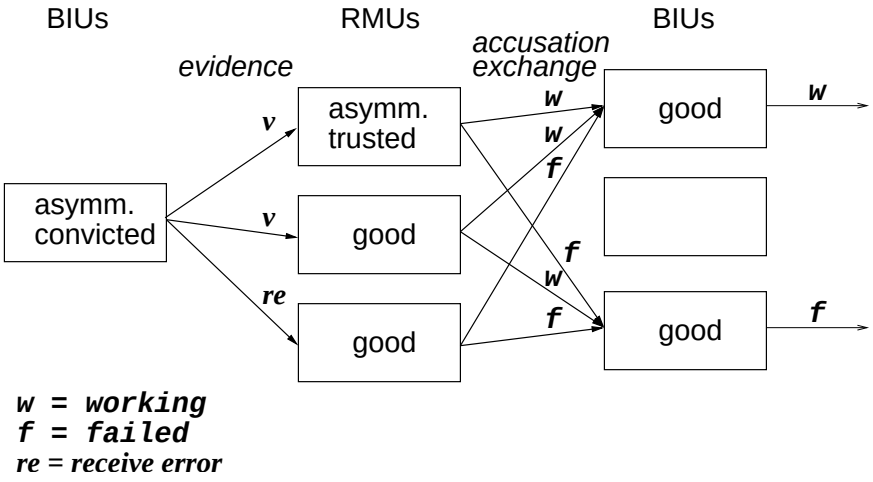



Figure 9: One accusation exchange stage may not provide agreement if the defendant is previously convicted (see Example 6)

By the Dynamic Maximum Fault Assumption, there may now exist a trusted asymmetric faulty LEFT node. Example 6 illustrates that a single accusation exchange does not provide agreement. Example 7 illustrates that indirect observations of a previously convicted peer need not agree.

**Example 6.** *Let $N = M = 3$. Suppose that BIU 2 is asymmetric and previously convicted, and that RMU 1 is asymmetric and trusted, and that all other nodes are good (Figure 9). Let BIU 2 send the correct value to RMUs 1 and 2, and source error to RMU 3. This provides RMU 3 with the basis of an accusation against BIU 2. In the following accusation exchange round, let RMU 1 transmit working to BIU 1, but failed to BIU 3. Thus BIU 3 finds BIU 2 guilty, but BIU 1 does not; there is no agreement.*

**Example 7.** *Let $N = M = 3$. Suppose that BIU 2 is asymmetric and previously convicted, and that RMU 1 is asymmetric and trusted, and that all other nodes are good (Figure 10). Let BIU 2 send the correct value, $v$, to RMUs 1 and 2, and source error to RMU 3. Let RMU 2 relay $v$, let RMU 3 relay source error to indicate that it did not receive valid data, and let RMU 1 send $v$ to BIU 1 and another value, $v'$, to BIU 3. Thus BIU 3 sees no majority and concludes that BIU 2 is guilty, but BIU 1 has no such evidence; there is no agreement.*

### 5.2 Extended Diagnosis Protocol

The problem can be remedied by a slight modification to the simple protocol presented in Section 4.3. The protocol for diagnosing a previously convicted RIGHT *def* is:

1. The LEFTs broadcast their accusation against *def* using the accusation exchange protocol.

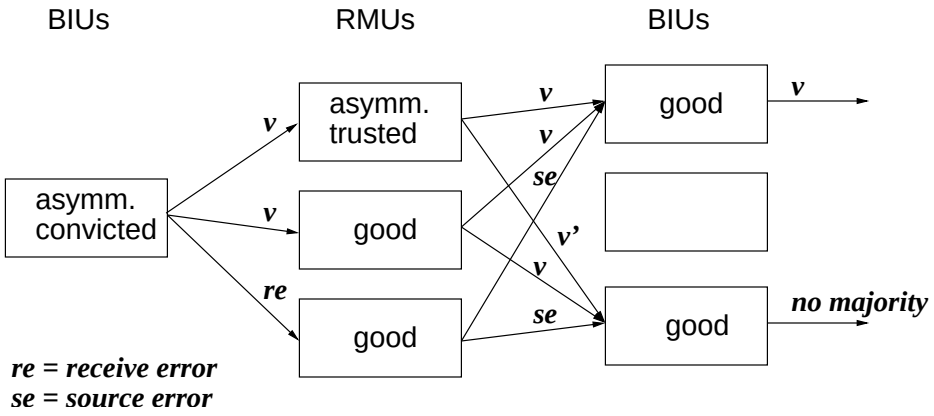



Figure 10: Indirect observation may not provide agreement if the defendant is previously convicted (see Example 7)

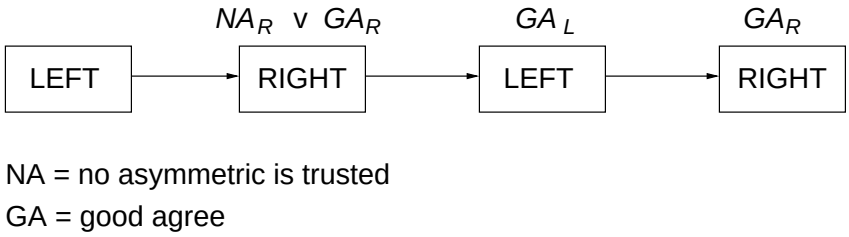



Figure 11: The three-stage diagnosis protocol

2. The RIGHTs convert the verdict into an accusation (since we might not have agreement) and merge this result with any existing local accusation against *def*.

3. The RIGHTs broadcast their merged accusation against *def* to the LEFTs using the accusation exchange protocol.

4. The LEFTs convert the resulting verdict against *def* into a declaration and use this result to set the new conviction for *def*.

5. The LEFTs broadcast the verdict from the previous step to the RIGHTs using the accusation exchange protocol. The RIGHTs use the resulting verdict to determine whether *def* is convicted.

The stages and properties of this extended protocol are illustrated in Figure 11.

The arguments for correctness and conviction agreement are similar to those for the simple protocol. The main difference is that prior to the first exchange, we cannot guarantee $NA_L \vee GA_L$. Therefore, we cannot guarantee agreement after the first accusation exchange. The properties of the extended protocol are the same as for the simple protocol, except that they occur one stage later.

**Theorem 5.1 (Correctness and Conviction Agreement).** *Let DMFA and CAS hold and let def be a RIGHT node. Let all local accusations by RIGHTs against def satisfy* $NA_L \Rightarrow GA_R$. *Then after the three-stage exchange, CAS holds and either all nodes or no nodes convict def.*

Correctness follows from *CAS* by three applications of Theorem 4.1.

To show Conviction Agreement, we begin by showing that after step 1, we have $NA_R \vee GA_R$. Assume $\neg NA_R$. Then by the *DMFA* we have $NA_L$. By Theorem 4.3, we get $GA_R$. In step 2, the merge preserves $NA_R \vee GA_R$. After step 3 we get $GA_L$ either from $GA_R$ by Theorem 4.2 or from $NA_R$ by Theorem 4.3. After step 5 we get $GA_R$ from $GA_L$ by Theorem 4.2. Theorem 4.1 ensures that the convictions coincide after the last step.

This protocol gives us a means to readmit nodes that have been upset by some transient disturbance, but have since restored correct state. However, this protocol also makes it possible to readmit a node that is still faulty. Readmission of a faulty node can immediately violate the *DMFA*. It is necessary to perform some reliability analysis to determine how frequently we should consider nodes for readmission.

# 6 Comparison to the Old Protocol

In this section we present our original diagnosis protocol and highlight the performance improvements of the new protocol.

## 6.1 Old Diagnosis Protocol

The old Diagnosis Protocol works as follows for each defendant *def*:

1. BIUs reliably exchange their accusations against *def*. If a majority of undeclared BIUs accuse *def* then *def* is declared.

2. RMUs reliably exchange their accusations against *def*. If a majority of undeclared RMUs accuse *def* then *def* is declared.

3. BIUs broadcast their declarations to the RMUs; the RMUs merge these with their declarations.

4. RMUs broadcast their declarations to the BIUs; the BIUs merge these with their declarations.

We exploited the obvious symmetry by providing an *accusation exchange* protocol and a *declaration exchange* protocol, each parameterized with LEFT and RIGHT.

The old accusation exchange protocol works as follows:

1. each LEFT node, $G$, uses the Interactive Consistency Protocol to broadcast, to all LEFTs, its accusation against *def*.

2. after step 1, each LEFT node, $p$, has collected a vector of accusations against *def* (one value from each LEFT).

3. LEFT nodes from which LEFT $p$ received *source_error* are declared by $p$.

4. If a majority of the remaining undeclared LEFT nodes accuse *def* then *def* is declared.

The declaration exchange protocol works as follows:

1. each RIGHT node, $r$, broadcasts its verdict (either working or failed) against *def* to all LEFT nodes.

2. each LEFT node, $p$, removes any RIGHT for which it records *receive_error* from its set of eligible voters, $E_p$.

3. each LEFT node, $p$, determines whether *def* has been declared faulty by performing a majority vote of the received eligible verdicts against *def*.

4. each LEFT node, $p$, merges the received declaration into its own active sources vector.

In our earlier paper [GM02], steps 1 to 3 of the declaration exchange were presented as an instance of a consistent source exchange. The properties of the consistent source exchange for diagnostic data have been subsumed by the new accusation exchange protocol presented in this paper. As with the new protocol, the defendants may be processed in parallel by packing the accusations against all defendants into a single vector.

### 6.2 Performance Comparison

The principal difference between the two protocols is in the accusation exchange. The original protocol used $N + M$ IC exchanges for each defendant, one exchange for each BIU and RMU in the system. The new accusation exchange requires only a single instance of the new accusation exchange described in Section 4.2. This new exchange corresponds to steps 2 to 4 of the IC protocol described in Section 3.3. Thus, there is a linear improvement in performance of the new protocol.

## 7 Discussion

The new SPIDER Diagnosis Protocol presented in this report offers the same correctness, completeness, and agreement properties as the original protocol. However, the new protocol exploits stronger assumptions about the underlying accusations. Theorem 4.5 depends on the fact that good nodes agree with respect to non-asymmetric defendants. The corresponding result in [GM02] does not require this agreement property.

The extension for readmission of previously convicted nodes presented in Section 5 suffices to overcome this difference. The additional accusation exchange ensures conviction agreement, even when the symmetric agreement clause of admissible accusations is violated. This extended protocol still provides a linear performance improvement over our original protocol.

The new protocol is also simpler than the original protocol. For the original protocol, we used two different sets of eligible voters. One was employed during Interactive Consistency exchanges. The other set of eligible voters was used to process the exchanged accusations. This latter set required stronger agreement properties [GM02]. The new protocol only employs the set of eligible voters required by the Interactive Consistency

Finally, the formal proof of the new protocol is much simpler than the original protocol. The proof of the original protocol required properties of three distinct message exchange primitives. In the new protocol, all message exchanges are instances of the accusation exchange described in Section 4.2. This provides an economy of effort in our proofs. We were able to formally prove the properties of the accusation exchange mechanism once and then re-use the results where needed.

# Conclusion

We have designed a new Diagnosis Protocol for the SPIDER family of Byzantine fault-tolerant architectures. The new protocol is much simpler and more efficient by a linear factor than the original one. Further, the new protocol admits a simpler proof of correctness.

The new Diagnosis Protocol preserves the Dynamic Maximum Fault Assumption and the Eligible Voters Property. It achieves agreement of convictions. It guarantees conviction of:

- all benign faulty nodes,
- all symmetric nodes that are accused by any good node, and
- any asymmetric faulty node that is accused by enough good nodes.

We have shown how to extend this protocol to allow for diagnosis of previously convicted nodes seeking readmission into the sets of eligible voters. This raises the difficult question of how frequently we should consider a node for readmission. It is possible for a failed node to behave well enough to qualify for readmission. Thus, the extended protocol is not guaranteed preserve the Dynamic Maximum Fault Assumption. We will have to appeal to failure modes analysis and reliability analysis to determine how often we should consider convicted nodes for readmission.

In some situations, it is not necessary to readmit previously convicted nodes. The simple protocol is appropriate for these situations. When there is a need for transient recovery, all diagnosis may be based upon the extended protocol, with little additional overhead.

The results of this research have also provided better understanding of the other SPIDER protocols. We have adapted the lessons learned to establish simplified arguments for the correctness of all SPIDER protocols. The principal benefit is a cleaner structure for the formal proofs. We are currently using this new structure to re-verify the SPIDER clock synchronization protocol in PVS.

## References


[DW78] Daniel Davies and John F. Wakerly. Synchronization and matching in redundant systems. *IEEE Transactions on Computers*, 27(6):531–539, June 1978.

[GM02] Alfons Geser and Paul S. Miner. A formal correctness proof of the SPIDER Diagnosis Protocol. In *Theorem Proving in Higher Order Logics*, Track B Proceedings, pages 71–86, August 2002. NASA Technical Report NASA/CP-2002-211736.

[HD92] Kenneth Hoyme and Kevin Driscoll. SAFEbus$^{TM}$. In *11th AIAA/IEEE Digital Avionics Systems Conference*, pages 68–73, Seattle, WA, October 1992.

[KG94] Hermann Kopetz and Günter Grünsteidl. TTP – a protocol for fault-tolerant real-time systems. *IEEE Computer*, 27(1):14–23, January 1994.

[KWFT88] R. M. Kieckhafer, C. J. Walter, A. M. Finn, and P. M. Thambidurai. The MAFT architecture for distributed fault tolerance. *IEEE Transactions on Computers*, 37(4):398–405, April 1988.

[MMTP02] Paul S. Miner, Mahyar Malekpour, and Wilfredo Torres-Pomales. Conceptual design of a Reliable Optical BUS (ROBUS). In *21st AIAA/IEEE Digital Avionics Systems Conference DASC*, Irvine, CA, October 2002.

[ORSvH95] Sam Owre, John Rushby, Natarajan Shankar, and Friedrich von Henke. Formal verification for fault-tolerant architectures: Prolegomena to the design of PVS. *IEEE Transactions on Software Engineering*, 21(2):107–125, February 1995.

[Rus03] John Rushby. A comparison of bus architectures for safety-critical embedded systems. Technical Report NASA/CR-2003-212161, NASA Langley Research Center, Hampton, VA, March 2003.

[Smi84] T. Basil Smith. Fault tolerant processor concepts and operations. In *Fault Tolerant Computing Symposium 14*, pages 158–163. IEEE Computer Society, 1984.

[SR87] K. Shin and P. Ramanathan. Diagnosis of processors with Byzantine faults in a distributed computing system. In *17th Fault Tolerant Computing Symposium (FTCS 17)*, pages 55–60, 1987.

[ST87] T. K. Srikanth and Sam Toueg. Optimal clock synchronization. *Journal of the ACM*, 34(3):626–645, July 1987.

[TP88] Philip Thambidurai and You-Keun Park. Interactive consistency with multiple failure modes. In *7th Reliable Distributed Systems Symposium*, pages 93–100, October 1988.

[WLS97] Chris J. Walter, Patrick Lincoln, and Neeraj Suri. Formally verified on-line diagnosis. *IEEE Transactions on Software Engineering*, 23(11):684–721, November 1997.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*



| 1. REPORT DATE *(DD-MM-YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From - To)* |
|---|---|---|
| 01- 12 - 2004 | Technical Memorandum | |

**4. TITLE AND SUBTITLE**

A New On-Line Diagnosis Protocol for the SPIDER Family of Byzantine Fault Tolerant Architectures

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**

Geser, Alfons; and Miner, Paul S.

**5d. PROJECT NUMBER**

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

762-60-21-02

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

NASA Langley Research Center
Hampton, VA 23681-2199

**8. PERFORMING ORGANIZATION REPORT NUMBER**

L-18306

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

National Aeronautics and Space Administration
Washington, DC 20546-0001

**10. SPONSOR/MONITOR'S ACRONYM(S)**

NASA

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

NASA/TM-2004-212432

**12. DISTRIBUTION/AVAILABILITY STATEMENT**

Unclassified - Unlimited
Subject Category 62
Availability: NASA CASI (301) 621-0390 Distribution: Nonstandard

**13. SUPPLEMENTARY NOTES**

Alfons Geser is with the National Institute of Aerospace, Hampton, VA; Paul S. Miner is with NASA Langley Research Center, Hampton, VA
An electronic version can be found at http://techreports.larc.nasa.gov/ltrs/ or http://ntrs.nasa.gov

**14. ABSTRACT**

This paper presents the formal verification of a new protocol for on-line distributed diagnosis for the SPIDER family of architectures. An instance of the Scalable Processor-Independent Design for Electromagnetic Resilience (SPIDER) architecture consists of a collection of processing elements communicating over a Reliable Optical Bus (ROBUS). The ROBUS is a specialized fault-tolerant device that guarantees Interactive Consistency, Distributed Diagnosis (Group Membership), and Synchronization in the presence of a bounded number of physical faults. Formal verification of the original SPIDER diagnosis protocol provided a detailed understanding that led to the discovery of a significantly more efficient protocol. The original protocol was adapted from the formally verified protocol used in the MAFT architecture. It required $O(N)$ message exchanges per defendant to correctly diagnose failures in a system with $N$ nodes. The new protocol achieves the same diagnostic fidelity, but only requires $O(1)$ exchanges per defendant. This paper presents this new diagnosis protocol and a formal proof of its correctness using PVS.

**15. SUBJECT TERMS**

Fault Tolerance, SPIDER, Byzantine, Reliability, Diagnosis, Interactive Consistency, Group Membership

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | STI Help Desk (email: help@sti.nasa.gov) |
| | | | | | 19b. TELEPHONE NUMBER *(Include area code)* |
| U | U | U | UU | 27 | (301) 621-0390 |