CARNEGIE MELLON UNIVERSITY

CARNEGIE INSTITUTE OF TECHNOLOGY

Reliability Validation of Group Membership Services for X-by-Wire Protocols

Elizabeth Ann Latronico

A DISSERTATION SUBMITTED TO THE GRADUATE SCHOOL IN PARTIAL FULFILMENT OF THE REQUIREMENTS

for the degree of

DOCTOR OF PHILOSOPHY

in the department of

ELECTRICAL AND COMPUTER ENGINEERING

Advisor: Philip Koopman

Pittsburgh, Pennsylvania

May 2005

Abstract

Distributed fault tolerance algorithms are used for many ultra-reliable systems. For example, aviation fly-by-wire and automotive drive-by-wire network protocols need to reliably deliver data despite the presence of faults. Careful design is required, since ultra-reliable systems permit a failure rate on the order of just 10^{-9} failures per hour. Unfortunately, investing more effort at the design stage does not assure a more reliable product if no objective measurement technique is used at this stage.

The key idea of this dissertation is to estimate the reliability of a service by measuring the probability that the algorithm's maximum fault assumption will be violated. An algorithm's maximum fault assumption states the number of active faults that can be tolerated. The service (and the system) may fail if this assumption is violated. The maximum fault assumption can be tested at design time, before costly design commitments have been made.

This dissertation defines a methodology to measure the reliability of a service's maximum fault assumption. The methodology is applied to clock synchronization and group membership services from three safety-critical protocols — the FlexRay Consortium's FlexRay protocol, TTTech's Time Triggered Protocol Class C (TTP/C), and NASA Langley's Scalable Processor Independent Design for Electromagnetic Resilience (SPIDER). First, I compose an extensive, reusable physical fault model for the aviation and automotive domains. Next, I show how to map this physical fault model to the hybrid fault models in the specifications. For each protocol, I define a Markov model template consisting of an extensible set of states and transitions. Over twenty thousand models are then generated and solved using the NASA Langley Semi-markov Unreliability Range Estimator tool suite.

The methodology identifies the type of fault expected to cause the most failures, and locates trade-off points in the design space where a different fault type becomes dominant. Armed with this information, a designer can target improvements to create and validate a more reliable service. For FlexRay clock synchronization, the Welch and Lynch and the Strictly Omissive formally proven services are compared. For TTP/C and SPIDER membership, customizing the fault diagnosis algorithms to handle transient faults improves the overall estimated reliability.

Acknowledgements

Completing a dissertation can be a long and demanding process. I would like to thank my mom, dad, and Greg for all of their love and support throughout these past five years. Your faith in me made a world of difference! I would also like to thank my brother and sister and extended family for their confidence in me and their kindness.

I would like to thank my advisor, Philip Koopman, for all of his advice, guidance, and encouragement throughout my stay at Carnegie Mellon. I would also like to thank my thesis committee, Priya Narasimhan, Michael Reiter, and Jeffrey Donne, for their valuable feedback and interest. I was fortunate to be able to work with a number of dedicated individuals from industry and government research and development centers. I would like to thank Paul Miner and the Formal Methods group at NASA Langley Research Center for guidance on reliability modeling techniques for protocols and use of the excellent Markov modeling tool suite. Thank you to Alan Baum from General Motors and Michael Paulitsch, Brendan Hall, and Kevin Driscoll from Honeywell for feedback throughout my career as a graduate student. I would also like to thank Michael Bigrigg for use of the machines in the Embedded and Reliable Information Systems Laboratory.

Finally, I would like to thank my research sponsors for their generous support. This research was sponsored by the General Motors Collaborative Research Laboratory and Carnegie Mellon University, the United States Department of Defense (NDSEG/ONR), the National Aeronautics and Space Administration, Langley Research Center, under agreement NCC-1-02043 awarded to the National Institute of Aerospace, the Intel Masters fellowship program, the American Association for University Women Selected Professions fellowship and the Zonta International Amelia Earhart fellowship. I also gained valuable experience through my internships at Robert Bosch Research and Technology Center and NASA Langley Research Center through the National Institute of Aerospace. Thank you to all of the staff who took the time to foster my research.

Table of Contents

List of Figures

List of Tables

1 Introduction

Ultra-reliable systems are difficult to design, and even more difficult to test. Safety-critical aviation and automotive systems allow a failure rate of only 10^{-9} failures per hour [76], [90]. This presents a significant validation challenge. Exhaustive testing is infeasible for ultra-reliable systems, since approximately a billion hours of testing would be needed to demonstrate a 10^{-9} failure rate [15]. It is especially challenging to predict system reliability at design time, when changes are easier to make. X-by-Wire network protocols are an emerging safety-critical technology slated for deployment in production aviation and automotive systems. The X stands for a safety-critical function such as braking or steering, which would be accomplished through a set of local sensors and local actuators communicating over a network. The goal is to eventually create a fully electronic system with no mechanical backup. X-by-Wire systems are expected to ultimately be safer and more cost efficient than current mechanical systems. For example, without some of the restrictions from mechanical components (especially the steering column), the layout of a vehicle could be changed for better crash performance. However, the reliability requirements are greater than the requirements for currently deployed automotive networks which only provide braking and steering assistance. For aviation, manufacturers plan to use X-by-Wire protocols for high-reliability applications such as jet engine control. These networks are also being considered for Unmanned Aerial Vehicles, which may require a low number of failures per hour due to longer missions (since reliability is the probability of providing functionality for the duration of the mission).

Distributed fault tolerance algorithms are used for many services that require high levels of reliability, where a centralized component might present a single point of failure. These algorithms tolerate faults through a combination of redundancy, diagnosis and fault removal. Each service includes a maximum fault assumption - the maximum number of faults that can be present. The goal of the fault tolerance algorithm is to keep the number of active faults within the maximum fault assumption bound. If the maximum fault assumption is violated, the guarantees provided by the service may not hold and the system may fail.

The key idea of this dissertation is to estimate the reliability of a service by measuring the probability that the service's maximum fault assumption will be violated. Essentially, the methodology measures the ability of the service to provide the guarantees it claims to provide. This work focuses on the clock synchronization service of the FlexRay protocol and the membership services of the TTP/C and SPIDER protocols. The FlexRay protocol was created by a consortium of automotive and electronics manufacturers, and is geared towards production automotive systems. The Time Triggered Protocol Class C (TTP/C) was designed by Hermann Kopetz at the Vienna University of Technology and is now managed by TTTech Computertechnik AG. The NASA Langley Scalable Processor Independent Design for Electromagnetic Resilience (SPIDER) protocol suite was created as a case study of reliable protocol design for the US Federal Aviation Administration. The contributions of this dissertation are:

- A methodology for evaluating the reliability of agreement services for X-by-Wire protocols, with respect to a realistic fault model
- Predicted reliability of three protocol services (FlexRay clock synchronization, TTP/C group membership, and SPIDER group membership)

I define a methodology for measuring the failure rate of a service's maximum fault assumption, starting with a realistic physical fault model. I introduce four categories of physical faults and provide a reusable set of fault arrival rates for the aviation and automotive domains, based on industry standards and field data. Next, I show how to map these physical faults to the hybrid fault model used in the specification. For each protocol, I give a Markov reliability model template which includes an extensible set of states and transitions. Fault arrival rates are often uncertain at the design stage. This methodology handles the uncertainty by testing a large number of combinations of fault arrival rates.

Over twenty thousand models are created from the physical fault rate scripts and reliability model templates and then solved, using the NASA Langley Semi-markov Unreliability Range Evaluator (SURE) tool suite. The designer can explore a wide range of design space with a very reasonable amount of work.

To demonstrate the value of this methodology, I study two types of design decisions. One way to try to improve an algorithm is to refine the algorithm's maximum fault assumption, by subdividing one of the classes of faults into easier-to-tolerate fault classes. For clock synchronization, I compare the Welch and Lynch algorithm versus the Strictly Omissive Asymmetric algorithm by Azadmanesh and Kieckhafer. Another important design decision is choosing an appropriate conviction strategy for a fault tolerant membership service. The conviction strategy dictates the conditions required to declare a node faulty and remove it from the group. However, removing too many nodes can also cause the maximum fault assumption to be violated, especially if there are many transient faults. For the TTP/C and SPIDER membership services, I examine three different fault conviction strategies. I show that the standard strategy of removing a node after a single faulty frame is brittle when the system needs to tolerate transient faults. In some cases, failing to use metrics during the design process can lead to a less reliable service. By utilizing this measurement methodology, I show that a superior conviction strategy can be constructed.

The results also show that it is important to include transient faults in the physical fault model. For an early SPIDER study, a simple fail-silent permanent only fault model underestimated the assumption failure rate by a factor of 10^{10} compared to a more comprehensive fault model including transient faults. A more detailed permanent only fault model still underestimated the assumption failure rate by a factor of 10^{5}. All three protocols were more sensitive to the transient fault arrival rates than the permanent fault arrival rates. Finally, I demonstrate how to use this methodology to identify trade-off points in the design space and use these trade-off points to choose system enhancements. Typically, out of the many fault types, there is one dominant fault type that is most likely to violate the maximum fault assumption. The dominant fault type may change at different points throughout the design space. By identifying these trade-off points, a designer can target enhancements appropriately.

Two common enhancements are adding more redundant components, and using higher quality components. In some cases, adding redundant components can actually increase the assumption failure rate, since there are more components that may become faulty (as Powell noted in [91]). This methodology can show whether adding nodes is expected to improve the reliability of a configuration. Using higher quality components can reduce the fault arrival rate of one or more types of physical faults. Sensitivity analysis examines the change in the assumption failure rate due to changes in fault arrival rates, and can identify which fault arrival rate should be reduced. This methodology also paves the way for adaptive fault diagnosis strategies. For example, since the membership services studied guarantee agreement on the number of nodes in the group, a fault diagnosis strategy could be more lenient when there are few nodes in the current group and be more aggressive when there are many spare redundant nodes in the group. Chapter 2 covers related work, Chapter 3 introduces the methodology, Chapter 4 summarizes results, and Chapter 5 contains conclusions.

.

2 Related Work

This methodology for assumption reliability measurement draws on a number of different research areas. Assumption reliability measurement complements other design and testing activities. First, this chapter talks about how the idea of assumption reliability came about, and how one might use this methodology in conjunction with other techniques. This methodology is placed in the context of the V-model, which outlines the set of system development and testing steps. Also, the protocol services are described in relation to the Open Systems Interconnect model. Markov models have been used for many years as a reliability measurement tool, with roots in fault tree analysis which is used to identify single points of failure in a system. More detail is given about the tools used and other tools that are available.

The protocol specification is one input to the methodology, which includes a hybrid fault model that specifies classes of faults with respect to the set of receivers in the system. The hybrid fault model chosen depends on what the algorithm is trying to guarantee. This chapter reviews important developments in guaranteeing agreement and approximate agreement. Each protocol specification also includes information on how frames are scheduled and transmitted on the network, and how fault diagnosis and fault removal are performed.

One contribution of my work is a reusable survey of physical fault data. Physical fault rate data is another input to the methodology. It is common to see in-depth studies of one type of physical fault, but it is difficult to find discussions that include multiple types of physical faults. Also, at the specification stage it may not be possible to get a precise failure rate estimation. I define four categories of physical faults for the aviation and automotive domains, and summarize the important points in each of these categories. There are two benefits here. The fault rate numbers themselves are useful, and can be reused within a domain since most of the faults depend on the environment or the typical component quality. Second, the survey provides a good starting point for a designer who wishes to learn more about a particular category.

2.1 Scope of This Technique

This section discusses assumption reliability as it relates to other system testing techniques. Assumption reliability measurement allows the system to be tested (to some extent) while the specification is still being designed. Testing maximum fault assumption reliability can complement fault injection techniques well. The Markov modeling techniques used are an outgrowth of hazard analysis techniques for safety-critical systems. Assumption reliability testing can be used to reduce risk in a system by assessing the probability of being in a hazardous state. The tools used to evaluate the Markov models are reviewed, as well as other modeling tools that are useful for reliability measurement.

2.1.1 Relationship to Assumption Coverage and Fault Injection

Assumption reliability is a complementary, orthogonal idea to assumption coverage. Powell defines assumption coverage as "The failure mode assumption coverage (px) is defined as the probability that the assertion X defining the assumed behavior of a component proves to be true in practice conditioned on the fact that the component has failed: px = PrX = true|component failed" [91]. Bauer, Kopetz, and Puschner discuss assumption coverage in the Time-Triggered architecture, stating that "In general, every fault-tolerant system relies on the existence of a minimum number of correct components. Thus, even an optimal system architecture, which has 100% assumption coverage with respect to the tolerated failure modes, can never have 100% assumption coverage with respect to the number of coincident faults" [8]. Per Powell's definition, assumption reliability is the ability of the system to withstand faults when all faults are covered (detected through value or timing checks), but coincident faults may exceed the maximum fault assumption.

It is helpful to define the scope of this work with respect to the protocol services provided by the specifications. The Open System Interconnect (OSI) reference model defines seven layers of services for protocols (from highest to lowest): Application, Presentation, Session, Transport, Network, Data Link, and Physical [129]. This model was developed in 1984 by the International Organization for Standardization (ISO), and updated in 1994 [55]. The idea is to separate tasks according to layers so that a layer does not need to know the implementation details for tasks in other layers. (In practice, the OSI reference model serves as a guideline, and layer boundaries are not so strict.) Tasks for the lower four layers are often done in hardware or firmware, and tasks for the upper three layers are often done in software.

The protocol specifications studied describe the functions of the protocol at Data Link and Physical layers. The clock synchronization and group membership services are at the Data Link layer, and require only a few constraints on the physical layer (for example, once the bandwidth is chosen then clocks will have a certain maximum drift rate to support this bandwidth). Current embedded network protocols also mainly support tasks at the Data Link and Physical layers (for example, the Controller Area Network, currently deployed in automotive systems and factory automation applications [93]).

The Data Link and Physical layers are defined as follows:

> Data Link Layer: "The purpose of the Data link Layer is to provide the functional and procedural means to establish, maintain, and release data links between network entities" [129].
>
> Physical Layer: "The Physical Layer provides mechanical, electrical, functional, and procedural characteristics to establish, maintain, and release physical connections (e.g., data circuits) between data link entities" [129].

The methodology I present is a form of specification validation. The goal of the methodology is to measure how well a specification provides the guarantees it claims to provide, when subjected to a realistic fault model. Verification and Validation (V & V), as stated

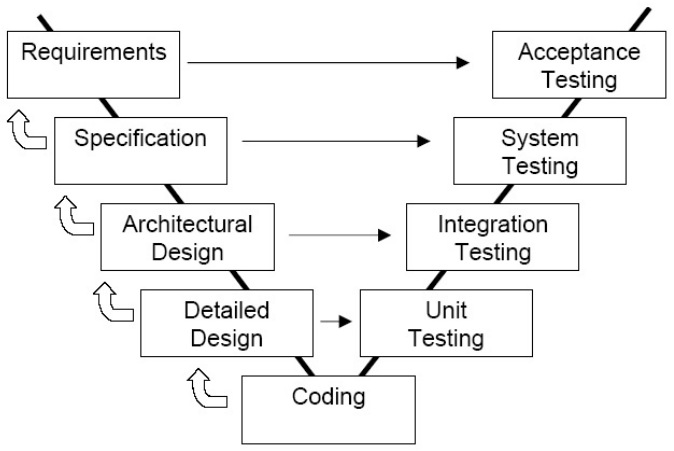


Figure 1. V-model, from [108]

by Boehm, ask two questions [107]: “Validation: Are we building the right product?”, and “Verification: Are we building the product right?” [107]. Sommerville states that “Verification involves checking that the program conforms to its specification. Validation involves checking that the program as implemented meets the expectations of the software customer” [107]. There are many design choices that need to be made at the specification level. For example, what fault diagnosis and removal strategy should be used? My methodology can measure how each proposed fault diagnosis and removal strategy performs under a proposed set of real-world faults.

The V-model is one way to describe the relationship between software development activities and software verification and validation activities. The V-model was developed by IABG for the Federal Republic of Germany [50], and is pictured in Figure 1 from [108]. Software development activities are pictured on the left hand side, with arrows pointing to the right to the corresponding verification and validation activities. Looped arrows on the left between the software development activities indicate possible revisions of previous stages based on the progress of subsequent stages.

One major advantage is that the methodology I present can evaluate the outcome of design changes early, just from information in the specification. This methodology fits in at the Specification -> System Testing stage. The methodology presents a way to test if the specification will provide the claims it says it will provide. Are the specification's guarantees reliable? Typically, testing cannot be done until much later in the design cycle. Early systems developed in a manner similar to the V-model (Requirements through Coding) would be evaluated after the Coding phase was completed, going up the right side of the V (Unit Testing through Acceptance Testing). Problems discovered in later tests could require a large amount of rework.

This methodology meshes well with other verification and validation activities. For Requirements -> Acceptance Testing, the designer needs to decide which guarantees are needed, and what level of reliability should be provided. Ultimately, the methodology I have developed cannot tell the designer which guarantees the system should have. The methodology can only measure the reliability of a guarantee that the system claims to provide. Also, which guarantees are needed will depend on the applications using the protocol. The protocols here are being developed to work with a broad range of current and future applications, so the exact application suite is unknown at the time the protocol is designed. For example, this is one reason the FlexRay consortium has argued that a group memebership service (if provided) should be provided at the application level.

Guidelines are available to help the designer choose an appropriate reliability level, especially for safety-critical systems and for systems that go through a certification process by an outside certification authority. The amount of risk that can be tolerated determines the reliability goal (and level of assurance needed) for the system. Risk is a measure of both the severity and probability of a hazard, where a hazard is "a situation in which there is actual or potential danger to people and the environment" [112]. Aviation certification standards classify risk according to five levels (in order of increasing risk): No Safety Effect, Minor, Major, Hazardous, and Catastrophic [90]. The maximum acceptable probability of occurrence is also stated: none specified for No Safety Effect, Minor $< 1*10^{-3}$ per flight hour, Major $< 1*10^{-5}$ per flight hour, Hazardous $< 1*10^{-7}$ per flight hour, and Catastrophic <

$1*10^{-9}$.

The guidelines for automotive systems are similar. The Motor Industry Software Reliability Association defines five Safety Integrity Levels, according to the ability of the vehicle occupants to control the effects of the failures and the severity of the outcome of the failure [76]. The Safety Integrity levels (from lowest to highest) are: Nuisance Only (Level 0), Distracting (Level 1), Debilitating (Level 2), Difficult to Control (Level 3), and Uncontrollable (Level 4). The corresponding probabilities are Reasonably Possible (Level 0), Unlikely (Level 1), Remote (Level 2), Very Remote (Level 3), and Extremely Improbable (Level 4) [76]. While exact numerical probabilities are not given, similar probability names are used for the five safety criticality levels in the aviation domain [90].

Next in the V-model in Figure 1 are the Architectural Design -> Integration Testing phase and Detailed Design -> Unit Testing phase. For protocols, this often involves developing a static schedule that can fit all frames and ensures that all frames meet their deadlines. Techniques such as Rate Monotonic analysis might be used at this stage, where frames are ordered according to their data message latencies and periods in such a way that all frames can be proveably scheduled [70]. This might also involve a bus analysis tool (for example, CANalayzer is used for Controller Area Network systems). Protocol controller software and hardware would be developed and tested at these stages, using tools such as MATLAB or Simulink. Although the specifications do state limits on certain properties (for example, the acceptable time deviation for a slot window), the specifications try to impose as few limitations as possible and try not to state how to implement the protocol. The last development stage is the Coding stage. There is a lot of interest in ultimately moving to automatic code generation. Since certification costs can be a large portion of the cost of a system (especially aviation systems), the hope is that the code generator could be certified, which would reduce the certification burden for generated code. For example, Esterel Technologies is working on adapting their code generator to the Time Triggered Architecture of the Time Triggered Protocol, Class C [16].

The design time reliability analysis I perform complements existing work in the area of fault injection. In particular, fault injection is well suited to coverage testing, where coverage entails some measure of the percentage and type of faults successfully detected. Fault injection studies are usually performed on a specific implementation (either hardware or software). For example, Fuchs surveys software implemented fault injection (SWIFI) techniques in [38]. Some direct dependability calculations may be possible through simulation. Clark and Pradhan summarize a number of fault injection experiments, and give an example where faults were simulated to enable calculation of a statistically significant mean time between failures [21]. Simulations of up to 200 simulated hours were accomplished in that work. Since exhaustive physical testing is infeasible for ultra-reliable systems [15], fault injection techniques for ultra-reliable systems use accelerated time. White investigates a method for finding the pattern of faults that leads to the greatest system unreliability, called 'Worst Pattern Analysis' [124]. This is used to assess the ramifications of mis-classifying a fault when different fault handling strategies are used (i.e., faults can be treated as other than Byzantine). The methodology I present includes some provisions for misclassification.

Coverage testing through fault injection can be used to verify that the implementation fulfils its requirements (i.e., faults within the maximum fault assumption do not cause unacceptable errors). This would be the Requirements -> Acceptance testing step on the V-model if the complete system is subjected to fault injection. Subsystems could be tested using fault injection as well, which would map to previous testing phases in the V-model. Ademaj, Sivencrona, Bauer, and Torin investigate propagated faults in the TTP/C-C1 version of the TTP/C communication controller [1]. Through software fault injection and heavy-ion fault injection into the instruction memory, register file, hardware registers, and Communication Network Interface, that work reported the percentages of different types of observed errors (slightly off specification, reintegration, asymmetric, and babbling idiot) [1]. The study uncovered a protocol flaw, which was corrected and successfully tested. More detailed information can be found in Sivencrona's dissertation [105].

Fault injection has also been used to test dependability under conditions not covered by the maximum fault assumption. Herout, Racek, and Hlavička tested a C-based reference model of the TTP/C protocol coupled with a set of generic and automotive applications [46]. Part of that work investigated robustness to burst faults that did not conform to TTP/C's maximum fault assumption (the single fault hypothesis) [46]. Ultimately, if there are too many faults outside the maximum fault assumption, then either the maximum fault assumption needs to be revised, an alternative metric needs to be used (for example, availability, if short guarantee lapses are acceptable), or an additional level of services needs to be considered (for example, application level services). Generally, protocols incorporate some additional worst-case fault tolerance mechanisms (for example, blackout detection in TTP/C [118]). The methodology I present estimates the probability of multiple simultaneous faults exceeding the maximum fault assumption, for cases where each individual fault is covered.

Some data exists on the population of faults expected, and possible protocol improvements. Fault and error injection experiments can provide insight into the expected nature and rate of faults that might occur in a deployed system. One strength of fault and error injection is coverage analysis - are all observed errors handled? Results can suggest improvements for error detection and correction mechanisms. Data from fault and error injection experiments can guide choice of parameters in the assumption reliability models. For example, TTP/C prototype fault injection experiments measured a 0.4% percentage asymmetric faults out of all faults observed [1], [16]. The ability to handle transient node faults has also been studied. Rushby proposes a group membership service with a 'probation' state for suspected transient faulty processors and channels in [15]. The service proposed employs majority voting of channel values, which is not done in all group membership algorithms since majority voting requires at three or more channels to mask a single faulty channel. A two channel configuration using majority voting could detect, but not mask, a single faulty channel.

2.1.2 Relationship to Hazard Analysis Techniques: FMEA, FTA

Markov reliability analysis techniques share much in common with hazard analysis techniques originally designed to measure hardware reliability. One requirement of many safety-critical systems is that there cannot be a single point of failure. In other words, if one component fails, this failure cannot cause system failure. Markov analysis overcomes some of the shortcomings of traditional fault tree analysis. The methodology I present uses Markov analysis to be able to model a large range of design space, since many parameter values are uncertain. If the failure rates are precise and fairly certain, traditional techniques can compose these failure rates according to specified rules and produce an estimate of the system reliability. However, for novel designs and systems with transient faults, the expected fault arrival rate range for a particular fault can span two or three orders of magnitude. My methodology adapts to this uncertainty by testing a large number of configurations. This section reviews traditional hazard analysis techniques, and describes the differences between Markov reliability analysis and fault tree analysis.

There are a number of hazard analysis techniques. Storey defines a *hazard* as “a situation in which there is actual or potential danger to people and the environment” [112]. The goal of hazard analysis is to identify hazards, and then identify ways to prevent those hazards. A general guideline for safety-critical systems is that there should be no single fault that can cause a hazard. This is usually achieved through some sort of redundancy. Some types of hazard analysis also try to estimate the probability of a hazard by estimating the failure rates of components. Both bottom-up and top-down techniques are used, usually in combination. Bottom-up techniques start with the individual component faults and postulate possible hazards. Top-down techniques start with a hazard, and postulate sets of faults that could cause this hazard. While these techniques were designed for hardware failure modes, applications of these techniques to software failure modes are also mentioned.

Two important hazard analysis techniques are Failure Modes and Effects Analysis (FMEA) and Fault Tree Analysis (FTA) [112]. Failure Modes and Effects Analysis (FMEA) is a bottom-up technique that “considers the failure of any component within a system and tracks the effects of this failure to determine its ultimate consequences” [112]. The goal is to identify hazardous situations that may arise from component failures. Since this technique considers failures of all of the (hardware) components in the system, it can entail a large amount of work for complex systems. Also, FMEA usually does not consider multiple simultaneous faults [112]. A related technique, Failure Modes, Effects, and Criticality Analysis (FMECA) prioritizes the component failures to study [112]. High frequency and high consequence failures receive the most attention.

Failure Modes and Effects Analysis has also been applied to software faults, called Software Failure Modes and Effects Analysis. One approach to analyzing software failure modes proposed by Goddard treats the software variables (especially input variables) as “components” [42]. The analyst generates a matrix of incorrect variable values and their consequences for each routine being analyzed. Goddard gives an example for a software controlled parking brake in [43], using a Petri Net to model software variables and their values. Since there can be a very large number of software variables in the code, the amount of time required to perform a complete SFMEA for all software variables can be prohibitive, especially if done by hand. One way to address this issue is to perform the FMEA at a higher level. Goddard defines a reusable list of possible system-level failures, which focus on software functions and methods instead of individual variables within those methods [44].

Fault Tree Analysis (FTA) is a top-down technique that “starts with all possible hazards and works backwards to determine their possible causes [112]. Fault trees are a graphical representation depicting the identified hazard and a series of logical operations (AND and OR) that combine events. Fault Tree Analysis starts with the hazard (highest level event) and iteratively defines lower-level events, until the basic individual component failures are identified. Some fault trees also measure the probability of events occurring. If the component failure rates are known, then the event probabilities can be calculated. As an example, Ortmeier and Reif study parameterized probabilities in fault tree analysis for a safety-critical application [84]. Unfortunately, there can be a lot of uncertainty about these failure probabilities, which is difficult to represent in fault tree format. However, fault trees are still useful for describing the relationship between lower-level events and the top-level hazard. The Markov-based reliability analysis techniques are closest in nature to Fault Tree Analysis, but can model a number of additional behaviors. Butler and Johnson list the limitations of fault trees in [14]. Fault trees can be used to model a system with only permanent faults, no reconfiguration, no time or sequence failure dependencies, and no state-dependent behavior such as state-dependent failure rates [14]. However, Markov models are generally more complex to generate than fault trees, and can be difficult to portray graphically since models for realistic systems can have many states and transitions. For example, some of the models in my work had over a hundred thousand states and over two million transitions.

2.1.3 Assumption Reliability Testing as Risk Mitigation

Assumption reliability measures the ability of a service to provide the guarantees it claims to provide. Assumption reliability is one piece of the total system reliability. Leveson states that "*Reliability* is the probability that a piece of equipment or component will perform its intended function satisfactorily for a prescribed time and under stipulated environmental conditions" (italics per original) [69]. Assumption reliability is a conservative measure of system reliability in two ways. First, there may be cases where the maximum fault assumption is violated, but the service is still able to provide its guarantees. Second, there may be cases where the guarantees are not provided, but the applications that rely on the guarantees still perform their intended functions. The total system reliability will also depend on the applications themselves and the physical sensors and actuators. For example, a brake-by-wire system might involve electronic stability control and antilock braking applications, speed and acceleration sensors, and would use an electromagnetic braking actuator at each wheel. Applications use the protocol to transmit data and commands, and physical sensors and actuators collect the data and carry out the commands. If the applications are unreliable, or the sensors and actuators are unreliable, the system will be unreliable even if the protocol works perfectly.

It is helpful to relate assumption reliability to the concept of risk, since the ultimate goal of a safety-critical system designer is to reduce risk. Leveson defines risk as, "*Risk* is the *hazard level* combined with (1) the likelihood of the hazard leading to an accident (sometimes called *danger*) and (2) hazard exposure or duration (sometimes called *latency*)" (italics per original) [69]. Leveson defines a hazard as "A *hazard* is a state or set of conditions of a system (or object) that, together with other conditions in the environment of the system (or object), will lead inevitably to an accident (loss event)" (italics per original) [69]. For my methodology, the hazard is the violation of the service's maximum fault assumption.

To assess risk, the designer must know the hazard levels, the likelihood of the hazard leading to an accident, and the hazard duration. Acceptable hazard levels are defined by (for example) the Federal Aviation Administration for aviation and the Motor Industry Software Reliability Association for automotive, and are reviewed in section 2.1.1. The likelihood of maximum fault assumption violation leading to an accident depends on two issues. First, if the maximum fault assumption is violated, in what situations will the guarantee not be provided? Second, if the guarantee is not provided, in what situations will the system fail? Once the maximum fault assumption is violated, the duration of this violation depends on the protocol, and may be difficult to determine. The shortest hazard duration would be the time it takes to reach consensus (on a clock value within some error for clock synchronization, or on group members for membership). The time to reach consensus is usually a function of communication rounds. For example, the TTP/C best-case time to reach consensus is one round and worst-case two rounds, where a round is usually ten milliseconds long or so. The longest hazard duration is theoretically infinity, since even if consensus is reached quickly, the system can lose consensus again if the underlying problem is not corrected. Alternatively, a system could restart, but might end up in an infinite cycle of restarts.

Calculating risk directly is difficult, since many different applications can use a protocol, including future applications not yet developed. Network protocols are typically reused by a large number of systems and may be used with diverse applications. For example, the Controller Area Network (CAN) protocol is used in about eighty million production vehicles annually [101]. CAN is also one of the leading networks in factory automation (in conjunction with the DeviceNet upper protocol layers), and is used for a variety of other devices from routers to hospital equipment [101]. X-by-Wire protocols will likely be used in many future applications as vehicles and aircraft evolve.

The goal of assumption reliability testing is to reduce risk by reducing the probability of the hazard occurring (violating the maximum fault assumption). Since calculating risk directly is difficult, safety-critical system design techniques often focus on eliminating or reducing the probability of hazards (as in section 2.1.2 on hazard analysis techniques). Investing effort in reducing the probability of hazards at the protocol level should have great benefit, because protocols are reused heavily, and because the variety of applications exercising the protocol makes it more likely that the hazard will cause an accident in one or more of the applications. Another way to think about the problem is that the protocol designer does not have control over an application's behavior. The best a protocol designer can do is to control hazards within the protocol's scope. There will be some design decisions such as whether to offer a guarantee at the protocol level or the application level, but once these are made the protocol's scope is fairly well defined.

2.1.4 Tools

Three Markov analysis tools were used to compute the expected number of assumption violations for the set of configurations studied. The ASSIST program inputs a parameterized text specification and generates the corresponding Markov or semi-Markov model. Then, either the STEM (Scaled Taylor Exponential Matrix) or SURE (Semi-markov Unreliability Range Evaluator) solves the model. The tools require a specification listing state space items, types of transitions, and death state conditions. For our models, only exponential transition rates were used, although the SURE tool allows other types of transition rates if conditional probabilities are specified instead of automatically calculated. The modeling tools generate the complete state space, including all possible death states, which were aggregated according to the assumption violated for our calculations. Output is the expected probability of violating each assumption during the specified mission time. For computational efficiency, a separate model was created and solved for each parameter combination. Butler and Johnson describe the underlying mathematics and give fault tolerance examples in [6]. A detailed example of Markov model creation for the NASA Scalable Processor Independent Design for Electromagnetic Resilience protocol suite is given in [11]. The differential equations used to solve the Markov reliability models are derived by White in [123].

Some techniques use an easier-to-understand graphical model as a front end for the user, then transparently translate this graphical model into a Markov model and solve the Markov model. One approach is to use a graphical fault tree front-end coupled with a back-end Markov solver. This allows a user to augment fault trees in interesting ways without having to learn the intricacies of Markov models. For example, the Galileo tool described by Sullivan, Bechta Dugan and Coppit allows users to model dynamic fault trees, which extend static fault trees to allow failure modes that depend upon the ordering of components, such as a cascading failure [113]. The fault trees are solved using a combination of binary decision diagrams and Markov methods [113].

Petri Nets are another class of graphical models that sometimes use Markov solution techniques as a back-end. For example, with the Stochastic Petri Net Package a user creates a Petri Net model of the system which is translated into a Markov model by the tool [20].

A Petri Net is specified as a set of places, tokens, and transitions, where each token can occupy one place at a time, and the initial marking specifies the initial placement of tokens. Transitions move a token from one place to another, and may create or consume tokens. The advantage is that the user does not have to specify all possible combinations of tokens (i.e. the complete set of states in the Markov model). For example, the Stochastic Petri Net Package can automatically generate all possible states of the system [20]. Some multipurpose tools such as the Symbolic Hierarchical Automated Reliability Predictor (SHARPE) tool allow the user to create and solve a variety of models, such as fault trees, Petri Nets, and Markov models [45]. An application of SHARPE to software reliability estimation is discussed by Gokhale, Wong, Trivedi, and Horgan in [45]. Petri Nets have also been used to study fault tolerance and fault repair strategies, as in [58].

Each of the protocols studied also incorporates some formal techniques at the specification level. Theorem proving gives the strongest assurance of correctness, but also involves the most effort. Proofs could be done by hand (as in the Welch and Lynch clock synchronization work [120]), but modern proofs are often done with the assistance of an automated theorem prover such as the Prototype Verification System (PVS) [95]. Rushby and Stringer-Calvert state that “Formal verification can accomplish what massive simulation and testing cannot: examination of the behaviors of these designs under *all* circumstances” (italics per original) [95]. PVS was used to develop the NASA SPIDER proofs. A formally proven service is assured to provide its guarantees under all conditions, except when one of the assumptions are violated. PVS encapsulates and automates common theorem proving strategies, so the designer can concentrate on the design instead of the mechanical application of logic rules. For example, a fault diagnosis algorithm may have to be revised many times before the desired maximum fault assumption can be proven to be true. Alternatively, the maximum fault assumption may need to be revised.

Model checking is another area of formal techniques that can be used to investigate fault tolerance algorithms. Model checkers are adept at finding counterexamples to a condition.

For example, model checkers can be used to show properties such as a safety property (show that some undesirable behavior X never happens) or a liveness property (show that some desirable behavior Y eventually happens). Model checkers are less powerful than theorem provers, but also typically require less effort. For example, Steiner, Rushby, Sorea, and Pfeifer use the Symbolic Analysis Laboratory (SAL) model checker to analyze the startup algorithm of the Time Triggered Architecture with respect to different degrees of faults. However, they state that "... even with exhaustive fault simulation, a model checker still requires us explicitly to model each fault within the fault hypothesis" [109]. Their future work plans to integrate model checking with PVS proofs [109]. The Symbolic Model Verifier (SMV) is another popular tool, with Burch, Clarke and McMillan discussing the principles behind symbolic model checking in [12]. The SPIN model checker has also been used to check properties of distributed systems, with the distributed leader election example discussed by Holzmann in [47].

Markov techniques are also used in a type of model checking called probabilistic model checking. Instead of proving that a system always satisfies some guarantee, probabilistic model checking can show that a guarantee holds with some probability P. As an example of a general-purpose probabilistic model checking tool, the Probabilistic Model Checker supports probabilistic assurance of properties, including properties modeled with discrete space Markov chains [65].

Finally, there are different kinds of Markov models, including four classes of discrete space Markov chains [97].. If a model satisfies the Markov property (also called the memoryless property), then the "... state at time step k depends only on its state at the previous time-step k-1 ..." [97]. Discrete-time Markov chains only permit probabilistic choice, and Markov decision processes add nondeterministic choice to discrete time Markov chains [97]. Continuous-time Markov chains model continuous time plus probabilistic choice, and probabilistic timed automata allow continuous time, probabilistic choice and nondeterministic choice [97]. The methodology I propose uses continuous-time Markov chains. Some models also allow rewards (or costs) to be associated with model states. Also, steady-state computation is possible for some models, although for the reliability estimates I propose I use a specific mission time since traditionally reliability is measured as the probability of successfully completing a mission.

2.2 Hybrid Fault Models and Guarantees

Safety-critical systems may require distributed algorithms due to the high reliability level and the constraint that there can be no single point of failure. The failure rate of any one component is usually estimated to be about 10^{-6}. Safety-critical systems may require a failure rate on the order of 10^{-9}. Therefore, a centralized solution may be inadequate. However, distributed systems introduce new classes of problems. It is difficult to assure agreement across a group of nodes for some value (such as a data value, the set of members of the group or the current time). Designing an agreement algorithm that can also tolerate faults is even more difficult. The specific guarantee influences the nature of the solution, and subtle differences can require important changes to the algorithm and assumptions. This section will discuss some milestones in the history of agreement algorithms, then review different fault tolerance bounds that have been developed over time. Next, this section will explore some of the differences among guarantees, especially exact agreement and approximate agreement.

The Interactive Consistency problem defines the essential challenges in achieving agreement in a distributed fashion. The idea of Interactive Consistency was proposed by Pease, Shostak and Lamport [87]. The Interactive Consistency problem is defined for a set of n isolated processors, where each processor p has some private value of information V_p, and each processor keeps a vector of values for each of the n processors. An Interactive Consistency protocol assures that for each nonfaulty processor p, "1) the nonfaulty processors compute exactly the same vector", and "2) the element of this vector corresponding to a given nonfaulty processor is the private value of that processor" [87]. The SPIDER proofs restate Interactive Consistency as providing two guarantees, validity and agreement [75]. Validity is defined as “Every good node receives the value sent by a good node”, and Agreement is defined as “All good nodes agree in the value sent” [75]. Achieving Interactive Consistency is also called the consensus problem [36].

Unfortunately, Interactive Consistency is not achievable for certain types of systems (namely, asynchronous systems) in the presence of just a single fault. An asynchronous system makes “no assumptions about the relative speeds of processes or about the delay time in delivering a message” [36]. Fischer, Lynch and Paterson proved that deterministic consensus is impossible in an asynchronous system in the presence of a single faulty process [36]. The problem is that since there are no bounds on processing time or delay time, receivers cannot tell if a sender is faulty or just slow.

Fortunately, fault-tolerant Interactive Consistency is possible if some synchrony assumptions are introduced. Dolev, Dwork and Stockmeyer [28] proved that the (non-trivial) consensus problem requires both a “fixed upper bound Δ on the time for messages to be delivered” and a “fixed upper bound Ω on the rate at which one processor’s clock can run faster than another’s” [31]. Dwork, Lynch and Stockmeyer additionally proved that the bounds do not have to be known priori, and that it is acceptable if the bounds only hold starting at some unknown time T [31]. In that case, the system is called partially synchronous. A Time Division Multiple Access (TDMA) system conforms to both synchrony assumptions, since there is a fixed upped bound on the time for messages to be delivered (the slot window) and there is a fixed upper bound on the clock rate difference (defined in the clock synchronization algorithm).

Since its inception, there have been hundreds of papers published on variations of the Interactive Consistency problem, and there are a few surveys available on work in this area. Chockler, Keidar and Vitenberg review a number of implemented group communication systems in [19]. Galleni and Powell review a number of consensus and membership algorithms, dicsussing the assumptions of each algorithm, advantages, and drawbacks [39].

There are also a number of interesting impossibility proofs that have been developed over the years for group communication systems. Two surveys of impossibility results include the 1989 survey by Lynch [72] and the 2003 survey by Fich and Ruppert [34].

2.2.1 Byzantine Fault Model and Hybrid Fault Models

One of the motivations for distributed agreement algorithms is to be able to tolerate any single arbitrary faulty node, for any possible behavior of that node. This problem is called the Byzantine Generals problem, named after the seminal paper by Lamport, Shostak and Pease [66]. A system that can tolerate an arbitrary faulty node is called Byzantine fault tolerant, and is said to be able to handle a Byzantine fault model. Byzantine fault tolerance is expensive (in terms of the number of nodes required), so various hybrid fault models have been introduced. A hybrid fault model refines the Byzantine fault model to include additional categories for easier-to-tolerate faults. For example, a fail-silent node can be easier to tolerate than a node that transmits arbitrary frames. This section reviews the Byzantine fault model, talks about Byzantine faults in practice, and then reviews some hybrid fault models that are used in the network protocols studied.

The Byzantine fault model by Lamport, Shostak and Pease introduced the idea of classifying faulty nodes according to fault severity with respect to a group of observers [66]. The original Byzantine fault model placed no restrictions on the behavior of a faulty node, thereby covering all possible faulty behaviors. That work proved that Interactive Consistency requires $3f + 1$ processors to tolerate f faulty processors [66]. The $3f + 1$ bound is proveably tight; i.e., if the total number of processors is less than this then there exists at least one case where agreement is violated. This bound refers to the Oral Messages solution where no restrictions are placed on the message contents. In the Signed Messages solution, where an intermediate messenger cannot alter the contents of a message in an undetectable way, only $2f + 1$ processors are required to tolerate f faulty processors [66]. Network protocols do not assume perfect messengers since the network (the messenger) can alter the message in a potentially undetectable manner. A digital signature might be a way around this problem, but current embedded systems are too resource and time constrained to perform the computation required for a sufficiently strong digital signature. Byzantine faults are also referred to as ‘asymmetric’ faults, since observers in the group may have different perceptions of a message due to a single fault. Early implementations of distributed fault tolerance were developed in the late 1970s and in the 1980s. The Software Implemented Fault Tolerance (SIFT) system included an early form of agreement and some fault diagnosis in 1976 [122]. The Multicomputer Architecture for Fault-Tolerance (MAFT) approach in 1988 included a form of Byzantine agreement [60].

A related bound concerns the number of rounds that may experience asymmetric faults. Fischer and Lynch demonstrated that no algorithm can solve the consensus problem in fewer than $f + 1$ rounds if there are f Byzantine failures present [35]. There were many variations on this result, using different system constructions and different types of faults, summarized in [72]. This becomes important since both TTP/C and SPIDER use two rounds in their diagnosis procedures (although a TTP/C round is slightly different from a SPIDER round), so two asymmetric faults within these rounds can lead to cases where consensus cannot be assured.

Byzantine faults are more than just a theoretical category - they can and do occur in practice, and can be caused by natural phenomena. Driscoll, Hall, Sivencrona, and Zumsteg give examples of real-world Byzantine fault scenarios, including a digital signal stuck at 1/2 (which can propagate despite certain fault containment procedures) and a Cyclic Redundancy Code transmitted with weak voltage which receivers could interpret differently [30]. That paper also includes observed examples of Byzantine faults in practice. The authors argue that reducing the probability of a Byzantine fault to some unknown probability is not a strong argument [30]. Kull, Feser, and Köhler describe an automotive example, where a noise pulse occurs on one of the in-vehicle networks when the headlights are switched on [63]. Different voltage levels were measured at a node near the disturbance compared to a node far away from the disturbance.

Another example of a Byzantine fault is a Slightly-Off-Specification (SOS) fault. “In a distributed system where the individual nodes use local hardware to determine about the correctness or incorrectness of the transmission, there is always a ‘grey area’ of parameters (e.g. bit timing, voltage level, etc.) which will be accepted by some nodes but rejected by others” [118, p. 29]. SOS faults can occur in the time domain and the voltage domain. Active star couplers can reduce the chance of SOS propagation, as shown by Ademaj, Sivencrona, Bauer, and Torin, where some Byzantine faults were observed in a bus topology, but none were observed in a star topology where a central guardian utilized active frame reshaping [1]. However, absence of Byzantine faults in testing does not mean they are absent in the final system.

While a Byzantine model covers worst-case fault behavior, the Byzantine model can be very conservative. Hybrid fault models partition faults into categories according to severity. Meyer and Pradhan differentiated between Benign (self-evident) and Arbitrary (all other) faults [74]. Thambidurai and Park introduced a three category fault model including Non-malicious, (Malicious) Symmetric and (Malicious) Asymmetric faults [115]. A Non-malicious fault “. . . is detected by every non-faulty receiver of the message” [115]. A Symmetric fault “. . . refers to the case when all receivers obtain exactly the same message” [115]. An Asymmetric (Byzantine) fault “. . . refers to the case when a message is not received identically by all non-faulty receivers of that message” [115]. Azadmanesh and Kieckhafer introduce strictly omissive symmetric and strictly omissive asymmetric faults, defined below [6]. Powell sorts these categories into Value Error assertions and Timing Error assertions, and presents a hierarchy of fault models in [92]. Strictly omissive asymmetric faults are particularly interesting, because they map well to the behavior of a network protocol with fairly good error detection. In the network protocols studied here, error detection codes transmitted in a frame detect most bit and burst value errors. For timing errors, frames are required to arrive within a local timeslot so frames that are too early or too late are easy to detect. The arrival time of a frame is calculated with respect to the local clock. This avoids some of the problems that might occur if explicit timestamp values were transmitted but corrupted during transmission.

Since the terminology used in each of these papers is slightly different, I use the definitions from the NASA Langley Scalable Processor Independent Design for Electromagnetic Resilience (SPIDER) protocols [75] and from Azadmanesh and Kieckhafer [6]. The SPIDER definitions are based most closely on the Thambidurai and Park model. SPIDER is working on including the omissive category.

- *Good (G)* "Each good node behaves according to specification; that is, it always sends valid messages" [75].

- *Benign (B)* "Each benign faulty node either sends detectably incorrect messages to every receiver, or sends valid messages to every receiver" [75].

- *Symmetric (S)* "A symmetric faulty node may send arbitrary messages, but each receiver receives the same message" [75].

- *Asymmetric (A)* "An asymmetric (Byzantine) faulty node may send arbitrary messages that may differ for the various receivers" [75].

- *Strictly Omissive Asymmetric (A)* "A *Strictly Omissive Asymmetric* fault can send a single *correct* value to some processes and no value to all other processes" [6]. A fault can "garble a message in transit, but not in an undetectable manner" [6].

2.2.2 Group Communication Services and Agreement

A group communication service can play a vital role in the dependability of a network protocol. One type of group communication service is a group membership service. "The task of a membership service is to maintain a list of currently active and connected processes in a group", as stated by Chockler, Keidar, and Vitenberg in their recent Group Communication Specification survey paper [19].

For X-by-Wire protocols as currently designed, there is a single active group. Nodes may reintegrate, but the protocols do not allow for merging of groups (for example, there is no fault tolerance for network partitions). Since the number of nodes physically present in an automobile or airplane is relatively static (except for system upgrades) and is known a priori, having a single active group is a reasonable design. The time to reach consensus on group members is often stated with respect to the number of rounds.

Group membership algorithms are usually designed to withstand node crashes, send faults, and receive faults. Algorithms handle both permanent and transient faults, typically with restrictions on fault interarrival rates [61]. Group membership algorithms cannot compensate for loss of network connectivity or semantically incorrect data that is syntactically correct. Group membership requires at least four nodes to tolerate one Byzantine faulty node [88]. Faulty nodes that lose membership may reintegrate into the system, after the group has reached consensus on its members. However, formally proven reintegration algorithms are not available for these protocols yet. If a fault occurs in the group, additional faults are not tolerated while nodes in the group have inconsistent views of membership, although better fault tolerance is possible for some faults if a slightly longer time is allowed [61].

The TTP/C and SPIDER agreement guarantees are slightly different. TTP/C guarantees agreement on just the membership of the group. SPIDER additionally guarantees agreement on frame contents. FlexRay does not guarantee agreement on group members or data values, treating group membership as an optional application level service.

SPIDER's membership service is defined as follows, where PE stands for Processing Element:

> Message broadcast: Every scheduled message sent by a PE is delivered to all of the properly working PEs. Irrespective of the status of the source PE, all of the properly working PEs will agree on the content of the message. If the source is working properly, all of its messages will be received exactly as they

are sent. [117, p. 2]

TTP/C's membership service is defined in the specification as:

> Consistent Membership Service: The TTP/C controller informs its host computer about the state of every other computer in the cluster with a latency of less than two TDMA rounds. The membership service employs a distributed agreement algorithm to determine, in case of a failure, whether the outgoing link of the sender or the incoming link of the receiver has failed. [118, p. 6-7]

2.2.3 Clock Synchronization and Approximate Agreement

For some problems, exact agreement on an identical value is not needed or is not feasible — instead, nodes might need to agree on values that are close to one another. This version of agreement is called Approximate Agreement, and is used in clock synchronization. Dolev et. al. state that an approximate agreement algorithm must satisfy the following two conditions [29]:
"Agreement: All nonfaulty processes eventually halt with output values that are within ϵ of each other" [29].
"Validity: The value output by each nonfaulty process must be in the range of initial values of the nonfaulty processes" [29].

The goal of clock synchronization is to keep all of the good nodes' local clocks within some error ϵ of each other. There are two types of tasks in clock synchronization: rate and offset correction, and approximate agreement on the correction values. Proofs for clock synchronization will generally address both types of tasks. Clock synchronization requires both rate correction and offset correction. For two clocks c_{fast} and c_{slow}, the faster clock's value will eventually diverge from the slower clock's value, even if both start out at the same initial time value. Rate correction handles this issue. Also, clocks might not start out at the same exact initial time, or might decide on slightly different time values, even if the rates are the same. Offset correction reduces this fixed difference.

This methodology concentrates on the approximate agreement portion. Each node collects a set of clock values from the other nodes. In these statically scheduled networks, explicit timestamps are not used. Instead, the clock value is computed with respect to the receiver's local clock as the time when a frame arrives within the receiver's slot window. From these clock values, the receiver computes its own local rate and offset correction parameters.

Descriptions of the clock synchronization services for each of the protocols are given below. For each protocol, a minimum precision is also specified, and the precision is often related to other parameters such as the bandwidth, the number of frames supported and the physical layer properties.

FlexRay: "The primary task of the clock synchronization function is to ensure that the time differences between the nodes of a cluster stay within the precision" [33, p. 158].

TTP/C Fault-Tolerant Global Time Base: "The TTP/C controllers process a fault-tolerant clock synchronization that establishes a sparse global time base without relying on a central time server. The time base with a precision in the microsecond range is provided to all host computers" [118, p. 6].

SPIDER: "ROBUS-2 provides an accurate and precise time reference to the PEs, which they can use to coordinate their actions" [117, p. 2]. A PE is a Processing Element, and ROBUS stands for Reliable Optical Bus (although the proofs are valid for any communication medium).

2.3 Network Protocols

Time Division Multiple Access (TDMA) networks provide a statically-scheduled method of transmitting data on a broadcast bus. A frame is the basic unit of transmission, and includes a data payload plus overhead fields such as a Cyclic Redundancy Code for error detection and various header and status bits. A TDMA network contains one slot (per channel) for each frame to be transmitted. Slots are defined according to their order in a round. Each node typically has at least one slot per round [7]. Rounds are assembled into a cluster cycle [118]. Typically a dual-redundant bus (or star) is used, and safety-critical frames are replicated on all channels. Some protocols allow different frames to be transmitted on each channel as a performance optimization for non-critical frames. Three safety-critical network protocols are studied: FlexRay, the Time Triggered Protocol Class C (TTP/C), and the NASA Scalable Processor Independent Design for Electromagnetic Resilience (SPIDER) family of protocols.

Each studied protocol uses a TDMA media access scheme. The unique characteristic of a TDMA scheme is that access to the communication bus is statically scheduled. Each sending node receives a time slot, and bus guardians ensure that only the correct sender is allowed to send in a particular slot. The main motivation for using a TDMA strategy is to prevent a 'babbling idiot' fault, where a single faulty node monopolizes the bus. In a TDMA scheme, at least two faults are required in order for a babbling idiot node to monopolize the bus. Both the node and the bus guardian must fail. In priority-based schemes, any node could be the next sender, so designing a bus guardian would not be feasible without additional knowledge. The Controller Area Network (CAN) network currently used in automobiles is priority-based. Although messages are statically scheduled on top of CAN, FlexRay is being designed as the successor to CAN to eliminate the babbling idiot failure mode. This section describes properties of TDMA protocols in more detail, then reviews each of the three protocols studied and the supporting formal proofs for each protocol.

2.3.1 General TDMA Concepts

All three of these protocols use a Time Division Multiple Access (TDMA) scheme to broadcast data on the communication medium. In a TDMA scheme, each node is statically assigned some number of frame sending slots within a round. One or more rounds form a communication cycle, which is continuously repeated. Bus guardians protect against 'babbling idiot' faults, where a node attempts to transmit outside of its slot. Constructing a

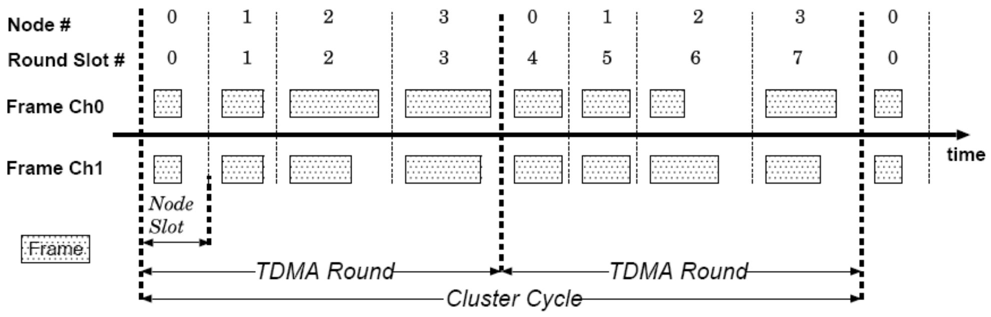


Figure 2. TDMA Round, from TTP/C Specification [118]

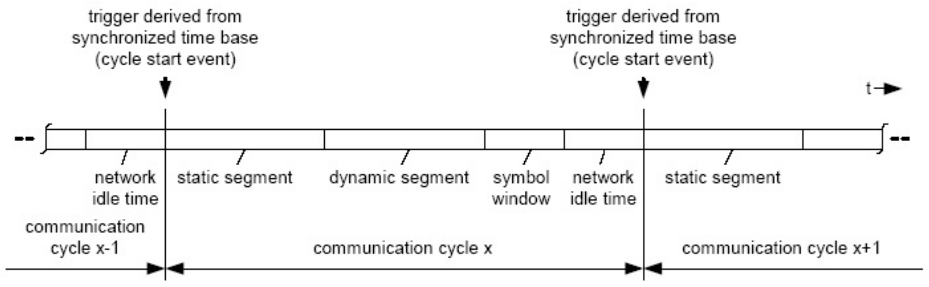


Figure 3. FlexRay Round, from FlexRay Specification [33]

bus guardian without a static schedule would be infeasible, since the bus guardian could not predict which node is supposed to send next. This is a key advantage over currently deployed priority-based bit dominance protocols (such as the Controller Area Network), where a single babbling idiot node might monopolize the network. TDMA networks can also achieve higher data transfer rates than the Controller Area Network since there is no need for arbitration for control of the network.

Figure 2, from version 1.4.3 of the TTP/C specification, illustrates the concept of slots, nodes, and cycles in a TDMA scheme [118]. The set of possible senders is known at design time. The SPIDER protocol also uses a sending scheme similar to Figure 2. The FlexRay communication protocol allows an optional Dynamic segment (permitted setups are completely Static, Static followed by Dynamic, and completely Dynamic). Figure 3 from version 2.0 of the FlexRay specification illustrates static and dynamic segments, plus some time slots reserved for other purposes. Safety-critical information would be sent in the Static segment of a FlexRay sending scheme.

Since one faulty channel could be a single point of failure, multiple channels are used. FlexRay and TTP/C employ a dual, redundant channel design where any well-formed frame is accepted. SPIDER uses a slightly different topology with three channels, as SPIDER additionally guarantees agreement on the frame contents which requires voting at the receiver to tolerate Byzantine faults. A well-formed frame must contain a valid Cyclic Redundancy Code check, must be within a certain time window, and must pass any additional error checks (for example, a matching membership view) [118]. For FlexRay and TTP/C, the channel may be a passive bus or frames may be broadcast via an intermediate star coupler. SPIDER broadcasts frames via Redundancy Management Units. The FlexRay clock synchronization service and the TTP/C membership service treat only nodes as first-class entities. The SPIDER membership service has two classes of nodes - Bus Interface Units (similar to the nodes in FlexRay and TTP/C) and Redundancy Management Units (similar to star couplers in FlexRay and TTP/C).

Many variations in topologies are possible, and are based on either a bus configuration or a star configuration. A topology may also be a combination of the two. All nodes are assumed to be fully connected, and all communication is broadcast. In general these protocols are not designed to handle network partitions (other than with a mechanism such as restart). The SPIDER topology is slightly different, and is covered in the NASA SPIDER section. Figure 4 from the TTP/C specification illustrates a bus topology and a star topology [118].

Other researchers have also evaluated safety-critical network protocols. In his detailed comparison of TTP/C, the NASA SPIDER protocols, the Honeywell SAFEbus network, and FlexRay, Rushby argues that "Any fault-tolerant system must be designed and evaluated against a specific *fault hypothesis* that describes the number, type, and arrival rate of the faults it is intended to tolerate" [96]. Kopetz discusses the fault tolerance abilities of TTP/C vs. Flexray in [62], and the PALBUS project reviews a number of data buses including an early version of TTP/C [106].

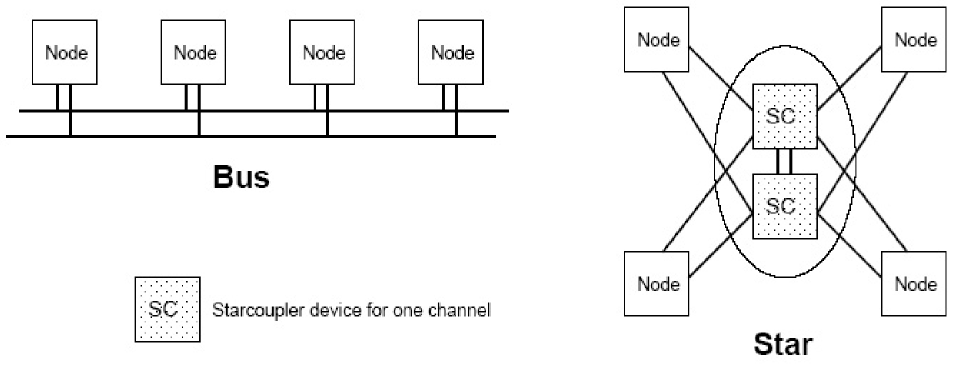


Figure 4. Bus Topology and Star Topology, from TTP/C Specification [118]

2.3.2 FlexRay

FlexRay is a next-generation automotive protocol intended for safety-critical automotive applications such as brake-by-wire, where electronic connections will replace the mechanical linkages between the brake pedal and the braking actuators [33]. The FlexRay protocol guarantees distributed clock synchronization among member nodes. FlexRay is being developed by a consortium of industry members.

FlexRay was designed as an alternative to TTP/C for the automotive arena, and has two main differences. In addition to the traditional TDMA statically scheduled round, FlexRay features an optional 'dynamic segment' where nodes may choose to send additional information. The name "FlexRay" stems from this flexible scheduling. However, safety-critical information will still be transmitted using the static segment. Besides support of event-based messages, one of the reasons for including a dynamic segment was that FlexRay could then support the byteflight protocol format. byteflight (intentionally lowercase) was developed by Motorola, BMW, and a few additional companies for use in BMW vehicles. The second main difference is the FlexRay perspective on group membership. The FlexRay view is that group membership, if implemented, should be implemented at the application level. FlexRay provides only clock synchronization at the protocol level.

The FlexRay clock synchronization algorithm is based on the formally proven algorithm from Welch and Lynch [120], involving rate and offset correction. Rate correction compensates for clocks running at slightly different speeds, while offset correction corrects fixed discrepancies in local clocks. Recently, an improved bound was developed for a family of clock synchronization algorithms. The Welch and Lynch clock synchronization algorithm belongs to a broader class of algorithms called Mean-Subsequence-Reduced (MSR) that operate on local sets of values [6].

2.3.3 Time Triggered Protocol, Class C (TTP/C)

The Time Triggered Protocol, class C is a protocol designed to meet "the requirements for safety critical distributed real-time systems in several application domains ..." [118, pg. 1]. TTP/C was originally developed at the Vienna University of Technology by Hermann Kopetz, and is now owned by TTTech Computertechnik AG. The class C portion of the name refers to the Society of Automotive Engineers (SAE) class C requirements for fault-tolerant real-time automotive networks. There are other TTP variants for less critical applications, namely TTP/A for SAE class A applications. Kopetz also developed the Time Triggered Architecture (TTA), which states architectural principles independent of a particular protocol specification [62].

TTP/C's fault tolerance is based in part on a formally proven membership service. The TTP/C protocol specification also employs some "Never Give Up" mechanisms for severe fault situations such as network blackout [118]. Reintegration strategies are also under development. However, reintegration and last resort mechanisms are still evolving and are not formally proven. Therefore, the reliability analysis does not investigate how any additional mechanisms might alter the system reliability. Strictly speaking, it is possible that reintegration or a "Never Give Up" mechanism might actually interfere with normal operation, unless proven otherwise. Some reintegration problems (and possible solutions) are discussed by Bauer, Kopetz, and Steiner [9]; namely, a faulty node could transmit faulty system state information which a reintegrating node may think is correct.

The TTP/C protocol offers an integrated membership service with low overhead per frame. As stated in the specification, the membership service "…informs all nodes of a cluster about the operational state of a node within a latency of about one TDMA round" [118, p. 67]. In the TTP/C membership service, each node keeps a local membership vector consisting of one bit per node in the system (system size is usually in the tens of nodes). Each sender incorporates its vector into the frames it sends. If the received vector does not match the local vector, the formally proven Clique Avoidance and Implicit Acknowledgement algorithms ensure that only the majority clique survives (with tiebreaker rules) and that consensus is reached within two rounds [88], [7]. Nodes are immediately removed if suspected. A sender will lose membership if not enough other nodes receive the sender's frame correctly [88]. If an asymmetric (Byzantine) fault occurs, there is no guarantee that the asymmetric faulty node will be removed, since it may or may not belong to the minority clique. When an asymmetric fault occurs, at least one node and at most half of the nodes in the current group will lose membership, if the Single Fault Hypothesis is not violated.

TTP/C's membership service relies on the TTP/C Single Fault Hypothesis. The TTP/C specification states, "TTP/C is based on the fault-hypothesis that any single component in the system can fail in an arbitrary failure mode. This assumption is based on the fact that the likelihood of two concurrent independent component failures is remote enough to be considered a rare event that can be handled by an appropriate *never-give-up (NGU)* strategy" (italics per original) [118, p. 27]. Specifically, if one fault occurs, and an additional fault occurs before the group has reached consensus on its members, it is possible that the group will never achieve consensus on its members. Proofs by Bouajjani and Merceron and by Pfeifer verify that after a fault, it will take at least one communication round and at most two communication rounds for the group to achieve consensus on its members if all nodes transmit a frame exactly once per round, and there are at least four nodes in the original group [11], [88]. If faults occur in a manner that violates the Single Fault Hypothesis, agreement may not hold and the system may fail.

TTP/C can save on bandwidth by including the membership vector implicitly in the error checking Cyclic Redundancy Code (CRC) calculation. TTP/C identifies improperly formatted frames using a CRC value included with every frame to detect corrupted data, and by comparing local membership vectors to detect mismatch. The CRC polynomial must have a Hamming distance of 6 [118, p. 43], which at minimum ensures detection of up to five single bit errors. Also, burst errors up to the length of the CRC are detected (usually 24 bits). Each frame includes the sender's local membership vector, either explicitly or implicitly. An explicit membership vector is transmitted as part of a frame's contents. For an implicit membership vector, the sending node calculates the CRC over the transmitted frame contents and its local membership vector, but does not transmit a copy of its local membership vector. A receiving node will check the CRC using the received frame plus the receiver's local membership vector (which should match the sender's if no faults have occurred.) If a replicated frame has an incorrect CRC for all channels, that frame is considered 'invalid' and cannot be used for a positive agreement [118, p. 41-42]. Note that TTP/C does not guarantee that the contents of a frame will match at all receivers, since it is possible that two different well-formed frames could arrive at a receiver. Also, some explicit membership vectors must be transmitted if nodes are permitted to reintegrate.

In TTP/C membership, only nodes can be members — link faults are mapped back to one or more nodes. Since TTP/C uses the CRC to determine the validity of a frame, network noise that corrupts the frame will result in a failed CRC and an invalid frame. If an invalid frame is received on all channels, the sending node is perceived as faulty by all other nodes (if the link fault is symmetric) or by some other nodes (if the link fault is asymmetric). TTP/C's membership assumptions therefore apply to network faults as well. Note that the membership service assumptions are stated with respect to units of one frame. A single noise fault on the network may create multiple erroneous frames, if the fault overlaps frame boundaries.

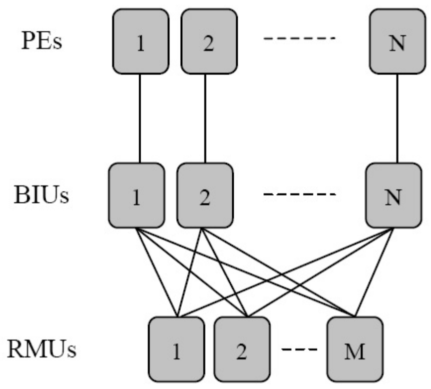


Figure 5. SPIDER Topology, from [117]

2.3.4 NASA SPIDER

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 [40]. At the heart of SPIDER is the Reliable Optical Bus (ROBUS), which processing elements use to reliably transmit data in a fully-connected, broadcast manner. Formal proofs define the fault tolerance abilities of the ROBUS. The proofs are valid for all transmission media. The ROBUS has two types of components: Bus Interface Units (BIUs) and Redundancy Management Units (RMUs). The BIUs are fully connected to all RMUs, and vice-versa. Each BIU has a one-to-one connection to a corresponding Processing Element (PE). A Processing Element cannot exhibit asymmetric ("Byzantine") faulty behavior, since there is only one direct consumer of its data. The SPIDER voting algorithms are based on work by Davies and Wakerly [24], and the hybrid fault model is based on the Thambidurai and Park model [115].

Asymmetric BIU or RMU faults are handled internally by the ROBUS, freeing the application designer from this concern, as long as the proof assumptions hold. Figure 5 from Torres-Pomales, Malekpour and Miner illustrates the SPIDER architecture [117].

The SPIDER Interactive Consistency algorithm and Diagnosis algorithm provide four guarantees [117]:

“Validity: Every good node receives the value sent by a good node” [117].

“Agreement: All good nodes agree in the value sent” [117].

“Conviction Agreement: All good nodes agree on convictions” [117].

“Correctness: No good node is ever convicted” [117].

Validity and conviction agreement are provided by both SPIDER and TTP/C, while only SPIDER guarantees agreement on the data values. Protocol designers can choose to guarantee correctness, completeness, or neither. Geser and Miner state, “In the presence of arbitrary asymmetric failures, it is impossible to guarantee both correctness and completeness, where completeness means that “all faulty nodes are eventually convicted” [40]. The correctness property is also called accuracy by some membership services [18]. By preserving correctness instead of completeness, SPIDER can accumulate evidence against a node as an alternative to immediate conviction. In comparison, guaranteeing completeness could require conviction of transiently faulty nodes, which can significantly decrease assumption reliability (as the results show). A main disadvantage to correctness is that it is not possible to convict faulty nodes in some cases.

2.4 Physical Fault Rate Data

This section surveys major types of physical faults that can affect embedded wired networks. I use observed fault data from experiments and real-world measurements to obtain parameters for the reliability models. I place faults into one of four categories according to the physical cause: permanent hardware and link faults (from normal use), single event effects (from radiation), bit error rate (uncorrelated network corruptions), and electromagnetic interference (bursts of noise on the network). Predictive rate models are outside the scope of this paper, but do exist, if more precision is desired. This survey is not intended to be used as a complete fault list, as each system will have its own unique set of faults. However, this survey should be helpful and reusable as a more comprehensive benchmark for assumption reliability testing.

2.4.1 Permanent Hardware Faults

Faults in this category are generally fail-silent, and may affect both nodes and links. Due to reasons such as wear-out or other physical damage, hardware may cease to function at all, becoming an inert component that neither generates nor receives messages. These permanent faults last indefinitely (or until component repair or replacement). The fault rate for a fault containment region will be a function of component fault rates within that region. Fault containment regions imply that no common-mode failures exist between two regions (otherwise, the components would belong to one fault containment region). Therefore, these faults are modeled as independent. For permanent hardware faults, the reliability models assume that components are in their useful life stage, with a constant fault arrival rate.

Extensive data is available for failure rates of electronic components. For example, MIL-HDBK-217 (Reliability Prediction of Electronic Equipment) describes the failure rates of hundreds of types of electronic components and connectors [119]. At the protocol design stage, order of magnitude approximations are appropriate. The failure rate of a bus, star coupler, or SPIDER RMU node is modeled as 10^{-6} failures/hour, which is in line with the military handbook data. FlexRay and TTP/C nodes and SPIDER BIU nodes have a larger fault containment region (including both the transmitter and a possibly complex processing element), so this failure rate is modeled as 10^{-5} failures/hour. The probability of a link or connector fault depends on many characteristics, including the type of link, the type of connector, the temperature and the thickness of the link. Base failure rates for a pair of connectors range from approximately 10^{-7} to 10^{-10} in MIL-HDBK-217F [119], before accounting for environmental and usage factors. Failure rates for the links themselves are quoted by Hyle in a 1992 study as approximately 4.35-5.26 * 10^{-6} for fiber and 1.15-1.81 * 10^{-6} for copper [49]. The reliability models use a range of 10^{-8} to 10^{-6}, which is slightly pessimistic compared to [119] and slightly optimistic compared to [49]. Care must be taken when researching cable failure rates, as a major source of link faults in wide area networks is unintentional cable dig-ups, which would not apply to embedded networks.

2.4.2 Single Event Effects

Single event effects (SEEs) arise from particle collisions that deposit energy which causes an inappropriate electric field or potential somewhere in the circuit. SEEs are classified both by the effects produced and the dominant particle source. The chance of a particle causing a particular effect depends on many parameters of the integrated circuit design, including voltage, transistor and feature size, and radiation hardening. The total number of observed effects also depends on the particle flux, which varies with altitude, latitude, and manufacturing materials (in the case of alpha particles). Since this research addresses broad design-time concerns, single event effects rates will be analyzed at order-of-magnitude granularity. No system implementation details are assumed. However, detailed single event effects arrival rate models and measurements for production integrated circuits are available if more is known about the implementation and environment [82]. First, the neutron and alpha particle sources of SEEs are discussed. Next, both non-destructive (transient) and destructive (permanent) effects are summarized.

Particles

Single event effects are caused by particles striking the integrated circuits and interfering with its operation. Four categories of particles are examined: heavy ions (defined as any particle with atomic number greater than or equal to two [27]), neutrons, protons, and alpha particles. The two primary methods by which particles interact with an integrated circuit are direct ionization (most common for heavy ions) or indirect ionization (most common for neutrons, protons, and alpha particles). "Direct ionization occurs when a charged particle passes through a circuit and becomes lodged in the semiconductor material, after losing all of its energy" [27]. Indirect ionization occurs when energy is deposited by colliding

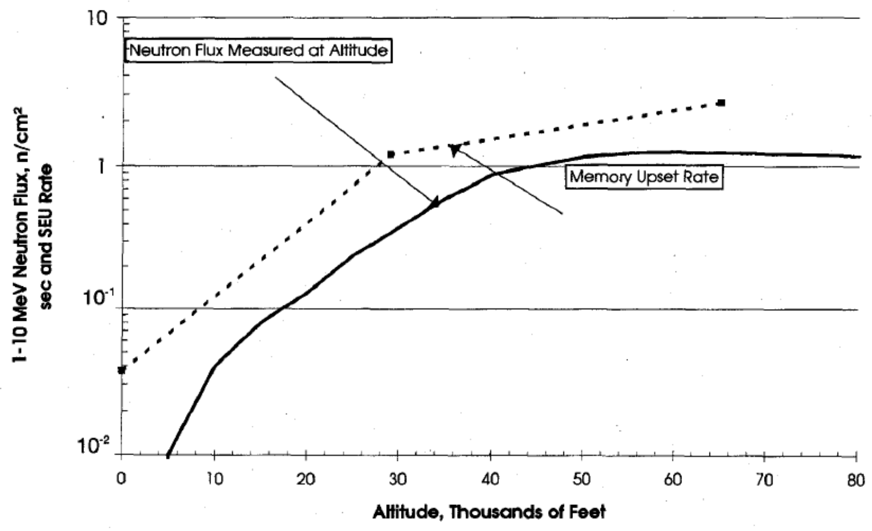
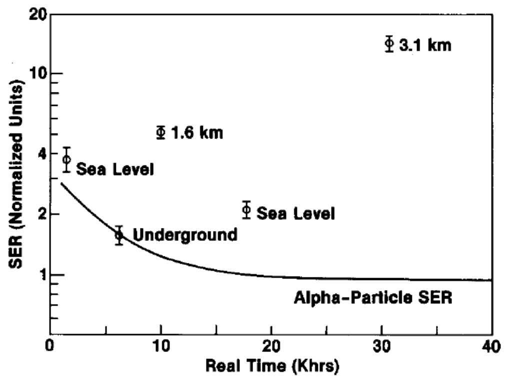


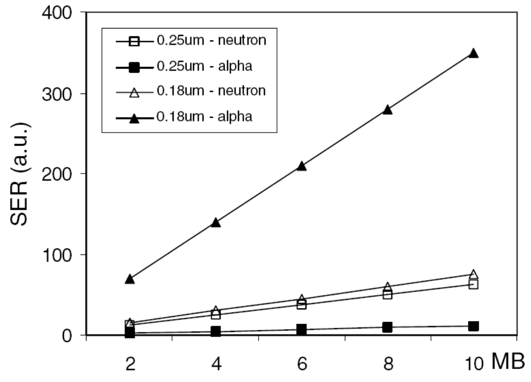
Figure 6. Neutron Flux vs. Atmospheric Depth, from [79]

particles, but the particles themselves do not remain lodged in the circuit [27]. The expected SEE arrival rate depends both on the circuit construction and the operating environment. Data is available on particle concentrations in various environments (both terrestrial and higher in the atmosphere), for example, as in [79]. Testing results are available for a large number of commercial components, for example, as in [82].

Atmospheric neutrons and alpha particles are the main contributors to SEE for the systems considered in this research. At elevated altitudes, atmospheric neutrons are the main cause of SEEs [79]. Neutron flux and proton flux are not uniform throughout the atmosphere, and therefore the number of SEEs experienced will depend on a circuit's operating environment. As shown in Figure 6 from [79], the amount of neutron flux increases up to an altitude of about 60 thousand feet before leveling off; proton flux also peaks at this height [79]. Neutron flux also depends on latitude, with higher flux at the poles and lower flux at the equator [79]. Proton flux is higher at the equator that the poles [79]. Figure 6 from [79] shows that neutron flux is the dominant cause of upset for aircraft, as changes in the in-flight upset rate follow changes in the atmospheric neutron flux (this is true for changes in atmospheric neutron flux due to latitude too [79]). In Figure 6, the solid line is



(a) Alpha Particle Contribution to Soft Error Rate, from [83]



(b) Single Event Rate for Two CMOS SRAM Generations, from [22]

Figure 7. Alpha Particle Contribution to Single Event Upset, from [83],[22]

the $1*10^{-10}$ MeV atmospheric neutron flux and the dotted line is the Single Event Upset rate in upsets/bit-day, multiplied by $1*10^{7}$.

While neutrons are currently the main contributor to SEEs at higher altitudes, alpha particle radiation may become an increasing concern in the future. For alpha particles, the method of manufacturing and the circuit geometry influence the rate of bombardment and susceptibility. Alpha particles are emitted by radioactive decay of trace elements in the chip packaging [83]. To determine the alpha particle contribution to the Single Event Rate (SER), O'Gorman studied the SER of chips placed at four different heights (including 200 m underground, where no atmospheric particle flux would be present). Figure 7(a) from [83] illustrates the alpha particle contribution to SER over time (since alpha particle radiation will decrease with time due to decay), and shows that for the 288 KB DRAM chip tested, alpha particles were not the dominant cause of SER at higher altitudes. O'Gorman also found that chips from two separate sample sets had different amounts of alpha particle radiation, due to a change in the manufacturing process [83]. The danger posed by alpha particles increases as integrated circuit geometries decrease. Constantinescu depicts this increase in Figure 7(b) [22] for CMOS SRAM technology. For the 0.25 μm technology, the neutron induced Single Event Rate (SER) was higher than the alpha particle induced SER. However, the situation is reversed for 0.18 μm technology, with the alpha particle induced SER dominating. While the susceptibility to alpha particles is different for different semiconductor technologies, overall, smaller cell capacities and lower supply voltages will increase the probability of Single Event Effects.

Non-Destructive Single Event Effects

Non-destructive single event effects refer to effects that do not cause physical damage to the circuit, although some may be permanent if not detected and corrected. In memory devices (DRAMs and SRAMs), single event upset (SEU) is the most prevalent type of single event effect. SEU refers to the change of a bit from one stable binary state to another, although the physics of the actual upset are different for DRAMs and SRAMs and the effects of the particle strike depend on the strike location and amount of energy [27]. A single event multiple bit upset (MBU) occurs when a single particle is responsible for the corruption of multiple bits. MBUs may become more prevalent as more bits are stored in smaller areas [27]. An integrated circuit (IC) may also experience single event functional interrupts, where a particle strike "triggers an IC test mode, a reset mode, or some other mode that causes the IC to temporarily lose functionality" [27]. In logic, particle strikes may cause combinational soft faults (SF), where an error event is said to occur if outcome of the logic circuit is altered because of the fault [27]. Mitigation techniques include error detection and correction, physical device hardening and insulation, and redundancy [27].

Neutron flux models continue to be a good predictor of SEU rate. Table 1 from [79] lists measured in-flight SEU rates compared to rates calculated from either the Burst Generation Rate (BGR) method or neutron-cross section method using neutron flux data for a variety of SRAMs. Note that all of the predictions were within an order of magnitude of the measured rates, and most were much closer. Experiments in a neutron beam test chamber at the Weapons Neutron Research facility corroborate with these results. SEU upset rates (upsets/bit-hr.) for ARINC 429 receivers were in the range of $1.1*10^{-9}$ to less than $5*10^{-10}$

Table 1. Measured In-Flight Occurrences of Avionics Single-Event Upset, from [79]

Flight Path	Altitude (K feet)	Operating Voltage	Measured Rate upsets/ bit-hr.	Calculated Rate upset bit/hr.
Seattle	29	2.5 V standby	$5*10^{-9}$	$4.4\text{-}8*10^{-9}$
N. California	65	2.5 V standby	$1*10^{-8}$	$1\text{-}2*10^{-9}$
Norway	65	2.5 V standby	$2.3*10^{-8}$	$2\text{-}4*10^{-8}$
Norway	65	2.5 V standby	$4.6*10^{-9}$	$8\text{-}14*10^{-9}$
Europe Area 1	29	5 V	$2.3*10^{-9}$	$1.8\text{-}4.7*10^{-9}$
Europe Area 2	29	5 V	$1.6*10^{-9}$	$1.3\text{-}2.7*10^{-9}$
Out of Seattle	29	5 V	$1.6*10^{-9}$	$1.3\text{-}2.7*10^{-9}$
Transatlantic	35	5 V	$2*10^{-9}$	$1*10^{-9}$
Worldwide	25	5 V	$3.3*10^{-10}$	$5*10^{-10}$
Worldwide	33	5 V	$4.1*10^{-10}$	$5*10^{-10}$

for a simulated neutron flux of 40,000 feet [81]. As noted in Figure 7(a), alpha particle contribution to SEU may be significant at sea level or where atmospheric particles are limited (such as underground) [83].

On a per-device basis, the number of device upsets/hour can be quite high. To get the expected device upsets/hour, the bit upsets/hour rate is multiplied by the number of bits that could be affected in the circuit. This computation is valid as the reverse process is typically used to calculate the upset bits/hour SEU rate, especially for memory circuits. The total number of observed faults in an experiment is divided by the total number of bits that could be affected.

Destructive Single Event Effects

Single event effects (SEEs) that may cause permanent hardware damage are called destructive single event effects. Sexton classifies destructive SEEs into four categories, depending on the effect produced and the susceptible devices [99]. Single event latchup is the primary destructive single event effect considered in the reliability models.

Latchup, where energy deposited by an incident particle creates a low resistance path is created between power supply and ground, may occur in devices with a four-layer *pnpn* structure such as the CMOS family of integrated circuits [99]. Latchup is characterized by a high current draw that persists until power is removed or the device fails [99]. Similar to latchup, Single Event Snap back (SES, also called transistor latchup) is characterized by a high current condition. Snap back requires sufficient current from an external circuit to be sustained, but does not require a four-layer *pnpn* structure. The chance of SES occurring also depends on the SES threshold versus the Single Event Upset (SEU) threshold, a non-destructive single event effect. Usually, SEU will occur before SES can, although SES has been shown to be a concern in ICs that are intentionally hardened against upset using feedback resistors and Silicon On Insulator technologies [99].

Single Event Burnout (SEB or SEBO) and Single Event Gate Rupture (SEGR) affect powered bipolar transistors and power MOSFETs. In single event burnout, energy deposited by a particle initiates a large source-drain current, which leads to destructive burnout [79]. In single event gate rupture, the gate dielectric that isolates the gate and channel regions fails [99]. Burnout has been induced by protons, neutrons, and heavy ions, whereas gate rupture has only been shown to be induced by heavy ions [79].

Overall, latchup is a greater concern than SEB and SEGR for avionics devices since most devices use low power, a trend that will continue in the future. SEB and SEGR become less important considerations as the power rating of devices drops. Devices with higher voltage ratings show a higher susceptibility to SEB and SEGR. In one study of power MOSFETs subjected to energetic protons, SEB events were observed in 200-V rated parts but not in 100-V rated parts [79]. In tests involving MOSFETs bombarded with a neutron beam, no SEB events were observed in devices rated 300 V or less [79]. Normand states that "Since in current avionics applications these parts are operated at voltages lower than 300 V, however, it appears that SEBO is not a real issue for current avionics" [79]. For SEGR, the operating voltages for current technologies should be below the minimum voltage required for SEGR to occur. Maximum operating voltages are typically around 5.5 V and 3.6 V [99]. In contrast, the minimum holding voltage for latchup is about 1.0 V [99].

A single event latchup range of 10^{-8} to 10^{-6} is tested in the reliability models. Worst-

Table 2. Latchup Rates in Gate Arrays in WNR Beam Tests, from [81]

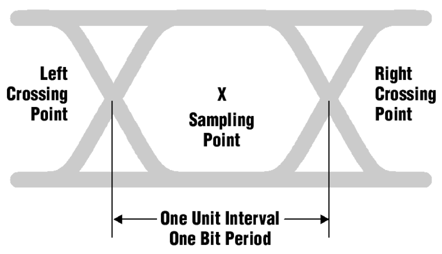
Part	Test Year	Equivalent Hours at 40000 Feet	Latchups	Latchups/Hr.
LCA100K	1994	$5.56*10^6$	1	$1.8*10^{-7}$
LCA100K	1992	$2.46*10^7$	19	$7.9*10^{-7}$
LCA200K	1994	$2.6*10^7$	1	$3.9*10^{-8}$
LCA200K	1992	$5.6*10^6$	7	$1.21*10^{-7}$

case latchup rates due to neutrons and protons are estimated to be in the range of 10^{-6} to 10^{-7} latch-ups/device-h [79]. Table 2 lists measured average rates from neutron beam tests at the Weapons Neutron Research (WNR) facility, showing a range of $1.2*10^{-7}$ to $3.9*10^{-8}$ latchups/flight hour, for a beam strength range similar to upper atmosphere levels [81].

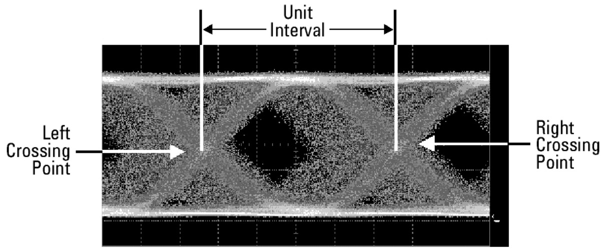
2.4.3 Bit Error Rate

A Bit Error Rate (BER), or bit error ratio, is a measure of a communication link's performance, stated as "…the ratio of the number of bits received in error to the total number of bits transmitted" [85]. The value of a bit is determined by the value(s) of the incoming signal sampled by the receiver at a particular point or points in time. However, due to noise and timing deviations, the received bit value might not equal the bit value that was intended to be sent. A logic 1 might be perceived as a logic 0, and vice-versa. This constitutes a bit error. The probability of a bit error occurring is a property of the sender, the receiver, and the communication link between them. This section examines bit errors from independent sampling events, so the faults modeled will be spatially and temporally independent. The bit error rate is also assumed to be constant for a given sender-receiver-link set. Bursts of noise that might cause a number of samples to be corrupted for a period of time are discussed under Electromagnetic Interference.

The window of time where the receiver is expected to obtain a correct bit value can be described with an eye diagram. "An eye diagram is a composite of all the bit periods of the captured bits superimposed on each other relative to a bit clock (recovered or available from the source). We call the area within the eye the eye opening" [85]. Figure 8(a), from



(a) Idealized Eye Diagram, from [3]



(b) Eye Diagram With Jitter, from [3]

Figure 8. Eye Diagrams, Bit Error Rate

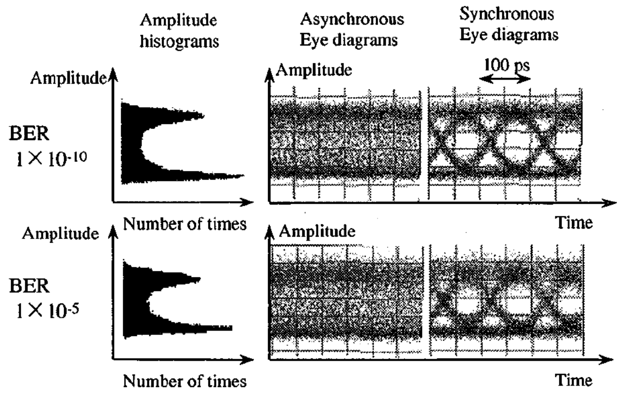
[3], shows an idealized eye diagram for a transmission scheme with a low value logic 0 and a high value logic 1 (other encoding schemes are possible). The sampling point represents the point in time least likely to result in a bit error, with values below the threshold classified as logic 0 and values above classified as logic 1 [3]. A signal that crosses the eye opening is violating the specification [85]. Generally, a larger eye opening corresponds to a lower BER. Timing deviations (modeled as jitter) and poor signal strength can affect both the horizontal width and vertical height of the eye opening. In most cases, careful design should lead to a low bit error rate. The next sections discuss the impact of jitter and signal strength on the BER.

Jitter

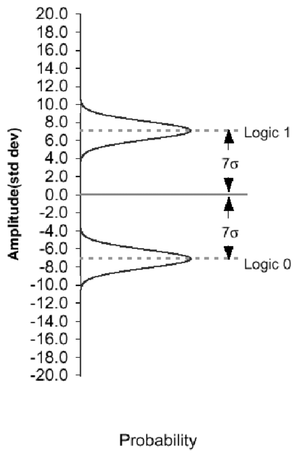
"Jitter is the deviation of a signal's timing event from its intended (ideal) occurrence in time ..." [85]. Both random and deterministic sources of jitter exist, and are usually modeled separately. Sources of random jitter include thermal noise, shot noise (electron and hole noise in a semiconductor), and flicker (pink) noise [3]. Random jitter is assumed to follow a Gaussian distribution, and is therefore unbounded, since the tails of a Gaussian distribution are infinite [110]. Deterministic jitter is mainly caused by "...electromagnetic interference, crosstalk, signal reflection, driver slew rate, skin effects, and dielectric loss" [85]. Certain forms of jitter are data-dependent, including intersymbol interference and effects that depend on the frequency of the transmission, such as attenuation due to skin effect and dielectric loss (both worse at higher frequencies) [85]. Unlike random jitter, deterministic jitter is bounded, since it reaches maximum and minimum deviation values within a bounded period of time [3]. The total jitter is the sum of the random jitter and deterministic jitter (or, if described as a probability density function, the total jitter PDF is equal to the convolution of the random jitter and determinisitic jitter components' PDFs) [85]. A narrower eye opening can be seen in the Figure 8(b) eye diagram from [3], from a signal experiencing jitter. In general, direct BER measurement is possible using a Bit Error Rate Tester scan down to at least a BER of 10^{-12} [110], so systems considered here should have measurable BERs. For extremely low BERs, extrapolating the tails of the scan may not be accurate [110].

Signal Strength and Noise

In addition to jitter, amplitude disturbances can have a significant impact on the BER. A signal must have adequate power for the receiver to distinguish the signal from noise. Designers often use the highest acceptable BER to determine the sensitivity needed for the receiver. "Sensitivity is a measure of how weak a signal can get before the bit-error ratio (BER) exceeds some specified number" [73]. Sensitivity is often expressed as the average power at which the maximum (worst allowable) BER can be sustained [73]. The amount of noise present in a system relative to the signal strength is often expressed as a signal-to-noise ratio (SNR), where a high SNR means that the signal is strong relative to the noise present. At a low signal-to-noise ratio, there is a greater chance of the receiver interpreting noise as a valid value. Shake et al. give an example of the relationship between signal-to-noise ratio and BER, shown in Figure 9(a) [100]. The upper set of diagrams shows the original system with a high signal-to-noise ratio, with a low standard deviation amplitude and a fairly clean signal sampling space in the eye diagram. In the lower set of diagrams, noise was injected via an optical amplifier, resulting in a lower signal-to-noise ratio. The amplitude histogram shows increased standard deviation, and the signal sampling window in the eye diagram is less clean [100]. This particular example tested



(a) Influence of Noise on BER, Optical Example, from [100]



(b) BER and Standard Deviation of Random Noise, from [73]

Figure 9. Noise and Bit Error Rate

a 10-Gb/s non-return-to-zero (NRZ) optical signal. Signal strength generally attenuates proportional to the distance the signal travels, so the power of the source signal must be chosen with respect to the most distant receiver. There are many causes of amplitude distortion, and the type of noise experienced depends on the medium. Dispersion, where the original pulses broaden as they travel through the medium, is one cause of signal distortion [125]. Electromagnetic interference can affect the links, sending devices and receiving devices. Copper wire can also be a source of electromagnetic radiation. Similar studies have been performed for copper; for example, Stephens et al. measured the BER of twisted pair copper cable connections of approximately 100-200 meters in a feasibility study for SONET/ATM frames over copper [111]. As the amount of injected noise increased, the SNR decreased and the BER increased.

BER Data

One way to design a system is to set a maximum allowable worst-case BER first, and design the system to meet that goal. For random noise, a 7σ separation between sampling point (in time) and sampling threshold (in value) is recommended to achieve a BER of

10^{-12}, for an interval of 14σ total [3], [73]. σ is the standard deviation of the (Gaussian) random noise distribution. If the left and right (and/or upper and lower) distributions are different, an average or approximate σ may be needed [3]. Figure 9(b) from [73] shows the 7σ separation for amplitude distrubances. Sources of deterministic noise need to be determined separately, and their noise contributions added to the random noise contributions. However, identification is usually more challenging than designing the tolerance strategy, since deterministic noise is bounded. Higher BERs are expected in harsher environments, for longer cable lengths, and for higher bandwidth connections (since there will be less energy per bit sent). Since the BER is influenced by many parameters, a wide range of BERs is possible, and research often optimizes the BER given some fixed parameters, or optimizes another parameter given a fixed BER.

Designs may be required to meet a BER specified in a standard. Generally, optical network standards mandate a lower BER than standards for copper wire. Also, more information on optical standards is available. Table 3 summarizes BER requirements from some common standards. The Boeing ARINC 636 standard and Fiber Distributed Data Interface (FDDI) standard both stipulate a maximum BER of $2.5 * 10^{-10}$ [17]. Gigabit Ethernet requires a maximum BER of 10^{-12}, and some of the Synchronous Optical Network (SONET) configurations require a maximum BER of 10^{-10} [73]. The 10 Gigabit Ethernet specification includes a set of stressed eye receiver sensitivity tests, to ensure that a receiver can operate at a 10^{-12} BER or better [110]. For copper, the draft IEEE802.3ah standard specifies requirements for copper Ethernet networks aimed at connecting individual homes and businesses to higher-speed networks ("Ethernet in the First Mile"). This standard states that one of the objectives for 2BASE-TL/2PASS-TL and 10PASS-TS is "To provide a communication channel with a mean bit error rate of less than one part in 10^7 with a 6dB noise margin . . ." [51]. For electric railway equipment, the Train Communication Network (TCN) conformance tests specify that no more than 3 frame errors are allowed in $3 * 10^6$ frames. For small frame sizes expected (frames are usually tens of bits long), this

Table 3. Bit Error Rate in Standards

Standard	Medium	Max BER	Ref.
Boeing ARINC 636	Optical	$2.5 * 10^{-10}$	[17]
FDDI	Optical	$2.5 * 10^{-10}$	[17]
Gigabit Ethernet	Optical	10^{-12}	[73], [110]
SONET	Optical	10^{-10}	[73]
IEEE802.3ah Draft	Copper	10^{-7}	[51]
Train Communication Network	Copper, Optical	10^{-6} to 10^{-7}	[53]

would correspond to a BER of about $1*10^{-6}$ to about $1*10^{-7}$ if the chance of each bit being corrupted is independent, since the chance of two corrupted bits in one frame would also be small. An Allen-Bradley installation of the copper Controller Area Network (CAN) for a robotic welding application had an average measured BER of $3.84 * 10^{-7}$, although the CAN standard does not specifically state a maximum BER.

2.4.4 Electromagnetic Interference

Electromagnetic interference (EMI) refers to internal and external disruptions that can affect a node's ability to operate or corrupt messages on a network. Electromagnetic interference generally causes correlated errors, either in space or in time. This work concentrates on disturbances induced by lightning. The United States Federal Aviation Administration estimates the frequency of lightning strikes for commercial airplanes to be one strike per 2500 operating hours [32]. The nature of electromagnetic interference from lightning is described in the RTCA DO-160D Environmental Conditions and Test Procedures for Airborne Equipment [94]. For example, Figure 10 from [10] shows pulse 5 from the RTCA DO-160D lightning tests.

Electromagnetic Compatibility (EMC) standards are the most common source of information on burst faults. Both aviation and automotive have burst testing standards. For aviation, RTCA DO-160D is the standard required by the Federal Aviation Administration (it includes other aircraft testing procedures as well) [94]. The ISO-7637 family and the IEC 61000-4-4 standards both describe electromagnetic test pulses for automotive compo-

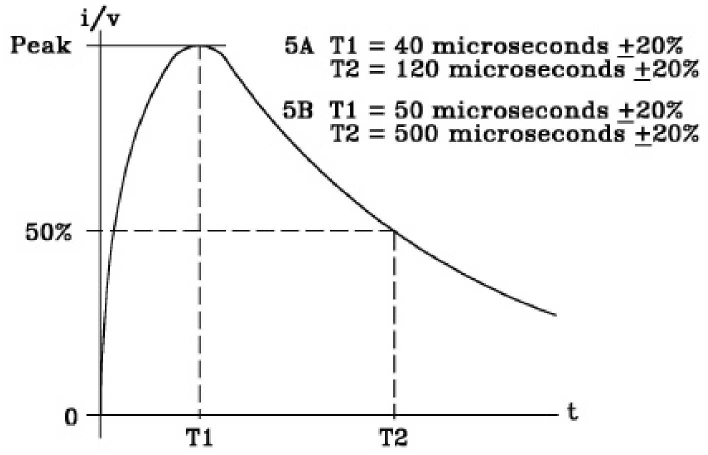


Figure 10. Waveform 5 from RTCA DO-160D Lightning Tests [10]

nents [54], [52]. Suppliers may have their own internal testing criteria, which are often based off of a standard but may include extra or slightly different tests. For example, Ford Motor Company requires its suppliers to meet a custom standard [37].

With respect to this methodology, one of the most important properties of a transient burst fault is the burst duration. The burst duration indicates approximately how many network messages would be lost, which determines the impact on the group membership service. Long duration bursts might even affect frames for an entire round. Bursts of 10ms duration are not uncommon - the IEC 61000-4-4 tests have some bursts with 15ms duration [64], and one of the Ford EMC test pulses represents a 150ms power interruption [37].

Lightning is modeled here as having a burst duration of 5ms, or half of a round. This value is in between the single stroke and multiple stroke test durations. The RTCA DO-160D standard specifies three different types of tests: single stroke, multiple stroke, and multiple burst. There are five different single stroke lightning waveforms in RTCA DO-160D applicable to cable bundle tests [10]. The single stroke waveforms are designed to stress the system with a high voltage or high current pulse with a short rise time. Each waveform has five voltage or current levels. The longest duration single stroke waveform would take about 0.5ms to decay down to 1 Volt at the lowest voltage level and about 1ms at the highest voltage level [10]. The multiple stroke test has a duration of 10ms to 200ms, with a primary stroke followed by thirteen secondary strokes within this timeframe [71], [10]. Finally, the multiple burst test has three bursts of up to 1ms within a 30m to 300ms timeframe [10].

3 Methodology

This chapter presents the assumption reliability measurement methodology. There are four main activities in the methodology. For each activity, a certain amount of reuse is possible. The four activities are listed below:

1. Define physical fault model (once per domain)

2. Map physical fault model to hybrid fault model (once per protocol/domain combination)

3. Define states and transitions for reliability model (once per protocol/domain/design combination)

4. Generate and solve reliability models for entire design space (once per configuration)

For the physical fault model, a domain is the environment in which a system (here, the network protocol) will be used. A domain may include certain quality factors regarding the components. For example, in the aviation domain a protocol will be deployed on aircraft and will be subjected to physical faults arising from atmospheric disturbances. As an example of a quality factor, optical fiber typically has a lower Bit Error Rate than copper. Optical fiber is less susceptible to electromagnetic disturbances than copper wire; however, optical fiber is more expensive. Once a physical fault model is defined for a domain, it can be reused to test other specification assumptions for systems in the same domain. Also, one contribution of this dissertation is grouping physical faults according to four main categories. While the physical fault sources may change for different domains, each of these four categories is likely to be represented.

Each protocol defines a hybrid fault model, which the formal proof (if used) is based upon. For the clock synchronization and group membership guarantees studied, the hybrid fault model describes faults in terms of a group of observers. Each physical fault type needs to mapped to at least one of the hybrid fault types. For example, the group membership hybrid fault models define three categories of faults: asymmetric, symmetric and benign. Each physical fault is assigned to one or more of these categories, and is either a permanent fault or a transient fault (with more detailed definitions discussed later).

Once this is done, the states and transitions for the reliability model can be specified, which may involve specifying additional system parameters and design decisions. System parameters include items such as the network bandwidth, chip sizes (for faults measured in errors/bit instead of errors/device), number of nodes and number of channels. For design decisions, this work focuses on fault diagnosis and fault tolerance strategies. The fault diagnosis strategy will have some probability of making the wrong diagnosis under various circumstances, and will take a certain length of time to diagnose a faulty node. A script defining the states and transitions is written for the NASA ASSIST tool, which will generate the complete Markov model state space from the script and variable fault rates and system parameters. While the exact script will be unique to each protocol/domain/design set, there is a lot of opportunity for reuse across design decisions and domains. For example, the membership models reused about half of the states and transitions from the clock synchronization models. The states and transitions can be reused if the physical fault rates change but the physical fault types and mappings stay the same.

Step four has the least reuse, but can be automated to a large extent. Once the ASSIST script with the states and transitions is defined, shell scripts can automate model generation and execution for each combination of fault rates and variable system parameters (e.g., the number of nodes). Here, the models were solved using the NASA Semi-markov Unreliability Range Evaluator (SURE) tool. As with any Markov modeling tool, the designer will need to take some care to limit the state space size. The Markov models are solved more quickly if the designer specifies pruning information, although the tools will try to automatically tune this parameter. However, for a large batch of models manual specification of some pruning may be required so that the total solution time is feasible. Also, there are cases where the NASA SURE tool cannot produce a solution, such as if the model is pruned too severely from pruning parameters that are too aggressive, or in cases with high fault arrival rates that create paths with a large number of loops.

First, this section reviews some assumptions of the methodology itself. Next, the physical fault model terms and hybrid fault model terms are introduced. The modeling process is described next, with a small example. Then, the model templates for each of the three protocols are introduced and discussed.

3.1 Methodology Assumptions

Like any methodology, the methodology I present makes some assumptions about the faults, the construction of the system, and the response of the system to faults. This section reviews those assumptions, and the reasons why these assumptions are made. It is important to note that for the most part, these assumptions come from the domain or from theoretical bounds — they were not put in place to make modeling 'easy'. The main exception is that some of the burst faults are difficult to represent with this methodology, since a burst fault could immediately violate the maximum fault assumption. In these cases, a reliability metric will be bound by the fault arrival rate of that burst fault, and an availability metric would be more appropriate. A reliability metric still can be helpful to investigate how long of a duration a burst fault can be before having extremely negative effects on the system, since the system designer has some control over the duration of the diagnosis rounds and could customize this parameter to the burst fault length.

A Time Division Multiple Access schedule allows the designer to make many useful assumptions. The motivation for using a static TDMA schedules is to enable implementation of a bus guardian to protect against 'babbling idiot' faults, where a node attempts to broadcast on the bus all the time. In priority-based sending schemes, such as the Controller Area Network, a high-priority babbling idiot node could be a single point of failure since the highest priority node always wins control of the bus. Having a static schedule also makes it more tractable to ensure that all messages will arrive by their deadlines, especially for systems with short periodic messages that have short deadlines (for example, 10ms or so). TDMA schemes can also support higher bandwidth, since there is no arbitration over which node controls the bus.

Synchrony. A TDMA schedule satisfies both synchrony requirements - there is an upper bound on the message transmission time (due to the slot window) and an upper bound on the amount of time to send a message (since each node typically sends once per round). Both upper bounds are known at design time. Since the system conforms to the synchrony assumptions, it is possible to construct deterministic fault-tolerant membership services. Removing the synchrony assumption could make it infeasible to develop a practical membership service. It is impossible to guarantee agreement using a deterministic algorithm for asynchronous systems in the presence of faults [36].

Signed Messages Solution Not Applicable. With regards to theoretical results, the membership services for these protocols cannot be assumed to have unforgeable messages, so the signed messages solution of the Byzantine Generals problem cannot be used [66]. In TTP/C, nodes exchange local membership vectors which reflect the status of other nodes in the system. A node might change its membership vector in any manner. For example, the vector might be changed so that it appears that another node did not successfully send a frame even though it did. In SPIDER, the Redundancy Management Units can generate a new frame instead of passing on the frame from the Bus Interface Unit sender. A service that does conform to the signed messages solution might have better assumption reliability since it can tolerate more faults with fewer components.

Lower Bound on Rounds for Interactive Consistency. While the exact assumptions differ slightly, the membership services are also influenced by the f_{round} + 1 to tolerate f Byzantine faulty rounds from [35]. TTP/C and SPIDER both essentially have two rounds. In TTP/C the diagnosis service is based on TDMA rounds, and in SPIDER the BIU to RMU frame transmissions form the first round and the RMU to BIU frame transmissions form the second round. A service that has more rounds might have better assumption reliability since it could tolerate more faulty rounds.

Reintegration Not Considered. At this time, reintegration is not considered, although this methodology could help guide a reintegration strategy. The goal of the methodology is to point out problems areas (for example, too many convicted transient faulty nodes), which the designer can address through the fault tolerance strategy or the reintegration strategy. Each of the protocols is developing a reintegration strategy, but full formal proofs are not available yet. One complication is that reintegrating a faulty node could decrease the reliability of the system. There is a relationship between fault diagnosis and reintegration — the reintegration strategy will be determined somewhat by the accuracy of the fault diagnosis, and the aggressiveness of fault removal will be determined somewhat by the ease of reintegration. For example, an aggressive fault diagnosis and removal strategy paired with a quick reintegration strategy might have similar reliability to a lenient fault diagnosis and removal strategy (if transient faults are present) and a slow reintegration strategy.

Set of Participants Known, No Partitions. A number of assumptions are made about the nodes in the system and their communication patterns. Since the schedule is statically determined ahead of time, the set of all possible participants is known. (Some of the protocols support schedule changes during operation, with schedule propagation algorithms, but even then there would be the same set of physical nodes connected to the network). The protocols do not allow two networks to merge, and do not handle network partitions (except with worst-case strategies such as system restart). Also, since dual redundant network cables are used, the system will still operate correctly if one network cable is partitioned. All frames are broadcast to all receivers on the network.

Limited Fault Correlations. Faults are assumed to be uncorrelated except for faults due to electromagnetic interference (lightning). The other fault categories - Permanent Hardware Faults, Single Event Effects, and Bit Error Rate - conform fairly well to this assumption. However, for any fault type there is always the possibility of fault correlation.

Designers attempt to detect and remove as many correlated faults as possible (often called 'common mode faults' or 'common mode failures'). For example, a power supply can easily become a common mode fault source if the same power supply supports components in two separate fault containment regions. Design rules are put in place to prevent this from happening, for example by requiring redundant power supplies. There are also techniques for managing both temporal and spatial correlation. For example, TTP/C allows a sender to broadcast a frame on channel A at one time slot and on channel B in another time slot (both time slots are within the same round). For spatial correlation, redundant components can be located in different areas of the system. The methodology can represent some types of fault correlation (as long as the maximum fault assumption is not violated immediately), so more types of correlated faults could be modeled. While the modeling tools are best suited for exponential rates, the SURE tool does have limited support for other types of distributions.

Constant Fault Arrival Rates. Fault arrival rates are assumed to be constant throughout the mission. However, this may not be true in practice. For example, the Single Event Latchup and Single Event Upset rates are a property of atmospheric depth which changes during an airplane flight. The lightning rate might also change, for example if an aircraft flies into a storm. The wide range of fault rates studied helps compensate for this somewhat. If the fault rate varies in practice, but is still within the fault rate range tested, then at least the range of the assumption reliability is known. Or, a designer could model the system using conservative worst-case fault rates. If the models are not sensitive to that particular fault, this may be a non-issue. Usually one or two of the fault types are dominant, with the models insensitive to changes in the other fault rates. Phased models could also be used, which are discussed further in the Extensibility section.

3.2 Physical Fault Model

I study four sources of physical faults: permanent hardware faults, single event effects, bit error rate, and electromagnetic interference. This section presents a short synopsis of the more detailed coverage of the physical fault model in the Related Work chapter. This physical fault model is a significant improvement over the traditional fail-silent model, as I discuss in the Results chapter. The fault rate data studied is primarily from the aviation domain, but these physical fault sources are common to other wired embedded systems such as automotive systems. The primary differences between automotive systems and aviation systems are that automotive systems have a higher bit error rate (since automotive systems use less expensive copper cable) and have a lower incidence rate of single event effects and lightning (since radiation levels and lightning strike rates are lower on the ground than in the atmosphere). Also, it is important to subject configurations to multiple physical faults at the same time, since the total failure rate can be worse than linear product of each of the individual physical fault occurrence rates. While this physical fault model covers many faults, there may be additional faults that a system designer wishes to protect against.

The physical fault types and rates are summarized in Table 4. For permanent hardware faults, I use a fault rate of 10^{-5}/hr for a node (larger fault containment region) and 10^{-6} for a star coupler or bus (smaller fault containment region) [119]. The tested link fault range is 10^{-8}/hr to 10^{-6}/hr, which is slightly conservative compared to [119] but slightly optimistic compared to [49]. These permanent fault rates apply to both the aviation and automotive domains.

The data reported in the Results section uses the fault rates for the aviation domain. The single event effects class includes radiation faults due to particle collisions. Single Event Latchup (SEL) is the dominant permanent effect [99], with observed SEL rates from about 10^{-8} to 10^{-6} latchups/device-hr [79]. Single Event Upset (SEU) is the most prevalent transient effect [27], with measured SEU rates ranging from $1*10^{-8}$ to $4*10^{-10}$ upsets/bit-hr [79]. The bit error rate class includes faults caused by jitter and amplitude disturbances on the network. Three optical standards give worst-case BERs ranging from 10^{-12} to 10^{-10} [17], [73], [110]; this work studies a less pessimistic range of 10^{-13} to 10^{-11}. The fourth class, electromagnetic interference, includes correlated burst errors [94], [52], [54]. This work focuses lightning strikes, estimated at one strike per 2500 flight hours [32].

For the automotive domain, the same fault classes apply, but some of the fault rates will be different. In automotive systems, the Bit Error Rate is higher since copper cable is typically used instead of optical fiber. Some copper standards stipulate maximum BERs of 10^{-7} (copper Ethernet) and about 10^{-6} to 10^{-7} (Train Communication Network) [51], [53]. Observed BER data for a Controller Area Network installation in a robotic welding application cites a BER of $3.84 * 10^{-7}$ [2].

Radiation levels due to neutron flux are lower on the ground than in the upper atmosphere. However, radiation due to alpha particles from contamination of the circuit packaging will remain the same. Right now, neutron cosmic ray effects still dominate alpha particle effects even at ground level [83], but faults due to alpha particles may increase as circuit geometries shrink [22]. O'Gorman states that "…even at sea level, cosmic rays were responsible for a significant fraction of the observed error rate …[83]". Normand reports measured average ground level SEU rates in the range of $2.1*10^{-12}$ to $3.1*10^{-13}$ upsets/bit hr, which correspond well with neutron beam testing rates [80]. The SEU range of 10^{-12} to 10^{-13} is used in Table 4 for automotive. This SEU rate range is about three times lower than the range for aviation (10^{-8} to 10^{-10}) upsets/bit hr. Since Single Event Latchup arises from the same radiation sources, for automotive a rate range of 10^{-10} to 10^{-11} latchups/device hr is used (compared to 10^{-6} to 10^{-8} latchups/device hr for aviation).

While there is some data for the lightning strike rate for airplanes, there is not as much data available for the automotive domain. However, there is data available on the number of people struck by lightning every year. The chance of a person being struck by lightning is about 1 in 700,000 per year, based on reported injuries and fatalities [126]. This gives a rate of about $1.6*10^{-10}$ per hour. For automotive, the rate will probably be higher since

Table 4. Physical Faults and Rates

Physical Fault Type	Rates, Aviation Profile	Rates, Automotive Profile
Perm. Node [119]	10^{-5} faults/hr	
Perm. Bus/Star [119]	10^{-6} faults/hr	
Perm. Link [119], [49]	10^{-8}, 10^{-7}, 10^{-6} faults/hr	
SEL [79], [80]	10^{-8}, 10^{-7}, 10^{-6} latchups/device-hr	10^{-11}, 10^{-12} latchups/device-hr
SEU [79], [27], [80]	10^{-10}, 10^{-9}, 10^{-8} upsets/bit-hr	10^{-13}, 10^{-12} upsets/bit-hr
BER [17], [73], [110]	10^{-13}, 10^{-12}, 10^{-11} err/bit	10^{-8},10^{-7}, 10^{-6} err/bit
Lightning [32], [126]	$4*10^{-4}$ strikes/hr	$1*10^{-9}$ strikes/hr

people use automobiles in adverse weather more often than people stand outside in adverse weather, and lightning strikes may be underreported. Table 4 uses a rate of $1*10^{-9}$ for automotive. There is also data on lightning flash density (the number of lightning flashes per area per year). Huffines and Orville show a range of between almost zero flashes / km^2 year to 11 flashes / km^2 year [48] for the United States, depending on geographic location. The chance of lightning striking a vehicle could be estimated from the flash density and the area around the vehicle that lightning would need to strike in order for the car to be affected.

3.3 Hybrid Fault Model

A hybrid fault model classifies faulty nodes according to fault severity with respect to a group of observers. The maximum fault assumption for a service is stated in terms of the fault types in the hybrid fault model. The original Byzantine fault model placed no restrictions on the behavior of a faulty node, thereby covering all possible faulty behaviors and requiring $3n + 1$ processors to tolerate n faulty processors [66]. However, many less severe faults are provably easier to tolerate. Meyer and Pradhan divide faults into two categories, Non-Malicious (faults that are self-evident to all receivers) and Malicious (all other faults, including Byzantine) [74], [6]. Thambidurai and Park introduced a three category fault model including Byzantine (also called asymmetric), symmetric, and benign faults

[115]. Azadmanesh and Kieckhafer introduced the strictly omissive fault category, including strictly omissive symmetric and strictly omissive asymmetric [6]. Since definitions vary, I use the definitions from the NASA Langley Scalable Processor Independent Design for Electromagnetic Resilience (SPIDER) protocols, a set of formally proven safety-critical network protocols [75]. The strictly omissive asymmetric category is from Azadmanesh and Kieckhafer [6].

- *Good (G)* "Each good node behaves according to specification; that is, it always sends valid messages" [75].
- *Benign (B)* "Each benign faulty node either sends detectably incorrect messages to every receiver, or sends valid messages to every receiver" [75].
- *Symmetric (S)* "A symmetric faulty node may send arbitrary messages, but each receiver receives the same message" [75].
- *Asymmetric (A)* "An asymmetric (Byzantine) faulty node may send arbitrary messages that may differ for the various receivers" [75].
- *Strictly Omissive Asymmetric (A)* "A *Strictly Omissive Asymmetric* fault can send a single *correct* value to some processes and no value to all other processes" [6]. A fault can "garble a message in transit, but not in an undetectable manner" [6].

To measure the reliability of a configuration, a Markov model is created with states given in terms of the hybrid fault model. The number and type of states depend on the physical fault model and the fault tolerance strategy of the specification. The designer needs to identify different components in the system and how faults might apply to those components. Transitions between states are specified with an exponential transition rate (which assumes uncorrelated fault arrivals and recoveries). An exponential transition rate is specified in the form $e^{-\lambda t}$ where λ is the transition rate per unit time, and t is time (here, in hours). A single transition may change the state of one or more nodes or channels. Correlated faults (such as lightning) are modeled as transitions that alter the state of multiple nodes or channels. With the ASSIST tool, the designer only needs to specify how a transition applies to a single component (or set of components, for correlated faults). The ASSIST tool will then generate the complete state space and transition set. Detailed descriptions of the state space and transitions are given later for each protocol. Types of transitions generally include fault arrivals, fault expiration (for transient faults), diagnosis and conviction of faulty nodes, conviction of good nodes (TTP/C), the influence of faulty components on other components (FlexRay, TTP/C), and the arrival of faults that are unable to be diagnosed (SPIDER). The next section presents an example.

3.4 Modeling Process

This section provides an overview of the modeling process, and gives a graphical example of a simple six state reliability model. Later sections will describe more details of the reliability models for each of the three services. Most of the reliability models had hundreds of states and thousands (or even millions) of transitions, so a graphical representation would be infeasible. Even the smallest models had about one hundred states and one thousand transitions, since many types of physical faults were represented.

The goal is to measure the probability that the maximum fault assumption will be violated before the end of the mission (here, an hour mission). Avizienis, Laprie, Randell, and Landwehr define the concepts of fault, error, and failure, and the relationships between these concepts [5]. A fault is “The adjudged or hypothesized cause of an error...” [5]. An error occurs when “at least one (or more) external state of the system deviates from the correct service state” [5]. A failure is “an event that occurs when the delivered service deviates from the correct service” [5].

In this methodology, a fault is the arrival of a physical fault. Each fault is covered and will cause an error, represented as the transition from a good component to a faulty component. In theory, a Byzantine fault model will cover all possible single errors. The set

Table 5. SPIDER Maximum Fault Assumption [75]

SPIDER MFA.1. $2\|G \cap E_{RMU}\| > \|E_{RMU} \backslash B\|$ for all RMUs RMU, and
SPIDER MFA.2. $2\|G \cap E_{BIU}\| > \|E_{BIU} \backslash B\|$ for all BIUs BIU, and
SPIDER MFA.3. $\|A \cap E_{RMU}\| = 0$ for all RMUs RMU, or $\|A \cap E_{BIU}\| = 0$ for all BIUs BIU.

of errors includes only errors within the scope of the protocol's fault tolerance ability. For example, a physical fault may cause an erroneous data value to be transmitted in a frame, but as long as the frame is correctly formatted this will be OK for the protocol since the protocol does not know what the data value means to the application. A failure occurs when the maximum fault assumption is violated. This is a conservative approximation of the system failure rate, since in some cases the guarantee is still provided even though the maximum fault assumption has been violated. However, for each of these protocols there are provable cases where violating the maximum fault assumption results in the guarantee not being provided.

The example presents a simple version of the SPIDER protocol. SPIDER has two types of components, Bus Interface Units (BIUs) and Redundancy Management Units (RMUs). The SPIDER maximum fault assumption (MFA) is stated in terms of the sets G, B, S, and A, which denote the sets of good, benign, symmetric, and asymmetric nodes respectively [75]. E_{BIU} and E_{RMU} refer to the sets of eligible voters for BIUs and RMUs, respectively. Table 5 lists the SPIDER maximum fault assumption. Figure 2 illustrates the Markov model for the example system configuration. This particular example is quite compact — most of the models have hundreds of states and thousands of transitions.

A designer would like to know how often the SPIDER maximum fault assumption will be violated. In the simple example, each BIU may be either good or faulty, and each RMU may be either good or faulty. Here, only transient faults are modeled, so all faults will expire after some duration. The faulty component will revert to being a good component.

3.4.1 Software Tools: ASSIST, SURE, STEM

Three Markov analysis tools developed at NASA Langley Research Center were used to construct the reliability models and to estimate the probability that each part of the maximum fault assumption would not hold. First, the designer specifies the states, transitions, and fault rate parameters in a text specification (usually under five pages). The ASSIST program translates the parameterized text specifications (in the ASSIST language) into Markov models. Then, either the STEM tool (Scaled Taylor Exponential Matrix) or the SURE tool (Semi-markov Unreliability Range Evaluator) is used to solve the Markov model. STEM provides an exact solution and is limited to pure Markov models. SURE provides upper and lower bounds on reliability, usually within five percent of each other [13], and can handle other classes of transitions besides exponential transitions. Butler and Johnson explain the underlying mathematics of these tools and give numerous fault-tolerance examples in [14]. Our models involved only exponential transitions, so either tool could be used. Note that another Markov solver could potentially be used — the approach is not limited to these three software tools. ASSIST, STEM, and SURE can be obtained from NASA Langley at: http://shemesh.larc.nasa.gov/fm/ftp/sure/sure.html

These modeling tools allow a designer to investigate a large range of possible fault arrival rates with a manageable amount of work. Since the NASA tools were developed specifically to model fault tolerant systems, they can handle ‘stiff’ models that have a set of slow transitions (here, fault arrival rates) and a set of fast transitions (here, fault expiration/recovery and conviction). Since the NASA tools use algebraic methods (exact solution for STEM and algebraic bounding methods for SURE), they can handle small probabilities well. For example, an assumption failure rate of 10^{-8} failures/hr or less is not uncommon. Tools that use iterative solution methods can experience problems with small probabilities since it may take a long time for the iterative solution to converge. Once an ASSIST script is specified, model generation and execution can be automated to a substantial extent. The designer may still need to manually adjust pruning guidelines if the model execution time is too long. Also, there are cases where SURE cannot converge on an answer (and STEM takes an unacceptable amount of time to find an exact answer), so the ASSIST script may need to be modified. More details on modeling tools can be found in the Related Work chapter.

3.4.2 State Space

A variety of configurations can be modeled with reasonable effort. The designer first specifies the possible component state space items. The two types of nodes, Bus Interface Units (BIUs) and Redundancy Management Units (RMUs), are not interchangeable. Link faults get mapped onto a BIU or an RMU, since links are not a first class entity. In this example, a node may be good or transiently faulty.

The example in Figure 11 shows transient fault arrivals and recovery only (there are no permanent faults and no convictions), so there are (2 component types) * (2 fault manifestations, good/faulty) = four items in the state space that the designer must specify. The ovals represent possible system states. For this example, the system state space S is given by the tuple {Good RMUs, Faulty RMUs, Good BIUs, Faulty BIUs} where Σ (Good RMUs, Faulty RMUs) equals the total number of RMUs and Σ (Good BIUs, Faulty BIUs) equals the total number of BIUs (i.e., components are neither created nor destroyed). Since redundant components are present, the system can tolerate some combinations of faulty components. The current system state is given inside each oval, listing the number of working and faulty components.

The example system in Figure 11 has three RMUs and four BIUs. In the start state at the top, all nodes are working, so that state space is (3, 0, 4, 0). Combinations of faulty nodes are listed in other ovals, for example, the state space (3, 0, 3, 1) represents three good RMUs, three good BIUs, and one faulty BIU and the state space (2, 1, 4, 0) represents two good RMUs, one faulty RMU and four good BIUs. There are a total of six states and eight transitions in this model. States that fail to satisfy the maximum fault assumption have been

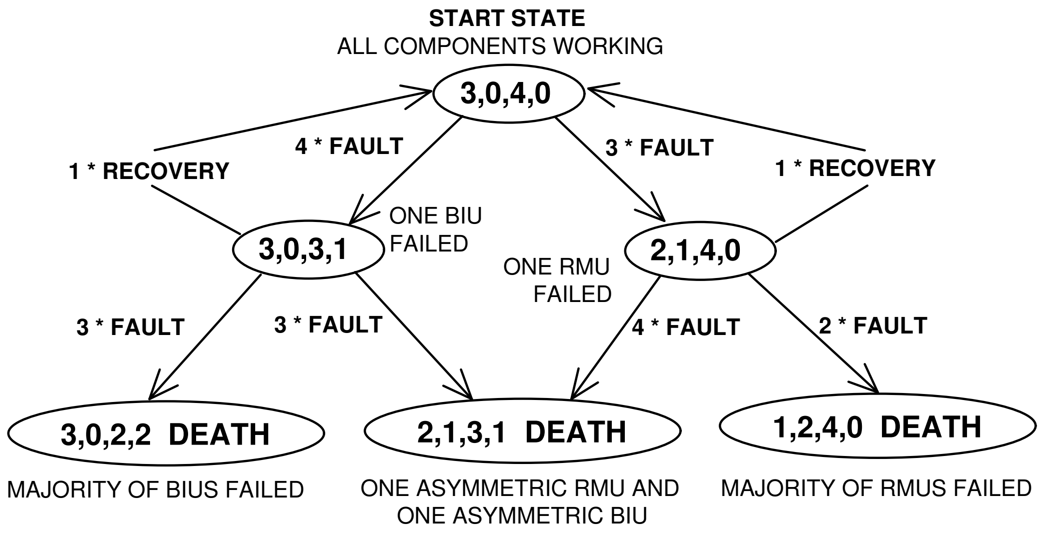


Figure 11. Markov Model Example (SPIDER)

aggregated into single states according to which piece of the maximum fault assumption has been violated first. For example, states (1, 2, 4, 0) and (1, 2, 3, 1) would both be aggregated into state (1, 2, 4, 0).

3.4.3 Transitions

Next, the designer must specify transitions. In this model, components are conserved, so a transition will take a component out of one local state and put it into another. There are three types of transitions. Fault arrival transitions transform a good component into a faulty component. Errors caused by transient faults have a finite duration, so when this duration expires a transient faulty node reverts to a good node. Finally, an asymmetric or symmetric node can be convicted, becoming a benign permanent faulty node. In this model, there is no reintegration of permanent faulty nodes, so these cannot be transformed into good nodes.

The transitions in Figure 11 occur at a rate defined by the fault arrival rate and by the number of components currently occupying a state. For example, if a fault occurs at rate F that transforms a good BIU into a faulty BIU, and there are currently 10 good BIUs, the Markov model transition rate will be 10*F. In later reliability models, there are some transitions that are independent of the number of nodes currently occupying a state. Since the transitions are modeled with exponential rates, this assumes uncorrelated, independent faults. The reliability models can express some limited forms of correlation (both temporal as in burst faults, and spatial as in multiple affected components). Correlated faults will be discussed later as well.

In Figure 11, transition rates are listed on the arrows between states. FAULT is the fault arrival rate and RECOVERY is the transient fault expiration rate (derived from the fault duration). The example uses constant rates; however, most of our models have multiple fault arrival rates according to fault type. The FAULT or RECOVERY rate is multiplied by the number of eligible components in the source state, for example, the 3*FAULT transition from the start state for the chance of an RMU becoming faulty and the 4*FAULT transition from the start state for the chance of a BIU becoming faulty.

3.4.4 Death States

Finally, the designer must specify the maximum fault assumption conditions, where the guarantees may not hold. The maximum fault assumption conditions will be given in the protocol specification and are usually derived from a formal proof, although a formal proof is not required to perform this reliability analysis. The maximum fault assumption conditions are specified as death state conditions in the ASSIST script, and are represented as death states in the model. The SPIDER example in Figure 11 has three death state conditions that map to the three parts of the SPIDER maximum fault assumption in Table 5.

The bottom of Figure 11 shows the three types of death states, labeled with the word 'DEATH'. MFA.1 assumes that a majority of BIUs are good (the leftmost death state: 3, 0, 2, 2). MFA.3 assumes that there will not be an asymmetric faulty RMU and an asymmetric faulty BIU at the same time (the middle death state: 2, 1, 3, 1). MFA.2 assumes that a majority of RMUs are good (the rightmost death state: 1, 2, 4, 0). More death state space combinations are possible - for these experiments, the combinations are aggregated

Table 6. System Parameters and Values

Parameter	Value
Bandwidth	$1*10^6$ bits/sec
Round Duration	10 ms
Frame Duration	0.1 ms
Frames/hour	$3.6*10^7$ (3600000 ms / 0.1ms)
Bits/Node	64 kilobytes
Bits, Asym. SEU	10 kilobytes
Clock Sync and TTP/C Nodes, SPIDER BIUs	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
Clock Sync, TTP/C Channels SPIDER RMUs	2 3

by the first MFA violation. Death state aggregation can substantially reduce the state space for larger models, without much loss of information, since the relevant information the designer wishes to preserve is which maximum fault assumption condition is more likely to be violated.

3.5 System Parameters and Discussion

This section covers the system parameters used, and discusses which systems would be feasible to build. In addition to the physical fault rate parameters, some system parameters need to be defined. System parameters include items such as bandwidth and round duration, and the accuracy of a membership service's diagnosis algorithm. While many combinations of parameters are modeled, some combinations may be difficult to construct actual systems for. In particular, diagnosis algorithms that accurately identify both permanent and transient faults might be challenging to develop. Like the fault rate parameters, multiple system parameter values can be tested if the designer is unsure of the parameter value in the final system. Finally, this section explains why the solution time of some models is sensitive to the number of nodes and the bit error rate.

Some system parameters were needed to specify the model transition rates. Table 6 lists the representative values assumed for these system parameters. All three protocols support

Table 7. Membership Conviction Probabilities

Probability	Strategy	Value(s)
Prob. of Convicting Permanent	Convict All	1.0
Prob. of Convicting Permanent	Convict None	0.0
Prob. of Convicting Permanent	Convict Some	0.99, 0.95, 0.90
Prob. of Convicting Transient	Convict All	1.0
Prob. of Convicting Transient	Convict None	0.0
Prob. of Convicting Transient	Convict Some	0.01, 0.05, 0.10
Prob. of Convicting Good Node (if asymmetric present)	TTP/C Convict All, Convict Some	1/Good Nodes
Prob. of Convicting Asymmetric	TTP/C Convict All, Convict Some	0.95

1 MBit/sec bandwidth, with plans to support 10 MBit/sec and possibly 25 MBit/sec [33], [118]. The round duration is determined by the shortest message period required by the system, since each node typically sends exactly once per round [118], [88]. A message period of 10 ms is representative of many embedded networks. In TTP/C a diagnosis cycle is completed every two rounds, or every 20 ms here. For SPIDER, the diagnosis period can be any integer number of rounds. To keep the parameters as equal as possible, the SPIDER diagnosis period is also set at two rounds. A frame duration of 0.1 ms would allow 100 frames of 100 bits each to be sent per second. The fault arrival rate due to SEU faults depends on the number of bits that could be affected. This work assumes 64 kilobytes, or 64*(2^{10})*8 bits. This is comparable to the size of protocol controllers. For example, the TTP-C2NF revision 1.2 chip has 40 kBytes of SRAM and 32 kBytes of ROM [4]. Sensitivity analysis is also performed for 256 kBytes. The FlexRay clock synchronization and TTP/C protocols both use two channels. SPIDER performs best on a three channel design, since receivers vote the three values received, although SPIDER can run with only two channels.

A set of conviction parameters is used for the membership services for TTP/C and SPIDER. Table 7 lists the conviction probabilities modeled, which correspond to different conviction strategies and can depend on the way that a protocol is designed. A diagnosis algorithm can attempt to remove faulty nodes from the group, according to some conviction strategy. The diagnosis algorithm may try to distinguish between permanent and transient faults, although perfect discrimination is not possible so there will be some misclassification [68].

There are three conviction strategies studied (for both TTP/C and SPIDER). The standard Convict All strategy removes a node after a single fault, the Convict None strategy removes no nodes, and the Convict Some strategy attempts to remove permanent faulty nodes and let transient faults expire. For the two strategies with conviction, the Convict All strategy is modeled with a Probability of Convicting Permanent of 1 and the Convict Some strategy uses a range of 0.99, 0.95, and 0.90. The Convict All strategy has a Probability of Convicting Transient of 1, and the Convict Some strategy uses the range 0.01, 0.05, and 0.10. Both strategies have special treatment for asymmetric faults. If an asymmetric fault occurs, the group might not be able to identify the sender. TTP/C and SPIDER differ slightly in their treatment of asymmetric nodes. For TTP/C, the probability of asymmetric permanent node conviction is multiplied by 0.95 for the Convict All and Convict Some strategies. In TTP/C, good nodes may be convicted as long as an asymmetric fault is present (since there may be good nodes in the minority clique), represented by the Probability of Convicting a Good Node in Table 7. For SPIDER, the probability of convicting all types of permanent nodes is limited to 0.95 in the Convict All strategy and 0.99, 0.95, and 0.90 are studied for the Convict Some strategy. SPIDER specifically prohibits the conviction of good nodes, with the consequence that some faulty nodes might never be convicted. (It is impossible to guarantee both eventual faulty node conviction and no good node conviction in the presence of asymmetric faults [102].) For SPIDER, special sink states are defined for undiagnosable nodes, and these nodes will never be convicted. An asymmetric faulty node might never be convicted in TTP/C as well, but the TTP/C protocol is not subject to the impossibility restriction since TTP/C allows good node conviction. For the Convict None strategy, all probabilities of conviction are 0.

While different conviction probabilities are studied, it is not known at the design stage whether it is possible to implement these misclassification rates. The reliability models only require a probability number. At this stage, the designer does not need to have developed a strategy that actually achieves this probability. For a protocol, a frame is the smallest indivisible item that may be faulty. An ideal permanent fault will last for an infinite number of frames, and an ideal transient fault will last for exactly one frame. In reality, a fault might persist for an arbitrary number of frames. The success of the diagnosis algorithm will depend on how well permanent faults and transient faults conform to these ideals. For the Convict Some strategy, missing a permanent fault can be thought of as a false negative, while convicting a transient fault can be thought of as a false positive. In many systems, lowering the false positive rate may increase the false negative rate, and vice-versa. For the protocols, the false positive and false negative rates will depend on how much the durations of permanent and transient faults overlap. Also, some types of faults do not fit neatly into the permanent/transient classification scheme. An intermittent faulty node might cause single frame corruptions, but at a much higher fault arrival frequency. Here, intermittent faulty nodes are treated as permanent faulty, but in an actual system this would be a design decision. Nodes might also accumulate internal latent faults, which are activated some time later and corrupt a frame. For errors due to latent faults, the frame corruption rate increases with time. The goal of this research is to examine the sensitivity of the models to the false positive and false negative rates, rather than forecasting a precise assumption reliability number. This would help a designer choose whether to try lower the false positive rate (where false negatives may increase), or try lower the false negative rate (where false positives may increase).

Another issue concerns what conviction strategies could actually be implemented without any proof changes. Currently, the TTP/C diagnosis algorithm proofs are integrated with the membership service proofs. Therefore, any changes in the diagnosis service might require changes in the proofs. The proposed Convict None and Convict Some strategies might require proof changes, and it is difficult to estimate the amount of work required for a proof change. Alternatively, the reintegration algorithm could be changed instead of the diagnosis algorithm. Since the reintegration algorithm is still being developed, this might require less work. For SPIDER, the diagnosis algorithm proofs are largely separate from the membership proofs, except for the *correctness* restriction that no good nodes may be convicted [117]. Therefore any of the three conviction strategies should be able to be implemented without proof changes, as long as the new strategy conforms to the correctness restriction.

When the value of a parameter is uncertain, different parameter values can be modeled to determine the sensitivity to changing the parameter. For example, radiation can induce asymmetric single event upsets, but the amount of single event upsets that will be asymmetric compared to benign or symmetric may be unknown. One way to address this issue is to study different values for the percentage of faults that will be asymmetric. Parameters that the models are sensitive to can be studied further with fault injection. For example, an early version of the TTP/C controller was subjected to both software fault injection and heavy ion radiation in [1]. The observed errors were categorized, with about 0.4% classified as slightly-off-specification for the bus topology, and no slightly-off-specification errors observed in the newer star topology [1].

Finally, there may be parameters values that would be interesting to test, but are beyond the capabilities of the modeling tool. For this work, the number of nodes and the bit error rate needed to be limited. Keeping a reasonable state space size is important for any modeling tool. Here, adding another node roughly doubles the state space (slightly less since the death states are aggregated). This can roughly double the model solution time. Because of this non-polynomial time scaling, only models with up to fourteen nodes were tested. However, the SURE tool successfully handled models with over one hundred thousand states and over two million transitions. The second concern for Markov modeling tools is that it is difficult to solve models where transition rates can differ by many orders of magnitude. Typically, fault arrival rates are low (on the order of 10^{-6} faults/hr or so), and fault recovery rates are high (on the order of 10^{6} recoveries/hr or so). Models with a group of fast transitions and a group of slow transitions are sometimes called 'stiff' models. The SURE tool handles most stiff models well, and had no problems modeling the fault rate ranges for three of the four types of faults. However, solution time increases significantly for some models with higher bit error rates, as the bit error rate becomes a mid-range transition instead of a slow transition. Since the bit error rate is multiplied by the bandwidth (1 MBit/second) and converted to hours (3600 seconds/hour), a bit error rate of 10^{-11} represents a fault arrival rate of about 10^{-2} faults/hr. For example, some SPIDER configurations took about ten seconds to solve at a bit error rate of 10^{-12}, but these configurations took over an hour to solve at a bit error rate of 10^{-11}.

3.6 Clock Synchronization

The reliability of a protocol depends in part on how the protocol's hybrid fault model classifies faults. I demonstrate this by comparing the Welch and Lynch clock synchronization algorithm to the improved strictly omissive asymmetric algorithm by Azadmanesh and Kieckhafer [6].

The FlexRay clock synchronization algorithm is based on the formally proven algorithm from Welch and Lynch [120], which involves an approximate agreement calculation over a set of received clock values. In the worst case, an asymmetric (Byzantine) faulty node will send a too-high value to one node and a too-low value to another. For a benign fault, the frame could arrive too early or too late at all receivers. The Welch and Lynch algorithm guarantees synchronized clocks as long as $n > 3a + b$ for n total nodes, a asymmetric faults and b benign faults. To separate failure due to too many asymmetric faults vs. failure due to inadequate redundancy, the modeled Maximum Fault Assumption (MFA) checks asymmetric and benign faults separately, as listed in Table 8. Since the modeling tools

Table 8. Clock Sync Maximum Fault Assumptions [120], [33], [6]

Clock Sync$_{WelchLynch}$ MFA.1: $n > 3a$ for n nodes and a asymmetric nodes
Clock Sync$_{WelchLynch}$ MFA.2: $n > b$ for n nodes and b benign nodes
Clock Sync$_{WelchLynch}$ MFA.3: $n > 3a + b$ for n nodes, a asymmetric nodes and b benign nodes
Clock Sync$_{Omissive}$ MFA.1: $n > \alpha$ for n nodes and α strictly omissive asymmetric nodes
Clock Sync$_{Omissive}$ MFA.2: $n > b$ for n nodes and b benign nodes
Clock Sync$_{Omissive}$ MFA.3: $n > \alpha + b$ for n nodes, α strictly omissive asymmetric nodes and b benign nodes

check death conditions in order, states that satisfy MFA.1 (for example) will be grouped together, and will not be checked further for MFA.2.

Recently, an improved bound was developed for a family of approximate agreement algorithms. Azadmanesh and Kieckhafer obtained better fault tolerance for strictly omissive asymmetric faults by enabling voting on different sized local sets [6]. The improved bound is $n > 3a + b + \alpha$, for n nodes, a asymmetric faults, b benign faults and α strictly omissive asymmetric faults. For clock synchronization, all asymmetric faults caused by non-malicious physical phenomena will be strictly omissive asymmetric, since a frame is either valid or detectably invalid. Therefore, the maximum fault assumption reduces to $n > b + \alpha$. Three maximum fault conditions were checked in order to determine the dominant cause of failure, listed in Table 8.

For TDMA clock synchronization, all asymmetric faults caused by non-malicious physical phenomena will be strictly omissive asymmetric, and the symmetric fault category is equivalent to the benign fault category. In a TDMA system, a frame's arrival time is calculated with respect to the time slot defined by the receiver's local clock. If the frame is too early or too late, it will be considered invalid. Asymmetric faults are possible since some nodes may receive a valid frame within the slot window and others might not. However, unlike an explicitly transmitted timestamp, undetected timing faults are not possible since a frame arriving within the slot window produces valid rate and offset correction values by definition, and a frame arriving outside the slot window is detectably invalid by definition.

All asymmetric faults will be strictly omissive asymmetric faults, since a frame is either valid or detectably invalid. Also, symmetric faults are equivalent to benign faults since there are no undetectably invalid frames. For a different fault model, undetected faults might be possible.

A fault may be (P) Permanent or (T) Transient. Abbreviations for the source and destination local states in the transition rate tables are (G) Good, (B) Benign, (S) Symmetric, (A) Asymmetric/Strictly Omissive Asymmetric. There are two types of components in each of the services. In clock synchronization and membership, nodes are the only first-class entities in the service. Nodes transmit frames over channels, which are not considered first-class entities. Therefore, faults on the channels may be assigned back to the node sending a frame at that point in time.

For clock synchronization and TTP/C membership, the hybrid fault model is applied to components in three ways. A node may become faulty (state with subscript N), a channel may become faulty (state with subscript C), or a node may appear faulty if both channels are simultaneously faulty (a state with subscript NC). All perceived node faults due to channel faults are transient (since if both channels are permanently faulty, the system has failed). A node can also be convicted (CONV), or permanently removed from the group by the diagnosis service. As two examples, PS_N would be a Permanent Symmetric faulty Node, and TA_{NC} would be a Node affected by Channel faults that appears to be Transient Asymmetric faulty.

For clock synchronization, the system state S is given by the tuple {G_N, PB_N, TA_N, TB_N, A_{NC}, B_{NC}, CONV, G_C, PA_C, PB_C, TA_C, TB_C}, where Σ (G_N, PB_N, TA_N, TB_N, A_{NC}, B_{NC}, CONV) equals the total number of nodes N and Σ (G_C, PA_C, PB_C, TA_C, TB_C) equals the total number of channels C. Unfortunately, a graphical representation would be prohibitive. Even the smallest clock synchronization models (four nodes) had 333 states and 2750 transitions. The largest clock synchronization models (fourteen nodes) had 24,783 states and 227,560 transitions. More information on model size and execution time is available in the

Table 9. Clock Synchronization Transitions

Source	Dest.	Items	[Guard], Main Rate Contributor	Rate Range Tested (λ), Per Hour
G_N	PB_N	1	Perm. HW	G_N*10^{-5}
G_N	PB_N	1	SEL	$G_N*(10^{-8}, 10^{-7}, 10^{-6})$
G_N	TA_N	1	SEU * Asym. Bits	$G_N*10K*8*(10^{-10}, 10^{-9}, 10^{-8})$
G_N	TB_N	1	SEU * Bits	$G_N*64K*8*(10^{-10}, 10^{-9}, 10^{-8})$
G_N	TB_N	$\lfloor N/2 \rfloor$	Lightning	$4*10^{-4}$
G_C	PA_C	1	Perm. Link (one link)	$G_C*(10^{-8}, 10^{-7}, 10^{-6})$
G_C	PB_C	1	Perm. Link (bus/star)	G_C*10^{-6}
G_C	TA_C	1	BER * Bandwidth	$G_C*10^6*3600*(10^{-13}, 10^{-12}, 10^{-11})$
G_C	TB_C	1	BER * Bandwidth	$G_C*10^6*3600*(10^{-13}, 10^{-12}, 10^{-11})$
G_N	A_{NC}	1	$[\neg(\exists\ G_C) \wedge \exists\ A_C]$, 1/Frame Dur.	$3.6*10^7$
G_N	B_{NC}	1	$[\neg(\exists\ G_C) \wedge \neg(\exists\ A_C) \wedge \exists\ B_C]$, 1/Frame Dur.	$3.6*10^7$
TA_N	G_N	1	1/Round Dur.	$TA_N*3.6*10^5$
TB_N	G_N	1	1/Round Dur.	$TB_N*3.6*10^5$
TA_C	G_C	1	1/Frame Dur.	$TA_C*3.6*10^7$
TB_C	G_C	1	1/Frame Dur.	$TB_C*3.6*10^7$
A_{NC}	G_N	1	$[\ \exists\ G_C\ ]$, 1/Round Dur.	$3.6*10^5$
B_{NC}	G_N	1	$[\ \exists\ G_C\ ]$, 1/Round Dur.	$3.6*10^5$

Results chapter.

Table 9 lists the clock synchronization model transitions. A transition moves one or more nodes or channels from a good local state to a faulty local state, or vice-versa. Some transitions involve a guard — a condition that must be true for the transition to be taken. For example, for a node to transition to a faulty state due to faulty channels, all channels must be faulty at that point in time (otherwise, at least one valid frame would be transmitted). For most transitions, each component (node or channel) has an equal and independent probability of being affected, so the rate is multiplied by the number of nodes or channels in the source state. The Table 9 transitions were determined as follows (from top to bottom).

Permanent Hardware Faults. A fail-silent node will not send any frames. This behavior is detectable by all receivers due to the TDMA schedule. Therefore, this fault is permanent benign.

SEL. Single Event Latchup may cause a node to transmit an improperly formatted frame or transmit a frame at the wrong time. Since the FlexRay clock synchronization service requires a valid frame to be both on time and correctly formatted, this fault is modeled as permanent benign.

SEU. Single Event Upset is modeled as a transient bit upset (either detected by other nodes or local error codes). If this occurs at the sending node, the effect would be benign. If this occurs in the clock synchronization logic of the receiver, this might be asymmetric, since a transient SEU might alter computation for a single frame only. (If all frames were altered, the receiver would be benign faulty since it would not be able to stay synchronized). The SEU rate is multiplied by the number of susceptible bits (here, 64 kilobytes is modeled). The SEU would have to hit a certain portion of the integrated circuit to cause the asymmetric fault described, so this is modeled as 10 kilobytes with sensitivity analysis in the results.

Lightning. Lightning is modeled as affecting half of the nodes simultaneously. These nodes are temporarily benign, recovering after the strike. In general, electromagnetic interference can have many effects, and this is a limited representation.

Permanent Link Faults. Link faults can have two effects. If a single link between a node and the bus or a node and the star coupler fails, the channel appears to be asymmetric faulty, since some nodes will receive the frame and others will not. If the entire bus or the star coupler fails silent, then the channel delivers no frames and appears to be benign faulty. A range is studied for the first case, and the second case is modeled as a permanent hardware failure at a rate of 10^{-6} failures/hour.

BER. Noise on the communication channel can also have two effects when detected. If the noise is localized near a particular receiver or group of receivers and the error detection does not catch all frame errors, the channel will appear to be asymmetric faulty, delivering different frames to different receivers. If the noise affects all receivers, the channel will appear to be benign faulty since no receivers get a valid frame. The BER is multiplied by the bandwidth and converted to hours to get the rate per hour.

Perceived Faulty Nodes due to Faulty Channels. If there are no good channels, and at least one asymmetric channel, then the sender will be perceived as asymmetric faulty since some receivers may get a valid frame and others may receive none (for example, if jitter causes the frame to be received too late at a subset of the receivers). If there are no good channels, and no asymmetric faulty channels, no valid frame will be sent to any receiver and the sender will appear benign faulty. Each time a frame is sent, one good node will be affected, for a trasition rate of 1 / Frame Duration. These transitions are not multiplied by the number of nodes in the source state since there is only one sender at a time (this also applies to transient fault expiration).

Transient Fault Expiration. All transient faults in the model eventually expire. For nodes, the effective fault duration is one message round, since a sender transmits once per round. For channels, a channel is considered good if it can send a frame. Transient channel faults (namely, bit errors) are assumed to affect a single frame, which is an appropriate model for bit errors. The effective fault duration for a transient channel fault is one frame. The transient expiration rates are stated as 1 / (duration in hours).

3.7 TTP/C Membership

The reliability of a group membership service depends on the diagnosis strategy chosen. A group membership service guarantees that all correct nodes in the group reach consensus on the members of the group within a certain period of time following a fault. The diagnosis strategy dictates which nodes should be convicted and removed from the group (if any). The diagnosis strategy must choose carefully, balancing the risk of too many active faulty nodes vs. the risk of inadequate redundancy. I investigate this tradeoff through a study of three application-level variations of the TTP/C group membership service.

The TTP/C protocol offers an integrated membership service with low overhead per frame. In the TTP/C membership service, each node keeps a local membership vector consisting of one bit per node in the system (system size is usually in the tens of nodes). Each sender incorporates its vector into the frames it sends (explicitly transmitted vectors are assumed here). If the received vector does not match the local vector, the formally proven Clique Avoidance and Implicit Acknowledgement algorithms ensure that only the majority clique survives (with tiebreaker rules) and that consensus is reached within two rounds [88], [7]. Nodes are immediately removed if suspected. If an asymmetric (Byzantine) fault occurs, there is no guarantee that the asymmetric faulty node will be removed, since it may or may not belong to the minority clique.

The maximum fault assumption I use extends the TTP/C single fault hypothesis slightly with respect to benign and symmetric faults when no asymmetric faults are present. The TTP/C group membership maximum fault assumption is that exactly one fault may occur within two rounds [88]. There are three pieces to the maximum fault assumption (MFA) I use, given in Table 10. Since the modeling techniques used do not explicitly support a notion of rounds, MFA.1 states there may not be an asymmetric faulty node and another faulty node at the same time. If there is a symmetric faulty node, all receivers increase their 'failed slots' counter [118, p. 68]. A node shuts itself down if its 'failed slots' counter exceeds half the number of current group members. Therefore, if only symmetric and benign faults are present, the system is expected to operate as long as half of the group members are good nodes (MFA.2). For benign nodes, null frames do not increase the 'failed slots' counter. The minimum fault tolerant configuration is four nodes with at least three good nodes (MFA.3) [118, p. 27]. These extensions for symmetric and benign faults have not been formally proven. However, this treatment is similar to the Thambidurai and Park hybrid fault model [115]. I believe that a single MFA condition of no two simultaneous faults would be pessimistic.

In my analysis, I examine three diagnosis strategies. The baseline diagnosis strategy is having all faulty nodes convicted and removed from the group (Convict All). The second

Table 10. TTP/C Membership Maximum Fault Assumption [118], [88]

TTP/C Membership MFA.1: If (∃ a), then $a + s + b = 1$ for a asymmetric, s symmetric, and b benign nodes
TTP/C Membership MFA.2: $s \leq g$ for s symmetric and g good nodes
TTP/C Membership MFA.3: $g \geq 3$ for g good nodes

Table 11. Membership Conviction Probabilities

Prob. of Convicting Permanent	1.0, 0.99, 0.95, 0.90
Prob. of Convicting Transient	0, 0.01, 0.05, 0.10
Prob. of Convicting Asymmetric	0.95
Prob. of Convicting Good Node [if ∃ ($A_N \vee A_{NC}$)]	$1/G_N$

diagnosis strategy is the opposite: no faulty nodes are ever convicted (Convict None). In the third strategy, the diagnosis service attempts to convict permanent faulty nodes and to leave transient faulty nodes in the group, with some misclassification (Convict Some). Note that the new strategies are not formally proven – the goal is to investigate robust application level diagnosis. For example, these could be run at the FlexRay application layer.

Alternatively, one could restate the diagnosis strategies as rapid reintegration rules. For the Convict None strategy, nodes could immediately be allowed to join the group after two rounds when consensus is reached. For the Convict Some strategy, the group could use a threshold where nodes are allowed into the group immediately after consensus has been reached, up until f faults within some time t, resulting in the node being permanently removed from the group. Since reintegration is substantially decoupled from membership, this approach should minimize any proof changes.

For membership, the system state S is given by the tuple {G_N, PS_N, PB_N, TA_N, TS_N, TB_N, A_{NC}, S_{NC}, B_{NC}, CONV, G_C, PA_C, PB_C, TA_C, TS_C}, where Σ (G_N, PS_N, PB_N, TA_N, TS_N, TB_N, A_{NC}, S_{NC}, B_{NC}, CONV) equals the total number of nodes N and Σ (G_C, PA_C, PB_C, TA_C, TS_C) equals the total number of channels C. For the Convict Some strategy, the smallest models (four nodes) had 128 states and 1121 transitions. The largest models (fourteen nodes) had 91,866 states and 1,104,902 transitions. Model size can vary by strategy (for example, in the Convict None strategy, Prob. Conv. Trans., Prob. Conv. Perm., and Prob. Conv. Good are zero so the related transitions are removed). More information on model size and execution time is available in the Results section.

Table 11 lists the range of conviction probabilities studied, and Tables 12 and 13 list the states and transitions for the hypothesized membership service. Only the states and transitions that are different from the Clock Synchronization model are reviewed here. A node that has been convicted enters the CONV sink state. For benign and symmetric faults, all receivers will correctly identify the faulty source. For asymmetric faults, an identification probability of 0.95 is assumed, and the probability of a good node being convicted if an asymmetric fault occurs is assumed to be $1/G_N$. (At least one good node will be convicted, since the symmetric category covers cases where all receivers correctly identify the fault source.) In the hypothesized membership service, ideally all permanent faulty nodes would be convicted, and all transient faulty nodes would not be convicted since the fault effects persist for only that round (intermittent and longer duration faults are conservatively classified as permanent). In practice, some permanent faulty nodes will be missed and some transient faulty nodes will be convicted, with studied probabilities shown in Table 11.

SEL, SEU, BER. For group membership, these faults are symmetric, since they may cause undetected errors. All receivers would receive the same erroneous frame for SEL faults and SEU faults or BER faults at or near the sender. Some SEU faults might be asymmetric, if a transient SEU alters computation at the receiver for a single frame only. The SEU rate is multiplied by the number of susceptible bits (here, 64 kilobytes is modeled). The SEU would have to hit a certain portion of the integrated circuit to cause the asymmetric fault described, so this is modeled as 10 kilobytes with sensitivity analysis in the results.

Permanent Fault Conviction. Permanent faults, if detected, are convicted and removed from the group. The probability of conviction depends on the diagnosis service discrimi-

Table 12. TTP/C Membership Transitions - Continued on Next Page

Source	Dest.	Items	[Guard], Main Rate Contributor	Rate Range Tested (λ), Per Hour
G_N	PS_N	1	SEL	G_N*(10^{-8}, 10^{-7}, 10^{-6})
G_N	PB_N	1	Perm. HW	G_N*10^{-5}
G_N	TA_N	1	SEU * Asym. Bits	G_N*10K*8*(10^{-10}, 10^{-9}, 10^{-8})
G_N	TS_N	1	SEU * Bits	G_N*64K*8*(10^{-10}, 10^{-9}, 10^{-8})
G_N	TB_N	$\lfloor N/2 \rfloor$	Lightning	4*10^{-4}
G_C	PA_C	1	Perm. Link (one link)	G_C*(10^{-8}, 10^{-7}, 10^{-6})
G_C	PB_C	1	Perm. Link (bus/star)	G_C*10^{-6}
G_C	TA_C	1	BER * Bandwidth	G_C*10^6*3600* (10^{-13}, 10^{-12}, 10^{-11})
G_C	TS_C	1	BER * Bandwidth	G_C*10^6*3600* (10^{-13}, 10^{-12}, 10^{-11})
G_N	A_{NC}	1	[$\neg(\exists G_C) \wedge \exists A_C$], 1/Frame Dur.	3.6*10^7
G_N	S_{NC}	1	[$\neg(\exists G_C) \wedge \neg(\exists A_C) \wedge \neg(\exists B_C) \wedge \exists S_C$], 1/Frame Dur.	3.6*10^7
G_N	B_{NC}	1	[$\neg(\exists G_C) \wedge \neg(\exists A_C) \wedge \exists B_C$], 1/Frame Dur.	3.6*10^7
G_N	CONV	1	[$\exists A_N \vee \exists A_{NC}$],(1/(2*Round Dur.))*Pr.Conv.Good	1.8*10^5*(1/G_N)
PS_N	CONV	1	(1/(2*Round Dur.))*Pr.Conv.Perm.	1.8*10^5*(1.0, 0.99, 0.95, 0.90)
PB_N	CONV	1	(1/(2*Round Dur.))*Pr.Conv.Perm.	1.8*10^5*(1.0, 0.99, 0.95, 0.90)
TA_N	CONV	1	(1/(2*Round Dur.)) *Pr.Conv.Trans. *Pr.Conv.Asym.	1.8*10^5* (0.0, 0.01, 0.05, 0.10)*0.95
TS_N	CONV	1	(1/(2*Round Dur.))*Pr.Conv.Trans.	1.8*10^5* (0.0, 0.01, 0.05, 0.10)
TB_N	CONV	1	(1/(2*Round Dur.))*Pr.Conv.Trans.	1.8*10^5* (0.0, 0.01, 0.05, 0.10)
TA_N	G_N	1	(1/(2*R. Dur.)) *(1-Pr.Conv.Trans. *Pr.Conv.Asym.)	TA_N*1.8*10^5* (1-((0.0, 0.01, 0.05, 0.10)*0.95))
TS_N	G_N	1	(1/(2*Round Dur.))*(1-Pr.Conv.Trans.)	TS_N*1.8*10^5* (1-(0.0, 0.01, 0.05, 0.10))
TB_N	G_N	1	(1/(2*Round Dur.))*(1-Pr.Conv.Trans.)	TB_N*1.8*10^5* (1-(0.0, 0.01, 0.05, 0.10))

Table 13. TTP/C Membership Transitions - Continued

Source	Dest.	Items	[Guard], Main Rate Contributor	Rate Range Tested (λ), Per Hour
TA_C	G_C	1	1/Frame Dur.	TA_C*3.6*10^7
TS_C	G_C	1	1/Frame Dur.	TS_C*3.6*10^7
A_{NC}	CONV	1	(1/(2*R. Dur.))*Pr.Conv.Trans. *Pr.Conv.Asym	1.8*10^5* (0.0, 0.01, 0.05, 0.10) *0.95
S_{NC}	CONV	1	(1/(2*Round Dur.)) *Pr.Conv.Trans.	1.8*10^5* (0.0, 0.01, 0.05, 0.10)
B_{NC}	CONV	1	(1/(2*Round Dur.)) *Pr.Conv.Trans.	1.8*10^5* (0.0, 0.01, 0.05, 0.10)
A_{NC}	G_N	1	[$\exists$ G_C], (1/(2*Round Dur.)) *(1-Pr.Conv.Trans. *Pr.Conv.Asym)	1.8*10^5* (1-((0.0, 0.01, 0.05, 0.10) *0.95))
S_{NC}	G_N	1	[$\exists$ G_C], (1/(2*Round Dur.)) *(1-Prob.Conv.Trans.)	1.8*10^5* (1-(0.0, 0.01, 0.05, 0.10))
B_{NC}	G_N	1	[$\exists$ G_C], (1/(2*Round Dur.)) *(1-Prob.Conv.Trans.)	1.8*10^5* (1-(0.0, 0.01, 0.05, 0.10))

nating between permanent and transient faults. The rate is multiplied by the probability of convicting a permanent faulty node. For asymmetric faults, the conviction rate is also multiplied by the probability of correctly identifying an asymmetric faulty node. The diagnosis algorithm takes two rounds (worst-case) to execute, so the rate per hour is (1 / (2*Round Duration)).

Transient Expiration; Transient Conviction. Ideally, transient faulty nodes will not be convicted and transient faulty nodes will revert to good nodes. In practice, some transient faulty nodes will be misdiagnosed as permanent faulty, and these nodes will be convicted, represented as the probability of convicting a transient faulty node. The rate for this set of transitions also (1 / (2*Round Duration)) for the same reason as the permanent fault conviction case.

Good Node Conviction. If an asymmetric fault occurs, the group will divide into two cliques, a majority clique and a minority clique. Good nodes may be in either clique. If the diagnosis strategy performs conviction, members of the minority clique may be convicted. A new transition is introduced from the G_N state directly to the CONV state, that occurs at a rate of (1 / (2*Round Duration)) with a probability of $1/G_N$. Other probabilities are investigated in the sensitivity analysis. Good node conviction only occurs if an asymmetric fault is present, so the guard is $[\exists A_N \vee \exists A_{NC}]$. This rate is not multiplied by the number of good nodes since an asymmetric fault does not necessarily affect more nodes if the group is larger.

Channels. In group membership, a channel may be symmetric faulty. In TTP/C, if one channel is noisy and the other silent, the receiver counts this as a null frame (not an invalid frame) [118]. Therefore, a symmetric faulty frame will only be received if both channels are symmetric faulty (since an asymmetric channel dominates all faulty channels).

3.8 SPIDER Membership

The Scalable Processor Independent Design for Electromagnetic Resilience (SPIDER) is a set of safety-critical protocols from NASA Langley Research Center [75]. The purpose of SPIDER is to provide a virtual reliable communication bus. The two relevant portions of SPIDER are its Interactive Consistency algorithm (which guarantees agreement among member nodes) and the Distributed Diagnosis algorithm (which proves if a faulty node is able to be identified by the group, and if so allows the node to be convicted and removed from the group). Since faulty nodes are not required to be convicted, permanent and transient faulty nodes can be handled differently. SPIDER uses two types of non-interchangeable nodes. Bus Interface Units (BIUs) are connected one-to-one to an outside processing element. Instead of passive channels, SPIDER elevates the interstage to first-class status, called a Redundancy Management Unit (RMU). BIUs are fully connected to all RMUs through point-to-point links and vice-versa. A SPIDER user specifies processing elements, with the details of reliable communication handled internally by the protocols controlling the BIUs and RMUs.

Table 14. SPIDER Maximum Fault Assumption

SPIDER MFA.1. $2\lvert G \cap E_{RMU}\rvert > E_{RMU} \backslash B$ for all RMUs RMU, and
SPIDER MFA.2. $2\lvert G \cap E_{BIU}\rvert > E_{BIU} \backslash B$ for all BIUs BIU, and
SPIDER MFA.3. $\lvert A \cap E_{RMU}\rvert = 0$ for all RMUs RMU, or $\lvert A \cap E_{BIU}\rvert = 0$ for all BIUs BIU.

Some common mode fault phenomena are outside the ability of a group membership service to handle, and are not modeled here. For example, a lightning strike could disrupt frames from all RMUs since they are sent at the same time (BIU nodes send one at a time), instantaneously violating the maximum fault assumption. These situations would need to be handled separately. One option is different functional requirements, similar to what is done in current lightning test standards [94]. The guarantees could be suspended for the duration of the strike, and would hold after the strike expires (as long as enough clocks stay synchronized). Fast restart/reintegration is another option. Also, at least three nodes of each type are needed to perform a Byzantine fault tolerant majority vote. However, a two RMU system may be desirable to reduce the number of links needed. Future proofs plan to incorporate strictly omissive asymmetric faults, where receivers receive either the same correct value or no value [6]. Only $t + 1$ nodes are needed to tolerate t strictly omissive asymmetric faults [6]. Most permanent link and BER faults would be covered by this new strictly omissive asymmetric category, an improvement on the regular asymmetric category.

The SPIDER maximum fault assumption (MFA) is stated in terms of the sets G, B, S, and A, which denote the sets of good, benign, symmetric, and asymmetric nodes respectively [75]. E_b and E_r refer to the sets of eligible voters for BIUs and RMUs, respectively. Table 14 lists the SPIDER maximum fault assumption. Since the RMUs are first-class entities, the hybrid fault model is applied in two ways: a BIU may become faulty or an RMU may become faulty. For SPIDER, individual link faults are mapped either to a BIU or RMU, and a failure of an entire channel is equivalent to the failure of an RMU. The three pieces of the maximum fault assumption are the death state conditions in the reliability model.

Any death state is a sink state. The reliability models use some additional information about the duration and detection of faults. A faulty node can have either permanent (P) or transient (T) fault duration. Permanent faults have an infinite duration or lasting effects on system state, and transient faults have a finite duration and no subsequent effects on system state. Intermittent faults are conservatively modeled as permanent. Nodes can be (A)symmetric, (S)ymmetric or (B)enign faulty. Nodes that are suspected of being faulty can be convicted (CONV) and removed from the group. In SPIDER, a convicted node is equivalent to a permanent benign faulty node. Transient faulty nodes can be incorrectly perceived as permanent faulty, since it is not possible to perfectly discriminate between transient faulty nodes and permanent faulty nodes in all cases. Also in SPIDER, due to the restriction that no good nodes are ever diagnosed as faulty, it is possible that some faulty nodes may never be diagnosed, since it is impossible to guarantee both conditions in the presence of asymmetric faults [75]. Permanent faulty nodes that are never detected are called undiagnosable (U) in the models. The undiagnosable state is conservatively modeled as a sink state.

The global state of the system S is specified by the tuple {G$_{BIU}$, PA$_{BIU}$, PS$_{BIU}$, TA$_{BIU}$, TS$_{BIU}$, TB$_{BIU}$, UA$_{BIU}$, US$_{BIU}$, CONV$_{BIU}$, G$_{RMU}$, PA$_{RMU}$, PS$_{RMU}$, TA$_{RMU}$, TS$_{RMU}$, TB$_{RMU}$, UA$_{RMU}$, US$_{RMU}$, CONV$_{RMU}$}, where Σ(G$_{BIU}$, PA$_{BIU}$, PS$_{BIU}$, TA$_{BIU}$, TS$_{BIU}$, TB$_{BIU}$, UA$_{BIU}$, US$_{BIU}$, CONV$_{BIU}$) equals the total number of BIUs N and Σ (G$_{RMU}$, PA$_{RMU}$, PS$_{RMU}$, TA$_{RMU}$, TS$_{RMU}$, TB$_{RMU}$, UA$_{RMU}$, US$_{RMU}$, CONV$_{RMU}$) equals the total number of RMUs M. The initial state for the models is G$_{BIU}$ = N, G$_{RMU}$ = M and 0 for all other items in the tuple. For example, one possible system configuration for a system with 4 BIUs and 2 RMUs might be two Good BIUs, one Undiagnosable Asymmetric BIU, one Transient Benign BIU, one Good RMU, and one Convicted RMU. The total Markov or semi-Markov state space depends on the values of N and M as well as the possible classifications, and can be large (in the thousands of states and tens of thousands of transitions).

Ranges for fault rate parameters are given in Table 4. Listed first are the physical fault arrival rates, and ranges studied. It is important to note that all of the ranges studied correspond to real-world expected ranges for these parameters. Table 6 lists the system parameters studied. Table 7 lists the probabilities specified for undiagnosable permanent and misdiagnosed transient faults. The probability of an undiagnosable permanent fault is expected to be low, since there are relatively few cases where the Diagnosis Protocol cannot determine the faulty source. The probability of misdiagnosing a transient fault could possibly be higher, since less emphasis has been placed on this area to date.

Table 15 lists the source and destination states specified in ASSIST for all the transitions in the SPIDER model, the number of nodes affected simultaneously by a transition, and the primary source of faults for fault arrival transitions. All rates are given per hour. For example, the first transition in Table 15 transfers (1) one node from a (G)ood state to a (P)ermanent (A)symmetric faulty state by a fault arrival rate corresponding to the Permanent Link Fault Rate times the number of nodes in the source state (G_x), where the x subscript indicates that this transition applies to both BIU and RMU nodes. Transitions that apply only to BIU nodes have a BIU subscript, and transitions that apply only to RMU nodes have an RMU subscript. BIU and RMU nodes are not interchangeable. There are N BIUs and M RMUs in a configuration. The transitions were determined as follows (listed by order in Table 15):

Permanent Link Faults. A permanent link fault will affect one link between a BIU and RMU. This will be an asymmetric fault since the source node will send good messages to some nodes and no message to the receiver with the faulty link to the sender.

SEL. Single Event Latchup may permanently change the ability of the circuit to produce a correct result. This is modeled as symmetric, since a corrupted value may or may not be detected, but all receivers will receive the same value. (A node might also fail silent and be benign faulty; this would be easier to handle).

Permanent Hardware Faults. Fail-silent nodes will not send messages to any receiver. This case is detectable because the protocol is synchronous during normal operation (different rules may apply for startup/restart). The BIU rate is higher since the BIU is in the same fault containment region as its associated processing element, so the total fault containment region is larger.

BER. Bit errors could cause a benign transient disturbance if all receivers get a garbled message, since error detecting codes that accompany the message catch bit errors with high probability. For example, a transmitter with inadequate power to overcome ambient noise might not be able to transmit a message to any receiver. Bit errors could also cause an asymmetric disturbance, if one receiver gets a garbled message and others receive a correct message. For example, a local noise disturbance might affect only the receivers located nearby.

SEU. A single event upset is a transient fault that may change a stored memory value or cause incorrect logic computations. This is modeled as symmetric, since a corrupted value may or may not be detected, but all receivers will receive the same value. Some SEU faults might be asymmetric, if a transient SEU alters computation at the receiver for a single frame only. The SEU rate is multiplied by the number of susceptible bits (here, 64 kilobytes is modeled). The SEU would have to hit a certain portion of the integrated circuit to cause the asymmetric fault described, so this is modeled as 10 kilobytes.

Lightning. Lightning causes a transient benign fault in BIUs it affects (lightning for RMUs is outside of our fault model, as mentioned previously). In this model, lightning simultaneously causes faults in half of all of the nodes in the system, rounding down for odd numbers of nodes.

Permanent Conviction; Undiagnosable Permanent Faults Once an asymmetric or symmetric permanent fault has occurred, the node will either be diagnosed and convicted or will remain in the system indefinitely, represented in the model by a probability of convicting a permanent node. (Permanent benign faulty nodes are equivalent to convicted nodes.) Since SPIDER guarantees the correctness property, no good nodes will ever be convicted, but this implies that there may be faulty nodes that are unable to be diagnosed. After two executions of the Diagnosis algorithm, a permanent fault will be either convicted or be undiagnosable (the first period might have insufficient evidence if the fault occurs midway through). The probability of permanent faulty nodes becoming convicted or undiagnosable is multiplied by the number of two Diagnosis periods in an hour, (1/(2 * Diagnosis Period)). The Convict Some strategy studies a range of parameters and the Convict All strategy uses 0.95 for the Probability of Convicting a Permanent faulty node (similar to the probability of convicting an asymmetric node in TTP/C). This probability is zero for the Convict None strategy since the p

Transient Conviction; Transient Expiration Once a transient fault has occurred, the fault may expire, or the faulty node may be wrongly convicted and removed from membership. This is modeled by the Probability of Convicting a Transient faulty node. Transient asymmetric faults will be detected when the Diagnosis algorithm executes, and resolved either in that period or in the next period, so the rate multiplier is (1/(2 * Diagnosis Period)). For benign and symmetric faults, transient faults persist for one frame, since the local sets of eligible voters differ only with respect to asymmetric faults.

3.9 Extensibility

Although this work focuses on a particular set of physical faults, the reliability models and modeling techniques are not limited to this set. The reliability models can be extended to include additional states and transitions. As with most modeling techniques, scalability problems can occur if there are too many states or if close transition rates create many loops. However, for many applications the fault arrival rates are much slower than the fault expiration rates since the transient fault durations are short. I demonstrate extensibility by representing a latent fault, as described in the DBench Dependability Benchmarking project [25]. Latent faults are characterized by a potentially long delay between the fault arrival and the observed component failure. For example, latent faults can occur due to accumulated errors in registers that are not directly observable by the user [25].

One way to represent this type of fault would be to explicitly model accumulated errors. This is similar to the treatment of faulty channels which can cause nodes to transition to faulty. Imagine two types of error counter states, a good register state G_R and a faulty register state F_R. At initialization, there is a specified maximum number RMAX of G_R registers and there are zero F_R registers. A latent fault is represented by a transition from G_R to F_R at some rate λ_{latent}. If latent faults have occurred, these faults may cause good nodes to become faulty. Assume that after some minimum number of latent faults MINFAULT that a good node may become permanent benign faulty. This can be modeled as a transition from G_N to PB_N at some rate $\lambda_{activate}$ (or, at a rate that is a function of the number of latent errors c*F_R*$\lambda_{activate}$ for some constant c). The transition will be guarded with [F_R > MINFAULT] for some value of $0 <=$ MINFAULT $<$ RMAX. If the failed component is modeled as containing the latent faults, this transition will also reset the registers, setting G_R to RMAX and F_R to zero. Other variations on this scheme could be modeled; for example, perhaps the good node will transition to a different type of faulty node, or errors continue to accumulate until the end of the mission.

A second way to represent this type of fault would be to use a phased mission, as described by Butler and Johnson [14]. This technique generally applies to missions with non-constant rates, and to missions where failures have different consequences during different operating stages (for example, during aircraft take-off vs. in flight) [14]. To perform a phased mission analysis, a model is created for each phase. Phase models may have different transitions, transition rates, and mission times. The models must have the same states, however. At the completion of a phase, the SURE modeling tool outputs the probabilities of being in each operational state and in each death state, which are used to initialize the next phase model [14]. This sequence is repeated until the last phase model is reached.

Table 15. SPIDER Membership Transitions

Source	Dest.	Nodes	Main Rate Contributor	Rate Range Tested (λ), Per Hour
G_x	PA_x	1	Perm. Link Fault	G_x*(10^{-8}, $10^{-7}10^{-6}$)
G_x	PS_x	1	SEL	G_x*(10^{-8}, 10^{-7}, 10^{-6})
G_{BIU}	C_{BIU}	1	Perm. HW Fault	G_{BIU}*10^{-5}
G_{RMU}	C_{RMU}	1	Perm. HW Fault	G_{RMU}*10^{-6}
G_x	TA_x	1	BER * Bandwidth	G_x*10^6*3600*(10^{-13}, 10^{-12})
G_x	TA_x	1	SEU * Asym. Bits	G_x*10K*8*(10^{-10}, 10^{-9}, 10^{-8})
G_x	TS_x	1	SEU * Bits	G_x*64K*8*(10^{-10}, 10^{-9}, 10^{-8})
G_x	TB_x	1	BER * Bandwidth	G_x*10^6*3600*(10^{-13}, 10^{-12})
G_{BIU}	TB_{BIU}	$\lfloor N/2 \rfloor$	Lightning	4*10^{-4}
PA_x	UA_x	1	(1/(2*Round Dur.)) *(1-Pr. Conv. Perm.)	PA_x*1.8*10^5*(0.01, 0.05, 0.10)
PS_x	US_x	1	(1/(2*Round Dur.)) *(1-Pr. Conv. Perm.)	PS_x*1.8*10^5*(0.01, 0.05, 0.10)
PA_x	$CONV_x$	1	(1/(2*Round Dur.)) *Prob. Conv. Perm.	PA_x*1.8*10^5*(0.90, 0.95, 0.99)
PS_x	$CONV_x$	1	(1/(2*Round Dur.)) *Prob. Conv. Perm.	PS_x*1.8*10^5*(0.90, 0.95, 0.99)
TA_x	G_x	1	(1/(2*Round Dur.)) *(1-Pr. Conv. Trans.)	TA_x*1.8*10^5*(0.90, 0.95, 0.99, 1.0)
TS_x	G_x	1	(1/Frame Dur.) *(1-Pr. Conv. Trans.)	TS_x* 3.6*10^7*(0.90, 0.95, 0.99, 1.0)
TB_x	G_x	1	(1/Frame Dur.) *(1-Pr. Conv. Trans.)	TB_x*3.6*10^7*(0.90, 0.95, 0.99, 1.0)
TA_x	$CONV_x$	1	(1/(2*Round Dur.)) *Prob. Conv. Trans.	TA_x*1.8*10^5*(0, 0.01, 0.05, 0.10)
TS_x	$CONV_x$	1	(1/Frame Dur.) *Prob. Conv. Trans.	TS_x*3.6*10^7*(0, 0.01, 0.05, 0.10)
TB_x	$CONV_x$	1	(1/Frame Dur.) *Prob. Conv. Trans.	TB_x*3.6*10^7*(0, 0.01, 0.05, 0.10)

4 Results

What types of design decisions can be studied? The methodology should enable comparisons between different design choices, and enable trade-off analysis to determine the system parameter changes that result in the highest reliability gain. This analysis is all done at the specification level, when design changes are easier to make. Also, it is important to use a thorough physical fault model.

One type of design decision is to determine the effects of changing the proof assumptions. As an example of an assumption change, I investigate the standard FlexRay clock synchronization maximum fault assumption compared to a novel maximum fault assumption from Azadmanesh and Kieckhafer which includes a strictly omissive fault type [6]. Assessing the reliability of a new maximum fault assumption has great benefit, since proving a new maximum fault assumption typically involves a significant time investment. If a candidate assumption has been proposed but not yet proven, reliability analysis can help the designer decide if the proof is worth the effort. If there is minimal improvement or the reliability decreases, the designer can select a different assumption to prove. Additionally, reliability estimation could increase the adoption rate of new assumptions and proofs that have already been completed. Industry is reluctant to change a system if the impact is unknown, especially if the change might be harmful. By quantifying the reliability improvement (if any), good assumptions and proofs are more likely to be adopted into mainstream use.

Another type of design problem is the construction of a robust fault tolerance strategy appropriate for the application domain. I present case studies of the TTP/C and SPIDER group membership fault tolerance algorithms, which try to balance the risk due to too many active faults vs. the risk of running out of redundancy. In group membership, a perceived faulty node may be convicted and removed from the group. However, if nodes are removed inappropriately (for example, due to transient disturbances or misidentification), the system becomes more fragile since a fault is more likely to cause a failure. For TTP/C, I show that a novel fault tolerance strategy achieves three orders of magnitude improved reliability over the standard strategy.

Each of the protocols has a different sensitivity to the physical fault set. I examine sensitivity to each of the physical fault categories studied, for the clock synchronization services and group membership services of each of the three protocols. Generally, the protocols were most sensitive to transient faults, although permanent faults had an effect on the assumption reliability as well. I also illustrate trend lines in the model results according the physical fault types.

In most cases, one of the fault types is the prime contributor to failure. Reliability analysis can be used to determine the dominant fault by using a separate death condition for each of the types of faults. The reliability analysis will then determine the chance of failure due to each death condition. This information can be used to perform trade-off analysis. For example, adding nodes may improve the reliability if the system is failing due to inadequate redundancy, or it may decrease the reliability if the system is failing due to too many active faults. I show an example of this trade-off for the TTP/C membership service fault conviction strategy.

Another factor to consider is the misclassification rate of the fault conviction algorithm. Even 99 percent accuracy can in some cases produce vastly different results from 100 percent accuracy. If the fault conviction algorithm treats permanent faults differently from transient faults, there will be some misclassification involved. For SPIDER, I show that misclassification can significantly reduce the assumption reliability. Also, in some group membership algorithms (namely TTP/C) it is possible that good nodes may be convicted if an asymmetric fault is present. Therefore it is also important to measure the ramifications of convicting good nodes.

First, this chapter investigates how much a simple physical fault model underestimates assumption reliability compared to a more thorough physical fault model. Next, results from each of the three protocols are presented. A summary of the features of each protocol follows, which could be helpful if a designer wishes to compare two different protocols. The assumption reliability measurements are not directly comparable since the protocols provide different guarantees. Finally, some limitations are reviewed, and areas for future work are presented.

4.1 Statistical Measures

A few statistical measures are used to describe sensitivity to fault rates and system parameters. Box plots are one way to illustrate the sensitivity of a service to a particular parameter. Box plots summarize key facts about a data set, such as the median, 25th and 75th percentiles, and maximum and minimum points. One box plot is drawn per fault type and rate. If the box plot changes when the rate changes, then the assumption reliability is sensitive to that rate. Analysis of Variance is used to determine which factors had the most influence on the output of the models (the assumption reliability measurement). This section gives an overview of these two statistical measures.

4.1.1 Box Plots

Which faults are the models most sensitive to? This question can be answered by comparing some statistics of the data sorted by fault type. Each variable quantity (fault type or system parameter) is called a primary factor. For example, in clock synchronization there are five primary factors studied: the number of nodes, Single Event Latchup rate, Single Event Upset rate, Bit Error Rate, and Permanent Link fault rate. The values of a factor are called levels or treatments.

The analyst would like to know which factors or combinations of factors influence the assumption failure rate. One way to test this is to conduct a full factorial experiment. In a full factorial experiment, "all possible combinations of the levels of the factors are investigated" [77]. "The effect of a factor is defined as the change in response produced by a change in the level of the factor" [77]. A main effect "refers to the primary factors in the study” [77]. Interaction effects are also possible, where an interaction effect is present if “the difference in response between the levels of one factor is not the same at all levels of the other factors” [77]. For example, the effects of any type of fault may depend on how many nodes are in the system. Systems with fewer nodes will be more susceptible to faults, producing an interaction effect between the number of nodes and the fault type. Interaction effects can also be called second order effects if the model is sensitive to a combination of two of the primary factors, or n-order effects where n is the number of primary factors involved.

By generating a box plot for each rate for each fault type, these box plots can be compared to determine sensitivity to each fault type. Figure 12 shows a sample box plot. The box plots were generated with SigmaPlot, which defines a box plot as follows. “Box plots graph data as a box representing statistical values. The boundary of the box closest to zero indicates the 25th percentile, a line within the box marks the median, and the boundary of the box farthest from zero indicates the 75th percentile. Whiskers above and below the box indicate the 90th and 10th percentiles” [104]. Also, all points not between the 10th and 90th percentile are shown on the plot. For example, to examine sensitivity to the Bit Error Rate, one can compare the box plot for all configurations with BER of 10^{-13} to the box plots for configurations with a BER of 10^{-12} and configurations with a BER of 10^{-11}. If the box plots look very different, then that fault type has an effect on the reliability calculation.

Box plots are used here since other common statistical measures could be misleading. Since the assumption reliability can differ by orders of magnitude, the mean value will be biased towards high failure rate outliers. The median and the mode are not necessarily appropriate either as sole measures. The set of configurations is not a random sample of what might be expected in practice. Instead, the set of configurations covers the range of what might be expected in practice.

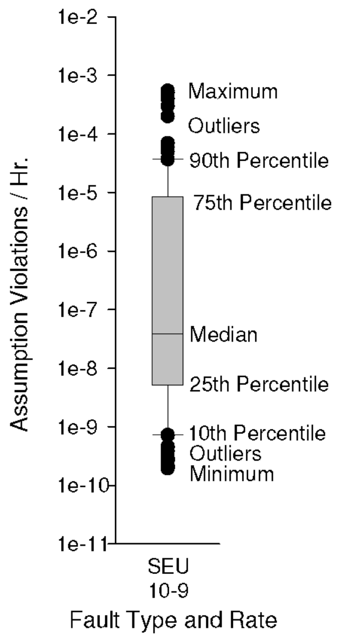


Figure 12. Sample Box Plot

4.1.2 ANOVA and Factors

The analysis of variance (ANOVA) technique tests for equality of treatment effects [77]. "…an Analysis of Variance can be used for comparing means when there are more than two levels of a single factor" [77]. The idea is that if a particular level or treatment of a primary factor has no effect on the outcome, then the mean for the subset of data for that treatment will equal the global mean for the entire population of data. The data is segmented into a set of separate populations, where each population has a given level of the factor being measured. For example, if analyzing the effect of the SEU rate, there will be three populations: a population of configurations with an SEU rate of 10^{-10}, a population of configurations with an SEU rate of 10^{-9}, and a population of configurations with an SEU rate of 10^{-8}. If a factor has an effect on the outcome, then the population means will differ from the grand mean of the entire data set. The same applies to combinations of two or more factors.

Since the set of assumption failure rate measurements forms a full-factorial experiment, the experiment is said to be balanced. In a balanced experiment, the number of observations for each level of a factor are the same [77]. Balanced experiments are less sensitive to deviations in the error and maximize the power of the test, in terms of the confidence intervals that can be calculated [77].

4.2 Physical Fault Model

Part of the contribution of this thesis research is defining a physical fault model for the aviation and automotive domains to test the protocol specifications. There is a large amount of domain specific fault arrival data, with whole research areas devoted to each of the four categories I define. Field data, predictive models (such as neutron flux prediction based on altitude and latitude), and standards that are hundreds of pages long cover many different types of physical faults and fault sources. However, there is a great need for a comprehensive survey of this data at the right granularity. It is unreasonable to expect a system designer to be an expert on all types of faults, even if the system designer is an industry expert.

The state of the art in academia is limited. Most work focuses on developing new algorithms and proofs, and not on developing better techniques to test existing proofs. The thought is that any new proof that further refines the hybrid fault model will be more reliable, but Powell shows that this is not always the case [91]. Therefore, it is imperative that the reliability of new algorithms be evaluated.

What should the algorithms be evaluated with? In the past, there has been some evaluation with selected fault parameters. However, if evaluation is done at all, the parameters tend to be fairly arbitrary. I introduce four main categories of physical fault sources for the aviation and aerospace domains. Data sources include standards, field data, and predictive models (such as neutron flux rate prediction based on altitude and latitude, for single event

Table 16. Faults in Each Fault Model

Fault Model Version	Perm HW	Perm Link	SEL	SEU	BER	Lightning
Permanent HW Only	Y	N	N	N	N	N
Comprehensive Permanent	Y	Y	Y	N	N	N
Comprehensive Transient	N	N	N	Y	Y	Y
All Studied	Y	Y	Y	Y	Y	Y

effects).

This leads to the question, what is the risk of having an overly simplistic fault model? In order to claim that the new physical fault model is a contribution, it must be shown that simplistic fault models give optimistic reliability estimates. I show that this is the case. In fact, the fail-silent hardware only fault model overestimates reliability by 11 orders of magnitude compared to the fault model with all four physical fault categories. Neglecting transient faults was the largest source of reliability overestimation.

4.2.1 SPIDER Physical Fault Model Overview

I tested the assumption reliability of the SPIDER protocol with four versions of the physical fault model, summarized in Table 16. The Permanent HW Only fault model includes only permanent hardware faults that would result in fail-silent nodes, a commonly used fault model. The Comprehensive Permanent fault model includes permanent hardware faults, permanent link faults, and single event latchup faults. The Comprehensive Transient fault model includes single event upset, bit error rate, and lightning fault sources. The All Studied model includes all of the fault sources studied. Every system has its own unique set of faults, and there may be relevant faults not included in the All Studied model here. However, the All Studied model does include many types of faults expected in real-world situations and could be reused for testing assumptions from different systems.

4.2.2 SPIDER Permanent Fault Calculations

The failure rate of the Permanent Hardware Only fault model can be easily computed, since each type of node (BIU and RMU) can only fail in one way. The failure rate of the system can be conservatively approximated as the failure rate of a serial configuration of the three parts of the Maximum Fault Assumption. This approximation is conservative since, although the fault bounds are tight, there may be some situations where part of the MFA has been violated but the system continues to function properly. Since permanent hardware faults map to benign faults, MFA.3 will never be violated because two asymmetric faults must be present to violate MFA.3. For N BIUs with only benign faults allowed, MFA.1 fails if all N BIUs become faulty:

$\lambda_{MFA.1} = \lambda_{BIU}{}^{N}$

For M RMUs with only benign faults allowed, MFA.2 fails if all M RMUs become faulty:

$\lambda_{MFA.2} = \lambda_{RMU}{}^{M}$

The total assumption failure rate is then:

$\lambda_{total} = \lambda_{MFA.1} + \lambda_{MFA.2} + 0$

4.2.3 SPIDER Fault Rates and Model

Since this study was done early in my work, the SPIDER fault rates and reliability model are slightly different from the final fault rates and reliability model used. However, the goal of this study was to show that it is important to include a comprehensive physical fault model. Slight changes in the fault rates or reliability model should not impede this goal. The fault rates used here are summarized in Table 17. The reliability model included the same states and transitions as the SPIDER model in the Methodology chapter, with two exceptions:

1) The transitions G_x to TA_x and G_x to TB_x by BER * Bandwidth were by 1/2 * BER * Bandwidth in this model. Originally, the 1/2 was included to distributed the BER

faults evenly among the two fault states. The 1/2 was removed for future protocol studies because it might be confusing; also, since various BERs are studied the value of 1/2 * BER * Bandwidth would just be between two of the BER * Bandwidth values studied.

2) The transition G_x to TA_x by SEU * Asym. Bits was not included in this model, since SEU faults were originally modeled as benign. This transition was added in later models since SEU faults might be asymmetric.

For the other three physical fault models (besides the Permanent Hardware Only model), many parameter combinations were studied, using the parameters and ranges from Table 17. For example, for the Comprehensive Permanent physical fault model, there were (11 BIU values) * (2 perm. link fault rates) * (3 SEL rates) * (4 undiagnosable probabilities) = 264 combinations. There were 132 combinations for the Comprehensive Transient physical fault model and 3168 combinations for the All Studied physical fault model. All configurations were equally weighted.

For each configuration, a Markov model was created and solved with the NASA ASSIST and SURE tools [13]. The tools output the probability that the SPIDER maximum fault assumption (described in the Methodology chapter) will be violated. This is a conservative approximation of the SPIDER membership service failure rate, since there are cases where the maximum fault assumption is violated but the service's guarantees are still provided. However, there are also provable cases where violating the maximum fault assumption results in the service's guarantees not being provided.

4.2.4 Physical Fault Model Comparison Results

Transient faults are important to include in the fault model, and the commonly used fail silent model is a poor predictor of assumption reliability. Some of the partial model reliability estimates differed significantly from the All Studied model, as shown in Table 18,

Table 17. System Parameters and Values for Early SPIDER

Parameter	Value
Bits Per Node	64 Kilobytes
Message Duration	0.1 ms
Messages/hour	$3.6*10^7$ (3600000 ms / 0.1 ms)
Bandwidth	$1*10^6$ bits/sec
Diagnosis Period	10 ms
BIUs (N)	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13
RMUs (M)	3
Mission Time	1 hour
Perm. HW Fault Rate	10^{-5} faults/hour for BIUs, 10^{-6} faults/hour for RMUS
BER	10^{-12} errors/bit
Perm. Link Fault Rate	10^{-8}, 10^{-7} faults/hour
SEL	10^{-8}, 10^{-7}, 10^{-6} faults/hour
SEU	10^{-10}, 10^{-9}, 10^{-8} faults/bit*hour
Prob. Conv. Permanent	1.0, 0.99, 0.95, 0.90
Prob. Conv. Transient	0, 0.05, 0.10, 0.25

Table 18. Average Assumption Failure Rate, by Fault Model

Model	Perm HW	Comp Perm	Comp Tran	All Studied
Average Assumption Violations/Hr.	$9.2*10^{-17}$	$3.8*10^{-12}$	$9.0*10^{-6}$	$9.0*10^{-6}$

which lists the average assumption failure rate for each fault model. The assumption reliability was overestimated by a factor of about 10^{11} with the Permanent Hardware Only model, by a factor of about 10^6 for the Comprehensive Permanent model, and by surprisingly little for the Comprehensive Transient model when compared to the All Studied model ($9.01*10^{-6}$ vs. $9.04*10^{-6}$). This shows that testing with a thorough physical fault model is important, especially a physical fault model that includes transient faults.

4.3 Clock Synchronization

Since the original Byzantine Generals paper by Lamport, Shostak and Pease [66], the Byzantine fault model has been refined by many researchers, producing a number of hybrid fault models. These hybrid fault models introduce additional categories of faults that are easier to tolerate. With tighter fault bounds, these hybrid models would be expected to improve computed reliability. However, the basic algorithm often needs to be changed in order to take advantage of these tighter fault bounds. If the new algorithm is more complex, new opportunities for faults may be introduced, and the new risks introduced may outweigh the benefits of the tighter fault bounds. Powell notes this in [91]. Even if a new algorithm is better, the algorithm might not be adopted quickly. Practitioners are reluctant to invest time and effort to overhaul a current system if the benefit of a new system is unknown.

By testing assumption reliability, one can quantify the expected increase (or decrease) in reliability of a service due to the new fault bounds. Showing that a new algorithm is more reliable than the old will improve adoption rates. Likewise, if a new algorithm is less reliable than the current algorithm, the designer will be saved from the costly mistake of switching to the new algorithm. Since most new algorithms are geared towards a specific types of systems (for example, there are group membership algorithms tailored to wireless networks) it is important to measure the reliability of the algorithm with respect to the desired domain.

This section compares the Welch and Lynch clock synchronization hybrid fault model (which FlexRay is based on) to a new model by Azadmanesh and Kieckhafer called the Strictly Omissive hybrid fault model. Both of these algorithms are part of a family called Mean-Subsequence-Reduced (MSR) algorithms defined by Azadmanesh and Kieckhafer in [6]. An MSR algorithm can be used to provide approximate agreement for a group of observers. MSR algorithms are used for clock synchronization to guarantee an upper bound on the error among distributed local clocks for a group of nodes. Azadmanesh and Kieckhafer created a new algorithm that could tolerate a new class of faults called 'strictly

Table 19. Clock Sync Maximum Fault Assumptions [120], [33], [6]

Clock Sync$_{WelchLynch}$ MFA.1: $n > 3a$ for n nodes and a asymmetric nodes
Clock Sync$_{WelchLynch}$ MFA.2: $n > b$ for n nodes and b benign nodes
Clock Sync$_{WelchLynch}$ MFA.3: $n > 3a + b$ for n nodes, a asymmetric nodes and b benign nodes
Clock Sync$_{Omissive}$ MFA.1: $n > \alpha$ for n nodes and w strictly omissive asymmetric nodes
Clock Sync$_{Omissive}$ MFA.2: $n > b$ for n nodes and b benign nodes
Clock Sync$_{Omissive}$ MFA.3: $n > \alpha + b$ for n nodes, w strictly omissive asymmetric nodes and b benign nodes

omissive.' The goal is to determine if introducing this new category will improve estimated reliability, compared to the formally proven Welch and Lynch clock synchronization algorithm [120].

Overall, the Strictly Omissive model shows significantly higher reliability than the standard Welch and Lynch clock synchronization model. The best-performing configuration for the Strictly Omissive model outperforms the best-performing standard Welch and Lynch configuration by approximately an order of magnitude ($1.1*10^{-12}$ compared to $1.3*10^{-11}$). The average reliability and worst-performing configuration for the Strictly Omissive model were both better than the Welch and Lynch counterparts.

There are 891 configurations studied for both the Welch and Lynch and Strictly Omissive hybrid fault models, from (3 SEL) * (3 SEU) * (3 BER) * (3 PermLink) * (11 Nodes). 810 of the 891 configurations are graphed here in this section. The four node configurations are omitted from some of the analysis. Since the four node configurations could fail after a single fault (for the Welch and Lynch hybrid fault model), the reliability of the four node configurations will be bounded by one of the fault arrival rates. Table 19 lists the maximum fault assumptions for the Welch and Lynch and Strictly Omissive clock synchronization algorithms.

4.3.1 Physical Fault Sensitivity

Which faults are the models most sensitive to? This section examines the sets of box plots for each clock synchronization algorithm. For clock synchronization, there are five primary factors studied: the number of nodes, Single Event Latchup rate, Single Event Upset rate, Bit Error Rate, and Permanent Link fault rate. (The relationship between the assumption failure rate and the number of nodes is investigated in the next section. Additional factors are considered in the sensitivity analysis section.)

Usually, the assumption reliability is more sensitive to one or two of the types of physical faults. Both modeled clock synchronization algorithms were most sensitive to the Bit Error Rate and the Permanent Link fault rate. The four node configurations for FlexRay were sensitive to different types of faults since a single fault could be a single point of failure. Not counting the four node configurations, the primary faults determining the assumption reliability for both clock synchronization algorithms were (in order) the BER, Permanent Link fault rate, and the combination of Permanent Link fault rate and BER.

The main effects are illustrated in Figure 13 for FlexRay hybrid fault model and Figure 14 for the Strictly Omissive hybrid fault model. In these figures, there are separate box plots for each level of each of the four physical fault types, given on the X axis. For example, the first three boxes correspond to the sets of data with a Single Event latchup rate of 10^{-8}, 10^{-7}, and 10^{-6} respectively. The assumption failure rate in assumption violations/hr is listed on the Y axis, using a log scale. The Y axis scales are the same for both Figure 13 and Figure 14, so the plots for the two clock synchronization schemes can be compared with each other.

These box plots show that the clock synchronization models were most sensitive to the Bit Error Rate and Permanent Link fault rate. For Bit Error Rate, both the median and range of the box (25th through 75th percentile) change quite a bit. This is especially true for the Welch and Lynch algorithm, where an order of magnitude increase in the BER produces an order of magnitude increase in the minimum assumption failure rate, shown

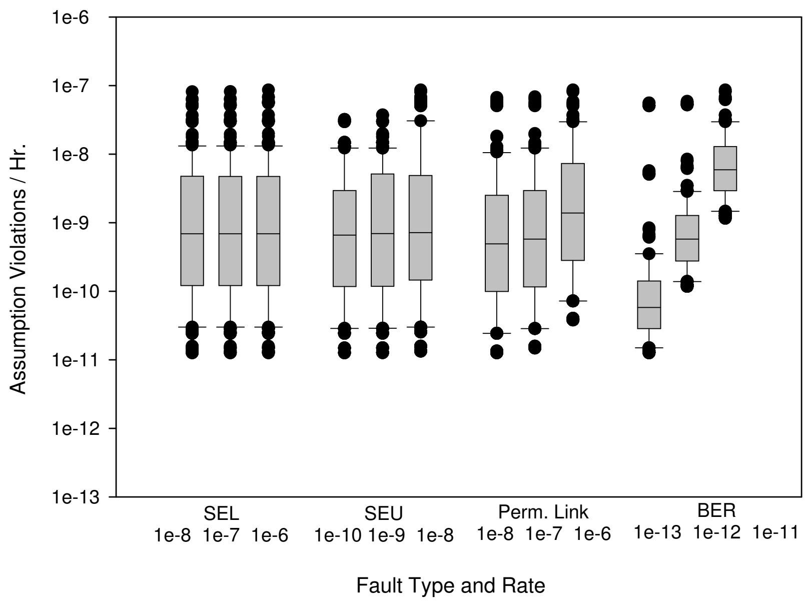


Figure 13. Welch and Lynch Box Plot

as the bottommost point on each box plot. For the Strictly Omissive model, while the minimum failure rate is less affected, an order of magnitude change in the BER produces an order of magnitude increase in the highest assumption failure rate (the topmost point on each box plot). Both clock synchronization algorithms show about equal sensitivity to the permanent link fault rate. The Welch and Lynch algorithm is slightly sensitive to the Single Event Upset rate, whereas the Strictly Omissive algorithm does not appear sensitive to the SEU rate. Neither algorithm appears sensitive to the Single Event Latchup (SEL) rate. The figures include configurations with four nodes, which produce the high failure rate outliers at the top of Figure 13 and Figure 14.

Since box plots only summarize main effects (effects due to a single factor), it is also important to investigate any interaction effects (effects due to a combination of two or more factors). Table 20 reports factors in order of their influence on the assumption failure rate for both clock synchronization models. While the order is interesting, the magnitude of the effect could be misleading since the assumption failure rates were scaled up in order to

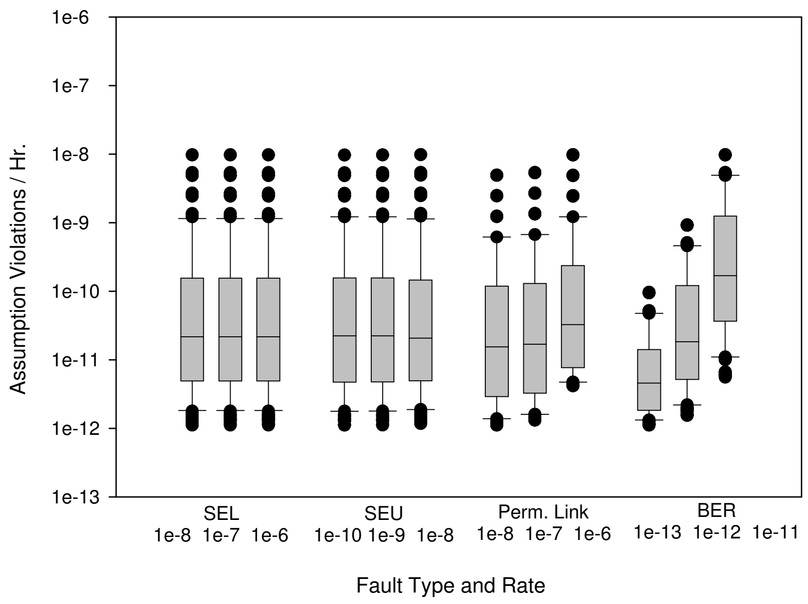


Figure 14. Strictly Omissive Box Plot

perform the analysis. The assumption failure rates were too small for the numerical analysis techniques to work properly. Scaling does preserve the order. Overall, the factors are similar for the two clock synchronization algorithms. The main difference is that Permanent Link faults have more influence for the Welch and Lynch algorithm. One interesting observation is that for both algorithms, one of the fault types has a greater influence on the assumption failure rate than the number of nodes does. This indicates that lowering this fault rate might have more benefit than adding another redundant node.

4.3.2 Welch and Lynch

Figure 15 plots the assumption failure rate of the Welch and Lynch clock synchronization algorithm vs. the number of nodes. The Y axis shows the maximum fault assumption failure rate as the number of assumption violations expected per hour, on a log scale. Overall, adding nodes decreases the expected assumption failure rate. For the Welch and Lynch algorithm, three nodes must be added in order to tolerate an additional asymmetric fault

Table 20. Clock Synchronization: Factors, in Order of Influence

Welch and Lynch	Strictly Omissive Asymmetric
BER	BER
PermLink	Nodes
PermLink*BER	Nodes*BER
Nodes	PermLink
Nodes*BER	PermLink*BER
Nodes*PermLink	Nodes*PermLink
Nodes*PermLink*BER	Nodes*PermLink*BER
SEU*BER	SEU*BER
SEU	SEU
Nodes*SEU	Nodes*SEU*PermLink
Nodes*SEU*BER	Nodes*SEU*PermLink*BER
SEU*PermLink*BER	SEU*PermLink*BER
SEU*PermLink	SEU*PermLink
Nodes*SEU*PermLink	
Nodes*SEU*PermLink*BER	

(since Clock Sync$_{WelchLynch}$ MFA.1 states that $n > 3a$). Figure 15 reflects this. For example, the assumption failure rate decreases when the number of nodes goes from 9 to 10, since 10 is greater than 3*(3 asymmetric faults) but 9 is not greater than 3*(3 asymmetric faults). This also means that asymmetric faults are a significant contributor to assumption failure. Otherwise, if benign faults were the dominant contributor (for example, if the benign fault rate was very high compared to the asymmetric fault rate) then Figure 15 would show more even improvement.

In Figure 15, the assumption failure rate clearly depends upon other factors in addition to the number of nodes. Two of the physical faults account for most of the difference - Bit Error Rate (BER) and Permanent Link fault rate. For the 9 combinations of BER and Permanent Link fault rates (3 BER * 3 PermLink), Figure 16 plots the average assumption reliability per number of nodes. For example, there were nine models studied with the same BER, SEU and number of nodes (since the single event latchup (SEL) rate and single event upset rate varied (SEU)). The assumption failure probabilities of these 9 models were averaged together, and the average assumption failure probability is plotted on Figure 16.

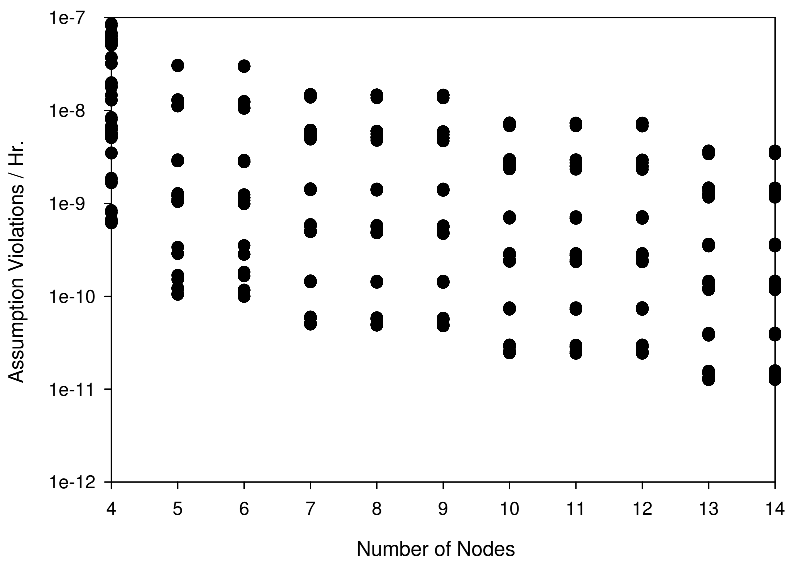


Figure 15. Welch and Lynch: Assumption Reliability vs. Nodes

The contour lines in Figure 15 correspond to these 9 combinations, and the error bars in Figure 16 depict the maximum and minimum assumption failure rate of those 9 combinations. In addition to increasing the number of nodes, lowering the Bit Error Rate or the Permanent Link fault arrival rate can also significantly improve the reliability of the system. For example, lowering the BER by an order of magnitude decreases the assumption failure rate by about an order of magnitude.

Overall, adding nodes decreases the expected assumption failure rate. For the Welch and Lynch algorithm, three nodes must be added in order to tolerate an additional asymmetric fault (since Clock Sync$_{WelchLynch}$ MFA.1 states that $n > 3a$). Figure 15 reflects this. For example, the reliability improves when the number of nodes goes from 9 to 10, since 10 is greater than 3*(3 asymmetric faults) but 9 is not greater than 3*(3 asymmetric faults). This also means that asymmetric faults are a significant contributor to assumption failure. Otherwise, if benign faults were the dominant contributor (for example, if the benign fault rate was very high compared to the asymmetric fault rate) then Figure 15 would show more even improvement.

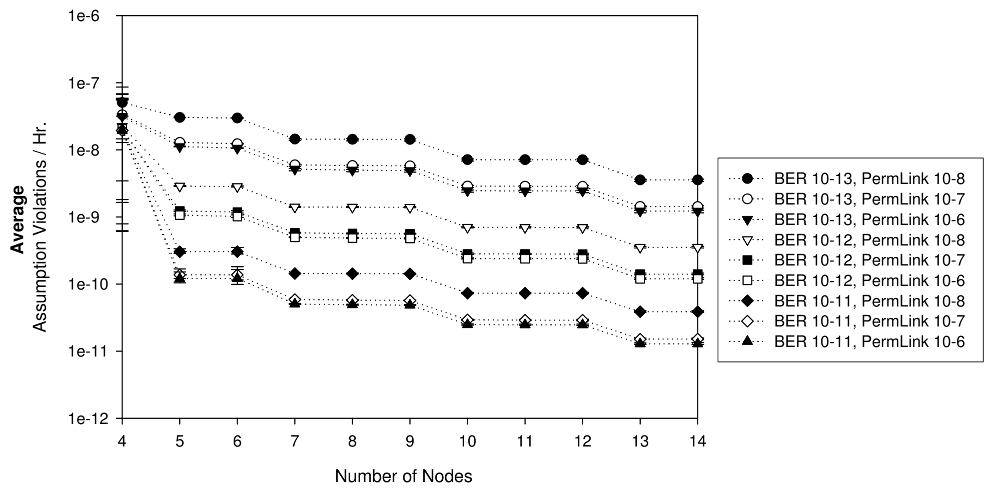


Figure 16. Welch and Lynch: Average Assumption Reliability (BER/PermLink) vs. Nodes

Figure 15 and Figure 16 look fairly similar except for the four nodes configurations. This means that knowing the number of nodes, the BER, and the Permanent Link fault rate will give a fairly complete picture of the assumption reliability of the system. Since these three quantities are the most important, the designer can focus on accurately estimating (and/or reducing) these and can spend less energy on the SEL and SEU rates. For the Welch and Lynch algorithm, note that the four node configurations Figure 16 have significantly larger error bars than configurations with other numbers of nodes. Since the four node systems will fail after a single fault, the dominant cause of failure will be whichever fault occurs most often and therefore will sensitive to all of the fault arrival rates. Therefore, if a four node configuration is used, a designer needs to be concerned about all of the physical fault types.

4.3.3 Strictly Omissive

Figure 17 plots the assumption failure rate of the Strictly Omissive clock synchronization algorithm vs. the number of nodes. The Y axis shows the maximum fault assumption fail-

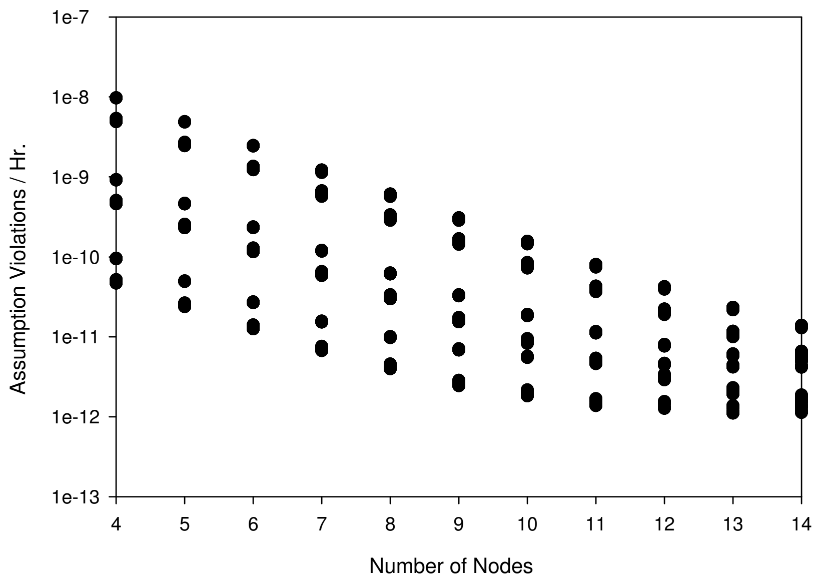


Figure 17. Strictly Omissive: Assumption Reliability vs. Nodes

ure rate as the number of assumption violations expected per hour, on a log scale. Again, adding nodes generally improves the reliability. For the Strictly Omissive algorithm, improvement is seen after adding each single node instead of after adding three nodes. Because all physical faults studied fell under either the strictly omissive asymmetric or the benign categories, Clock Sync$_{Omissive}$ MFA.1 through Clock Sync$_{Omissive}$MFA.3 state that one additional node will tolerate one additional benign or strictly omissive asymmetric fault.

Figure 18 shows the average assumption failure rate as a function of the Bit Error Rate and Permanent Link fault rate. For the 9 combinations of BER and Permanent Link fault rates (3 BER * 3 PermLink), Figure 18 plots the average assumption failure rate per number of nodes, with error bars depicting the largest and smallest number of assumption violations/hr for those configurations. Figure 18 corresponds fairly well with Figure 17. In Figure 18, the line for a BER of 10^{-13} and a Permanent Link fault rate of 10^{-6} is particularly interesting. After about 11 nodes, the assumption failure rate stops improving. However, for configurations with the same BER but lower Permanent Link fault rates (the bottom-

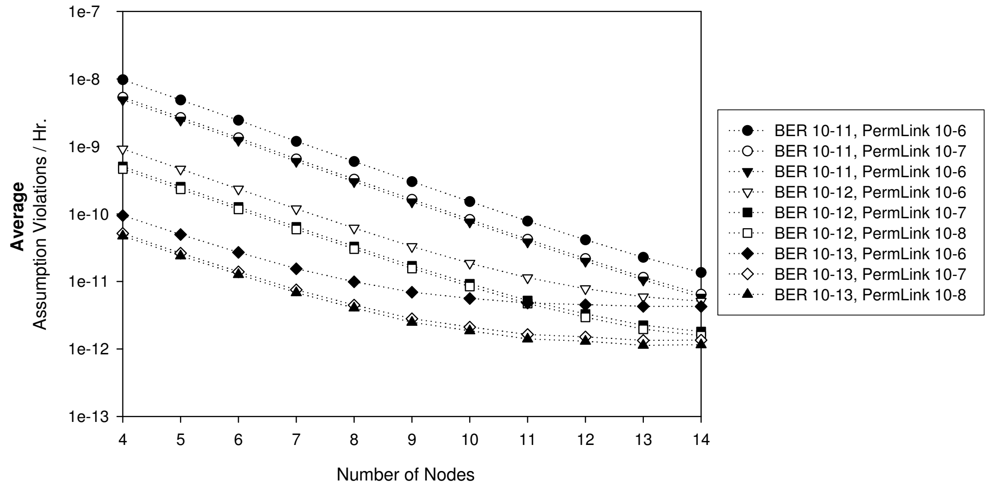


Figure 18. Strictly Omissive: Average Assumption Reliability (BER/PermLink) vs. Nodes

most two lines on the graph), the assumption failure rate continues to improve. This shows that the Permanent Link fault rate is the dominant cause of failure for the BER of 10^{-13} and a Permanent Link fault rate of 10^{-6} configurations after about 11 nodes. Therefore, the Permanent Link fault rate must be reduced to improve assumption reliability - reducing other fault rates will not help.

4.3.4 Welch and Lynch vs. Strictly Omissive

Overall, the Strictly Omissive Asymmetric hybrid fault model outperforms the standard Welch and Lynch hybrid fault model, as Figure 19 illustrates. Table 21 summarizes the number of configurations for each clock synchronization model into a number of reliability bins, and Figure 19 displays a histogram of the data in Table 21. 342 of the Strictly Omissive Asymmetric model configurations achieve an assumption failure rate of 10^{-11} or better, as listed in Table 21. In contrast, none of the Welch and Lynch hybrid fault model configurations achieve a failure rate equal to or lower than 10^{-11}. Figure 19 shows that the number of higher failure rate configurations is greater for the Welch and Lynch model. The

Table 21. Clock Synchronization Summary

Lists the number of configurations in each assumption failure rate bin

Assumption Violations/Hr.	Welch and Lynch configs	Strictly Omissive configs
More than 10^{-7}	0	0
10^{-7} to > 10^{-8}	126 (14.1%)	0
10^{-8} to > 10^{-9}	291 (32.7%)	90 (10.1%)
10^{-9} to > 10^{-10}	282 (31.6%)	171 (19.2%)
10^{-10} to > 10^{-11}	192 (21.6%)	288 (32.3%)
10^{-11} to > 10^{-12}	0	342 (38.4%)
10^{-12} or fewer	0	0

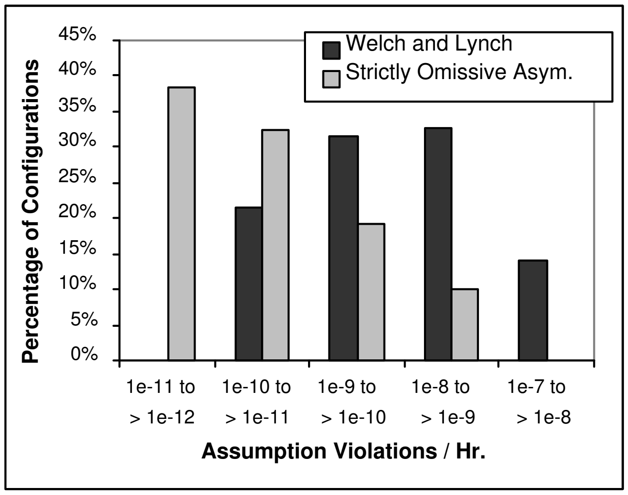


Figure 19. Clock Synchronization: Assumption Failure Rate Comparison

lowest assumption failure rates are $1.1*10^{-12}$ for the Strictly Omissive Asymmetric model and $1.3*10^{-11}$ for the Welch and Lynch model, both for the 13 node configuration with the lowest physical fault rates. The highest assumption failure rates are $9.8*10^{-9}$ for the Strictly Omissive Asymmetric model and $8.6*10^{-8}$ for the Welch and Lynch model, both for the four node configurations with the highest physical fault rates.

The Strictly Omissive Asymmetric model has a lower failure rate than the Welch and Lynch model when configurations with the same physical fault rates are compared. The Strictly Omissive model has a lower number of assumption violations/hr compared to the

Welch and Lynch model for every configuration tested. The amount of improvement is measured as (Welch and Lynch assumption violations/hr) / (Strictly Omissive assumption violations/hr). Note that this will produce a number greater than 1 since all Strictly Omissive configurations are better than the Welch and Lynch counterparts. The improvement is measured this way since the absolute difference between the two strategies could be misleading. For example, a $5*10^{-4}$ configuration compared to a $6*10^{-4}$ configuration would give an absolute difference of $1*10^{-4}$. In contrast, a $5*10^{-9}$ configuration compared to a $5*10^{-7}$ configuration would give an absolute difference of $4.95*10^{-7}$. However, the second pair of configurations shows about two orders of magnitude improvement in the reliability — this is much more interesting to the designer.

One additional stipulation is that the absolute difference between the two configurations must be greater than the error bound. Instead of an exact answer, the SURE tool gives a lower bound and an upper bound. By calculating bounds instead of an exact answer, the computation time is greatly reduced. The upper bound numbers are reported for all datasets (to be conservative - the exact answer will not be greater than the upper bound). Usually the bounds are within 5% of each other [13]. If the difference between the assumption reliability of two configurations is less than the difference between the error bounds for either of these configurations, then the assumption reliability difference is potentially not significant. However, this does not occur for any of the pairs of clock synchronization configurations.

Figure 20 plots the improvement of the Strictly Omissive algorithm over the Welch and Lynch algorithm, per number of nodes. The improvement for the four node configurations is expected, since some faults can be a single point of failure for Welch and Lynch four node configurations but not for Strictly Omissive four node configurations. Excluding the four node configurations, the configuration showing the most improvement under the Strictly Omissive algorithm had 14 nodes and had a BER of 10^{-11} and a Permanent Link fault rate of 10^{-6}, the highest rates studied for these two parameters. In the previous section, the

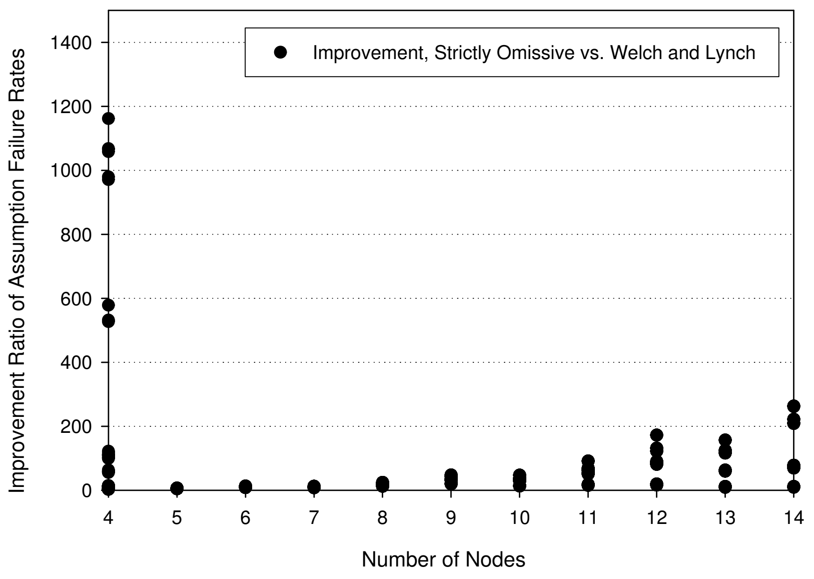


Figure 20. Clock Synchronization: Improvement, Strictly Omissive vs. Welch and Lynch

BER and the Permanent Link fault rate were identified as the dominant faults contributing to failure. This means that the Strictly Omissive algorithm does a better job of handling faults than the Welch and Lynch algorithm.

Figure 20 uses a linear scale on the Y axis, to be consistent with later graphs of the membership diagnosis strategy comparisons. Some of those have negative numbers which cannot be plotted on a log scale. The number of nodes is listed on the X axis. This figure includes all improvement measurements (i.e., there were not any measurements greater than 1500 or less than 0).

4.3.5 Death State Analysis

The clock synchronization algorithms strive to keep a balance among the various causes of failure. When a new hybrid fault model is introduced, the goal is to separate a hard-to-tolerate fault type into a set of new fault types. The hard-to-tolerate fault type can never be completely eliminated (according to the Byzantine Generals proofs [66]), but the frequency of this fault type can be reduced. The set of new fault types should split the original fault type into some easier-to-tolerate fault types and a hard-to-tolerate fault type that occurs less frequently than before. For example, the original Byzantine Generals work was in terms of asymmetric faults only [66]. The symmetric and benign fault types were introduced later, and covered some of the faults previously classified as asymmetric [115]. The frequency of asymmetric faults was therefore reduced, and the service reliability was expected to improve since the symmetric and benign cases were easier to tolerate.

The art in defining a new hybrid fault model is to create a new fault type that covers a useful percentage of the fault space. For example, introducing a fault type that covers asymmetric faults where exactly 3 out of the N receivers get a specific value A, 2 out of N receivers get a specific value B and the rest get a specific value C would probably not be very useful because this case is rare. Ideally all fault types would contribute equally to the total maximum fault assumption failure rate. This means that the hybrid fault model is (more or less) optimized for the physical fault profile. There would be easy-to-tolerate faults that occur more frequently, and hard-to-tolerate faults that occur less frequently. A new hybrid fault model should at least reduce the dominant contributor to assumption violation. For example, separating the benign fault type into two new fault types might not help at all if asymmetric faults are the dominant cause of assumption violation.

The goal of the Strictly Omissive hybrid fault model is to introduce a new type of fault (strictly omissive asymmetric) that will cover faults previously classified as asymmetric. Specifically, this type covers asymmetric faults where some receivers get the same value and others receive nothing or an obviously incorrect value (i.e. the local receiver can diagnose it as incorrect without input from the other nodes). As argued previously, this is exactly what happens in clock synchronization - a value is either correct (on time) or incorrect (not on time).

How does this affect the failure rate due to each fault type? In addition to the total

Table 22. Clock Synchronization Dominant Failure

Lists the number of configurations according to which condition of the maximum fault assumption fails first

Dominant Assumption Failure	Welch and Lynch configs	Strictly Omissive configs
Too Many Active Faults (MFA.1)	837 (93.9%)	540 (60.6%)
Too Few Nodes (MFA.2, MFA.3)	54 (6.1%)	351 (39.4%)

MFA failure rate, the reliability modeling tools output the probability of each piece of the MFA failing (see Table 19 for the clock synchronization maximum fault assumptions). For example, the probability of violating MFA.1. might be high while the probability of violating MFA.3. might be low.

Overall, the Strictly Omissive model seems to have better a fault tolerance scheme for the physical fault rates considered. The Strictly Omissive algorithm balances out the different pieces of the MFA, whereas for the Welch and Lynch algorithm MFA.1 is violated first in all of the configurations except for the four node configurations. The exact number of configurations with each dominant death state is summarized in Table 22. Note that for Welch and Lynch all configurations had either MFA.1 or MFA.3 as the dominant cause of failure, vs. MFA.1 and MFA.2 for the Strictly Omissive model. MFA.1 checks if the number of asymmetric faults has exceeded the total allowed, so this category is labeled "Too Many Active Faults" in Table 22. Both MFA.2 and MFA.3 are cases where benign nodes play a significant role in the failure to satisfy the MFA, so these are both included in the "Too Few Nodes" column in Table 22.

But, is the Strictly Omissive algorithm really balancing the faults well? In other words, are the failure rates of MFA.1 and MFA.2 fairly close together? From the aggregate data it is hard to tell, since it could be the case that the failure rate for MFA.1 is drastically higher than the failure rate of MFA.2 for some configurations, and vice-versa for other configurations. Even though the Strictly Omissive algorithm in Table 22 seems balanced, it might be a lucky coincidence that the range of configurations chosen just happens to produce about 60% of the failures due to MFA.1 and about 40% failures due to MFA.2.

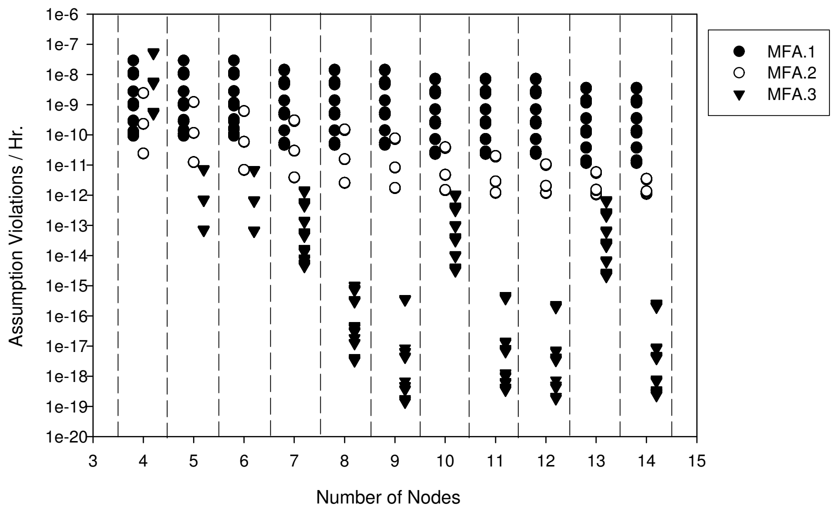


Figure 21. Welch and Lynch: Dominant Assumption Failure Scatter Plot

Figure 22 shows that this is not the case. For the Strictly Omissive algorithm, the failure rate of MFA.1 (solid circles) and the failure rate of MFA.2 (hollow circles) are fairly close together for most configurations. The failure rate due to MFA.3 is quite lower, but this is expected since MFA.3 is Clock Sync$_{Omissive}$ MFA.3: $n > \alpha + b$ for n processors, α strictly omissive asymmetric nodes and b benign nodes. Since MFA.3 is checked last, cases where MFA.1 ($n > \alpha$) and MFA. ($n > b$) have been violated first will not count towards MFA.3.

In contrast, Figure 21 shows that the Welch and Lynch algorithm is much more vulnerable to asymmetric faults than benign faults. For the Welch and Lynch algorithm, only some of the four node configurations fail due to inadequate redundancy (MFA.1), with the rest failing due to too many active asymmetric faults (MFA.3). Having benign faults only (MFA.2) was never the dominant assumption violated. Except for the four node configurations, MFA.1 (solid circles) fails about two orders of magnitude more often than MFA.2 (hollow circles). The somewhat strange looking pattern for MFA.3 is due to the fact that three nodes must be added to tolerate one additional asymmetric fault.

Adopting a Strictly Omissive Asymmetric model for the FlexRay protocol could improve assumption reliability. This could be done by having the clock synchronization algorithm

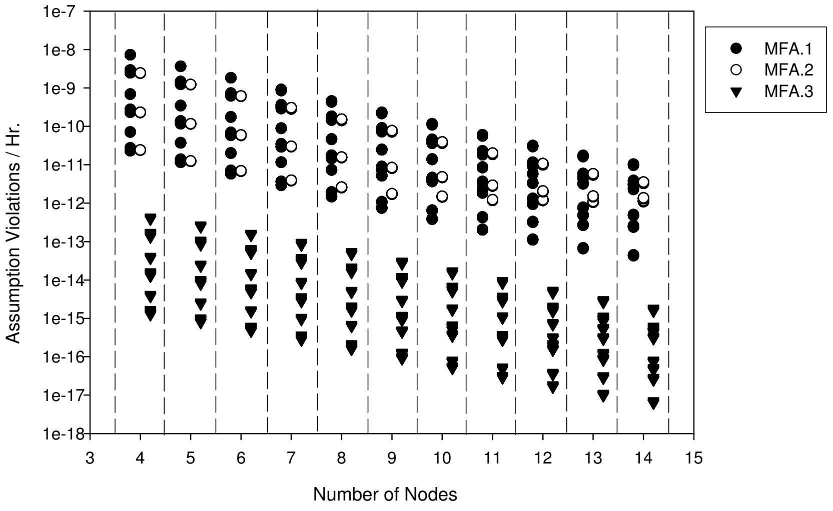


Figure 22. Strictly Omissive: Dominant Assumption Failure Scatter Plot

exclude null or detectably invalid values from the voting process. This should be straightforward to implement, since the rate and offset correction values are stored in a table, and the voting process could operate on just the set of valid values in the table.

4.3.6 Sensitivity Analysis

Some system parameters needed to be specified in order to create the reliability models, where Table 23 lists the system parameter values used for the regular models. This section explores the sensitivity of the Welch and Lynch clock synchronization algorithm to some of these parameters, in particular parameters that might change the impact of Single Event Upset faults. Since the SEU rate is reported in upsets/bit, it is important to estimate how many bits are susceptible to SEU. The size and capacity of the integrated circuit will influence the number of upsets/device-hr. Also, the chance of an SEU causing an asymmetric fault is an important parameter, since asymmetric faults are the most difficult to tolerate.

There is some data available to guide the choice of these parameters, and best-case and worst-case analysis can be done for the asymmetric SEU rate. Two different chip sizes (64K and 256K) and four different percentages of bits affected by asymmetric SEUs (0,

Table 23. System Parameters and Values

Parameter	Value
Bandwidth	$1*10^6$ bits/sec
Round Duration	10 ms
Frame Duration	0.1 ms
Frames/hour	$3.6*10^7$ (3600000 ms / 0.1ms)
Bits/Node	64 kilobytes
Bits, Asym. SEU	10 kilobytes
Clock Sync and TTP/C Nodes, SPIDER BIUs	4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14
Clock Sync, TTP/C Channels SPIDER RMUs	2 3

15%, 50%, and 100%) are explored here. The number of nodes is fixed at 8. There were (3 SELs * 3 SEUs * 3 Perm. Link fault rates * 3 BERs * 8 memory combinations) = 648 configurations. For chip sizes, aviation studies usually cite the chip size in addition to the Single Event Upset/bit rate [79]. Regarding the percentage of asymmetric faults, some fault injection studies have subjected chips to radiation and classified the observed errors. For example, fault injection studies for an early version of TTP/C report about 0.4% asymmetric faults observed out of all of the faults observed [1], [105]. Assumption reliability testing is a nice complement to fault injection studies, leveraging fault injection observations over a short amount of time to make predictions about the behavior of the system in the long run.

When compared to the fault arrival rate parameters, the Welch and Lynch algorithm was relatively insensitive to both the chip size (which would change the SEU fault rate) and the percentage of asymmetric SEU faults (compared to benign SEU faults). Figure 23 shows box plots of the mean, 25th and 75th percentile (box area), 10th and 90th percentile (whiskers) and outlier points for the data subset for each fault type value. In Figure 23, there is practically no impact on the assumption failure rate resulting from changes to the chip size or asymmetric SEU percentage when compared to the other influences (namely the Bit Error Rate and the Permanent Link fault rate).

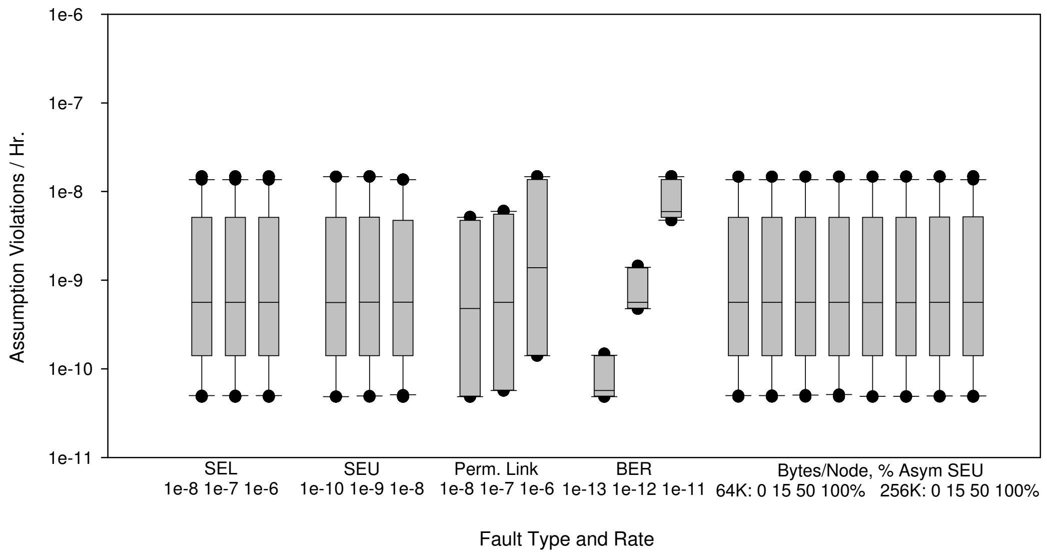


Figure 23. Welch and Lynch: Sensitivity Analysis Box Plot

Notice that the SEU rate also does not affect the assumption failure rate much. Since the assumption failure rate is not sensitive to the SEU rate, it makes sense that the assumption failure rate would also not be sensitive to the chip size and asymmetric percentage that modify this SEU rate. In fact, the assumption failure rate actually improves slightly (for the 8 node configurations studied here) as the SEU rate increases. This could be a consequence of the model, since the fault states are treated as exclusive. Therefore, more SEU faults means that the population of good nodes shrinks, so there are fewer nodes available to become faulty in other ways. In a real system, it would be possible for a node to experience multiple faults almost at the same time. In reality, the node would display behavior consistent with the most harmful fault, and not the fault that occurred first.

4.3.7 Other Modeling Information

The model evaluation execution time depends on a number of factors. The main factors that determine the execution time are the size of the model and (for SURE) the amount of model pruning. In SURE, the designer can specify the pruning level (using the PRUNE constant) and the loop truncation level (using the TRUNC constant). For the pruning level, any path with probability less than PRUNE will be removed from the model. For truncation, any looped path that includes more than TRUNC times through the loop is removed from the model. Pruning and loop truncation can greatly reduce the model solution time. However, if the pruning level or loop truncation level is too high, the SURE tool will be unable to calculate reliability bounds. The probability of the pruned paths is too great, compared to the probability of the other non-pruned paths. In this case, the user is notified with a "Pruning Too Severe" message which can refer to either the PRUNE or TRUNC constant. For all clock synchronization models, a PRUNE level of 10^{-24} and a TRUNC level of 9 is sufficient. In many cases a solution is possible with a lower PRUNE level, so levels from 10^{-20} through 10^{-24} are used. The solution time scales exponentially with the PRUNE level (i.e. reducing the PRUNE level by a factor of ten halves the solution time), so the lowest level possible is desired. The SURE tool aggregates death states according to which piece of the maximum fault assumption is violated first. This greatly reduces the number of death states, since there are only three death states after aggregation.

The SURE tool may be unable to converge on the reliability bounds if the fault arrival rates become too close to the transient recovery rates. Models with a set of slower rates and a set of faster rates are called "stiff" models, and pose a set of mathematical challenges. Some high fault arrival rates were untestable; Bit Error Rates above 10^{-11} are not tested for this reason. In general, increasing the fault arrival rates also increased the solution time .

Table 24 summarizes some of the performance statistics for the Welch and Lynch clock synchronization service. The Strictly Omissive models were the same size, but generally took longer to execute since the reliability level is higher on average. Therefore the Strictly Omissive models could not be pruned as much as the Welch and Lynch models. The CPU timer used rolled over for some of the executions so execution time is not available for all configurations of the Strictly Omissive algorithm. While the average execution time

Table 24. Welch and Lynch: Clock Sync Model Size and Execution Time

Nodes	States	Transitions	Avg. Execution Time (CPU sec.)
4	333	2750	112
5	648	5465	81
6	1143	9800	103
7	1908	16565	126
8	3033	26630	128
9	4608	40865	147
10	6783	60660	583
11	9708	87485	514
12	13533	122810	590
13	18483	168760	423
14	24783	227560	435

is between 80 and 600 seconds (about one to ten minutes), there was a large amount of variation. Most models were solved much more quickly (many in under ten seconds), with some models taking much longer (over an hour). Models with higher BERs (10^{-11}) overall took the longest to solve. Also, the pruning level was not optimized per model, so shorter times may be possible.

4.4 Membership Study

A group membership service may remove suspected faulty nodes from the group. This process is called *conviction*. The *conviction policy* states the conditions under which a node will be removed from the group. Choosing an apt conviction policy requires an in-depth understanding of the types of faults that can occur in the system, and the types of recovery actions needed to return a node to correct operation after a fault.

A successful conviction policy keeps the number of active faults within the bounds of the system's maximum fault assumption (MFA). Theoretical results for group membership show that there are two general pieces to the system's MFA — there is a limit on the number of simultaneous active faulty components within one round, and there is a limit on the number of rounds that may contain a Byzantine faulty component. Therefore, an

MFA will contain restrictions of the form $nodes > c1 * faultynodes + c2$ and $rounds >= c3 * faultyrounds + c4$ for some constants $c1$, $c2$, $c3$, and $c4$. For example, the traditional Byzantine fault tolerant MFA is $n >= 3 * faultynodes + 1$ and $r >= faultyrounds + 1$ [66], [35]. For Byzantine fault tolerant group membership, this MFA has been shown to be a lower bound [36]. Later work has defined a number of easier to tolerate fault classes and additional bounds with respect to those classes. For example, a fail-silent omission fault where no receiver receives any message provably requires fewer nodes to tolerate. The conviction policy must balance the risk of having too many active faulty nodes vs. the risk of removing too many nodes from the group (especially for protocols that allow conviction of good nodes, such as TTP/C).

I divide the space of faults into *permanent* faults and *transient* faults. As used here, the term transient fault refers to a fault which persists for a finite duration, ceases to exist after that duration has expired and does not alter the state of the affected component beyond that duration. For network protocols, it is convenient to think of a transient fault as something that affects a single frame. For example, some radiation-induced bit upsets might affect the current data value being sent, but can be corrected with error detection and correction codes in memory before the next data value is sent. One can also model a transient fault in terms of a number of consecutive frames affected from a single sender (for example, due to a burst of electromagnetic interference). A permanent fault is a fault with infinite duration or lasting effects on state. An intermittent fault could be classified either as permanent or as a set of transients. The exact duration allowed for a transient fault will depend on the fault identification procedure, which is outside the scope of this work. For modeling purposes, only the misclassification rate needs to be represented.

Note that it is theoretically impossible to perfectly discriminate between transient faults and permanent faults for asynchronous systems. There could be infinite delay between the sending and reception of a message. Waiting for an infinite time is not an option for practical systems, so timeouts can be used to discriminate a transient fault from a permanent fault by selecting an appropriate Δt. Faults with duration less than Δt are considered transient, while others are considered permanent. However, there is no value of Δt that is guaranteed to separate all transient faults from all permanent faults, since transient faults occur in actual time (which can be thought of as having a real number domain) [68].

The standard conviction policy for both the TTP/C and SPIDER is to effectively set Δt to zero, thereby making permanent faults and transient faults equivalent. This does not mean that treating transient faults as permanent faults will cause the system to fail, but it does indicate a major inefficiency. It is important to note that the maximum fault assumption implicitly covers both permanent and transient faults, since the MFA is a statement about the total number of currently active faults. The conviction policy, however, is in theory free to treat permanent and transient faults differently. In practice, the flexibility of the conviction policy depends upon the protocol proof. In TTP/C, the proof of the conviction algorithm is integrated with the proof of the rest of the membership service. Thus any change in the conviction algorithm would likely require changes to the proof, which could be a large time investment. The SPIDER conviction algorithm is more flexible, but with the restriction that no good node may be convicted (implying that there may be some undiagnosable faulty nodes that cannot be convicted). In general it is desirable to separate the membership algorithm from the rest of the system as much as possible, a concern that has also been explored (for example) in systems that guarantee Byzantine fault tolerance and provide some security services [127].

Reintegration is an interesting related issue. An inaccurate conviction policy that convicts many good nodes might be acceptable if a reintegration procedure is able to reintegrate nodes back into the group quickly. Unfortunately, at this time there are no formally proven reintegration strategies available for TTP/C or for SPIDER, although reintegration strategies are being developed. Also, the risk of reintegrating a faulty node needs to be considered. Once more progress has been made in this area, designers could include reintegration in the models as a transition from a convicted state back to a good (or possibly faulty) state. Since the relationship between the conviction policy and reintegration strategy is important, designers would want to consider the reliability of different combinations of conviction policies and reintegration strategies.

The three conviction policies I investigate are:

- Convict All (standard): Nodes are convicted after a single fault. This is equivalent to treating all faults as permanent.
- Convict None (no-op): No nodes are convicted. This is equivalent to treating all faults as transient.
- Perfect/Convict Some: The system attempts to convict all permanent faulty nodes and to let transient faults expire. There are two specified percentages of misclassification: permanent faults misclassified as transient and transient faults misclassified as permanent (roughly speaking, false negatives and false positives). If the protocol allows good node conviction, there is an additional percentage specified for incorrectly convicting good nodes.

An interesting observation is that in many of the experiments, doing nothing (Convict None) had superior reliability to the standard conviction policy (Convict All). This might seem straightforward in light of data that shows transient faults outnumbering permanent faults by a factor of a thousand. However, to this date specification efforts have largely ignored the problem posed by transient faults. Until now, there has been no attempt to critically examine the likelihood that the maximum fault assumption will be violated. The thought was that by removing a faulty node from membership immediately, the system would be 'safer.' My research has shown that the opposite can occur – the system may become less reliable – especially in the absence of reintegration (with no formally proven reintegration strategies, it is difficult to argue that reintegration is 'safe' at this time). For SPIDER, since the conviction policy is flexible, future work is likely to include an improved conviction policy. For TTP/C, the future is less certain, since the conviction policy is integrated with the membership proofs. One alternative for both protocols is to develop a reintegration strategy that overcomes the shortcomings in the conviction policy. For example, nodes may be voted out after a single fault but then quickly reintegrated after the group reaches consensus on its members (at minimum, after the distributed diagnosis period in SPIDER and after two rounds in TTP/C).

The Convict Some strategy shows improvement over the other two strategies. For TTP/C the amount of improvement is three orders of magnitude in some cases. For SPIDER, the amount of improvement is contingent upon the misclassification probabilities. This shows how crucial it is to test assumption reliability at the design stage. I present results from both the SPIDER and TTP/C group membership services. In addition to comparison of conviction strategies, I present a sensitivity analysis for both protocols to various types of faults.

4.5 TTP/C Membership Results

TTP/C provides a low-overhead, Byzantine fault tolerant membership service. In TTP/C the diagnosis service and the membership service are tightly coupled to achieve this low overhead (where overhead here is measured in extra bits per frame). In the lowest overhead configuration, a Cyclic Redundancy Code is calculated over the frame contents and sender's local membership vector. The membership vector is not explicitly included in the frame. The receiver calculates its CRC over the frame contents and its local membership vector. Thus if the membership vectors do not match, the receiver will consider the frame to be faulty and will remove the sender from membership. However, the membership service will require no extra bits per frame (except possibly if the CRC length needs to be extended to maintain detection power since the CRC is protecting more bits). The membership vector can also be explicitly included with every frame.

The TTP/C Maximum Fault Assumption is given in Table 25. MFA.1 is slightly different from the one given in the proofs of TTP/C's Interactive Consistency and Clique Avoidance

Table 25. TTP/C Membership Maximum Fault Assumption [118], [88]

Membership MFA.1: If (∃ a), then $a + s + b = 1$ for a asymmetric, s symmetric, and b benign nodes
Membership MFA.2: $s \leq g$ for s symmetric and g good nodes
Membership MFA.3: $g \geq 3$ for g good nodes

algorithm. In these proofs, if a fault occurs, and another fault occurs before the group has reached consensus on its members, then consensus might never be reached [88]. Therefore, MFA.1. is slightly optimistic since it only forbids two simultaneous faults, and not two faults within two rounds. MFA.1. is tested this way since the modeling tools do not specifically incorporate a notion of rounds.

This section examines the three conviction strategies individually, then compares them to each other. The Convict Some strategy performed the best, followed by the Convict None (do nothing) strategy, then the Convict All (standard) strategy. It was surprising that the do nothing Convict None strategy outperformed the standard Convict All strategy. This means that all of the effort invested in the membership proofs so far produced a system with less reliable guarantees (overall). This show the importance of measuring assumption reliability.

Once a shortcoming has been identified, assumption reliability testing can give an objective measurement of the outcome of changing the specification. For TTP/C, the amount of improvement of the Convict Some strategy was also very interesting. The Convict Some strategy performed up to three orders of magnitude better than the standard Convict All strategy. Because of the way the TTP/C proofs are written, altering the conviction algorithm to implement the Convict Some strategy might require changes to the proofs. One alternative would be to use a rapid reintegration strategy for transient faulty nodes, and a slower (if any) reintegration strategy for the permanent faulty nodes. Formally proven reintegration algorithms have not yet been written for TTP/C (or for SPIDER).

4.5.1 Physical Fault Sensitivity

Which faults are the models most sensitive to? This question can be answered by comparing some statistics of the data sorted by fault type. For the TTP/C Convict All and Convict None strategies, there are five primary factors studied: the number of nodes, Single Event Latchup rate, Single Event Upset rate, Bit Error Rate, and Permanent Link fault rate. For the Convict Some strategy, two additional factors are investigated: the probability of convicting permanent faulty nodes and the probability of convicting transient faulty nodes. These probabilities can be thought of as the (1 - false negative) rate and the false positive rate, respectively. Box plots are created for each primary factor investigated. For more detail on box plots and the statistics involved, please see the previous discussion in the Statistical Measures section. Sensitivity to the number of nodes is investigated in the next section, and additional factors are considered in the sensitivity analysis.

Like for clock synchronization, the reliability of the four node TTP/C configurations is low because a single lightning strike could cause failure. A lightning strike would be a single point of failure for the 4 node configurations since $\lfloor N/2 \rfloor$ nodes are affected, as the transitions are currently specified (please see the Methodology chapter for a complete list of the transitions). A lightning strike would eliminate two of the four nodes, and three good nodes are required at minimum.

There is one set of box plots for each strategy. Figure 24 shows the standard Convict All strategy, Figure 25 shows the Convict None strategy, and Figure 26 shows the Convict Some strategy. The X axis lists the different levels of each primary factor. The assumption failure rate is listed on the Y axis in terms of assumption violations/hr using a log scale. In all of these figures, the maximum assumption failure rate stays about the same (at about 10^{-3} assumption violations/hr). These maximum points are the four node configurations, which have single point of failure in the event of a lightning strike (according to the lightning model used). The Y axis scales are the same for all three figures, so that the figures may be compared with each other.

The box plots show that all three strategies were most sensitive to the Single Event Upset rate and the Bit Error Rate. For the SEU rate, both the median assumption failure rate and the assumption failure rate range change for all three strategies. For the BER, the median assumption failure rate stays fairly constant, and the maximum assumption failure rate stays the same, but the minimum assumption failure rate is affected. For each order of magnitude increase in the BER, there is an order of magnitude increase in the minimum assumption failure rate. All three strategies seem slightly sensitive to the permanent link fault rate and are fairly insensitive to the Single Event Latchup rate.

The Convict All strategy is interesting since this strategy is significantly affected by lightning and the way this fault transition was modeled ($\lfloor N/2 \rfloor$ nodes become faulty). Since the Convict All strategy will always remove these nodes from membership, the maximum fault assumption will be violated after two lightning strikes since that will result in all of the nodes being removed from the system (half of the total group N the first strike, and half the second strike). The probability of two lightning strikes as modeled is $(4*10^{-4})^2$, or $1.6*10^{-7}$. Thus the best assumption failure rate achieved was about 10^{-7} assumption violations/hr, no matter what the other fault arrival rates were. Fortunately the median assumption failure rate is close to 10^{-7} assumption violations/hr, meaning that most of the configurations perform this well. Sensitivity analysis examines different lightning models.

There are two additional factors for the TTP/C Convict Some model - the permanent fault misclassification rate and the transient fault misclassification rate. These are represented in Figure 26 as the Probability of Convicting a Permanent faulty node and the Probability of Convicting a Transient faulty node. Convicting a permanent faulty node can be though of as a ‘false negative’, and convicting a transient faulty node can be though of as a ‘false positive’. Ideally, the probability of convicting a permanent faulty node would be 100%, since the reliability will improve if permanent faulty nodes are removed. The probability of convicting a transient faulty node would be ideally be zero, since transient faults will expire and the affected node will return to normal the next communication round. Removing

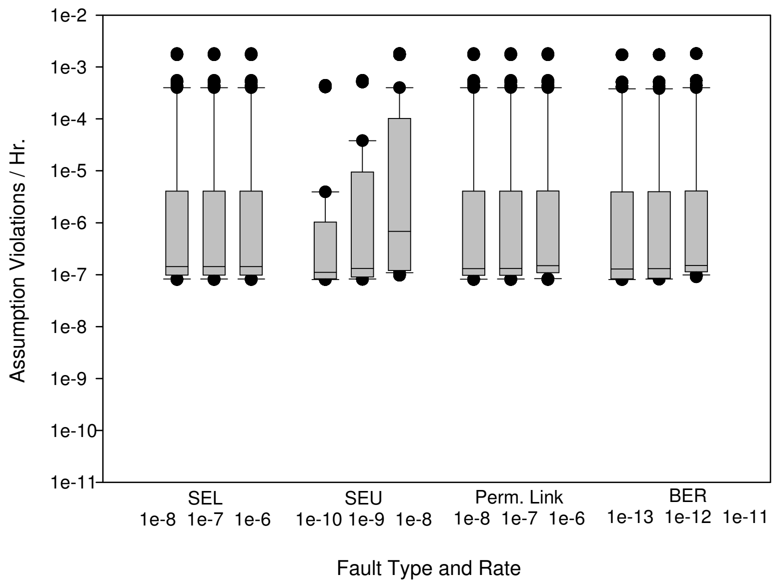


Figure 24. TTP/C Convict All, Box Plot

transient faulty nodes would reduce available redundancy. However, no algorithm will be able to perfectly distinguish between permanent and transient faults.

The misclassification rates do not change the reliability much by themselves for TTP/C, as Figure 26 shows. This makes sense, because the effects of misclassification depend on the fault arrival rate. For example, a high rate of misclassification with a low fault arrival rate might produce the same reliability as a low rate of misclassification with a high fault arrival rate.

While the main effects seem small, notable interaction effects are present. After the effects listed previously, the Convict Some models are most sensitive to the probability of transient node conviction, the SEU/transient conviction interaction, the nodes/transient conviction interaction, and the SEU/nodes/transient conviction interaction. In contrast, there was very little sensitivity to the permanent misclassification rate. This means that ‘false positives’ were more detrimental than ‘false negatives’ for the range of fault arrival rates studied. Overall, a less aggressive fault tolerance strategy is preferred. Since all group

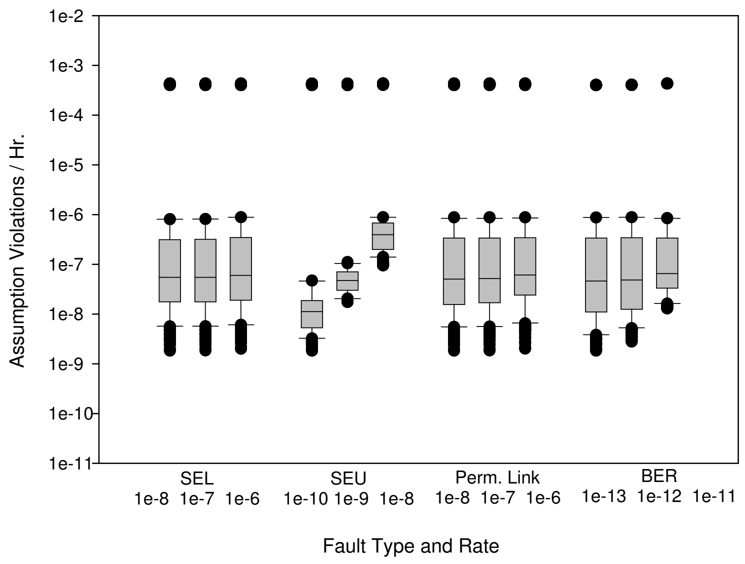


Figure 25. TTP/C Convict None, Box Plot

members agree on the group size, membership could use a diagnosis strategy tailored to the current number of nodes in the group, being more aggressive for large groups and less aggressive for small groups.

Table 26 lists factors and combinations of factors that influenced the assumption failure rate, from greatest influence to lowest influence. Effects due to a single factor are called ‘main effects’ and effects due to a combination of factors are called ‘interaction effects’. While the ordering is interesting, the actual magnitude of the effect is not very useful in this case since the assumption failure rate measurements had to be scaled up to perform the analysis. Scaling does not affect the order, however. These factors were analyzed without the four node configurations, since the Nodes factor would then appear to have quite a bit more influence. Also, some factors at the bottom of the list that did not satisfy the confidence interval requirements were omitted. All possible factor combinations were tested except for the combination involving all of the factors, since the analysis techniques require at least one degree of freedom for the error term.

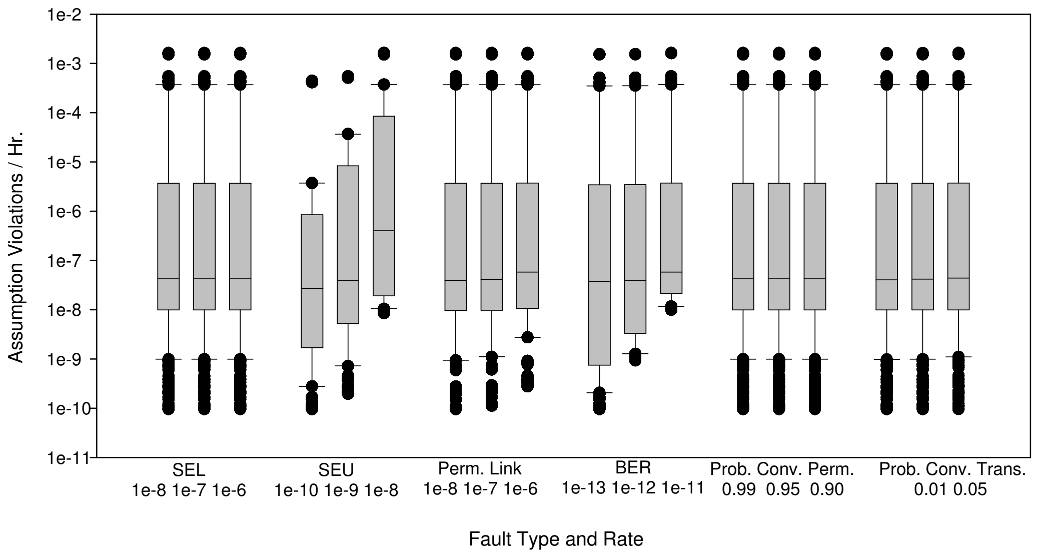


Figure 26. TTP/C Convict Some, Box Plot

From Table 26, the Convict All and Convict Some strategies have almost the same ordering of factors (except for the probability of convicting transient faulty nodes, which is always one in the Convict All strategy). For all strategies, the SEU rate had more of an influence on the assumption failure rate than the number of nodes. This indicates that reducing the SEU rate through better quality components might improve the reliability more than adding another redundant node. Looking at Table 26, for the Convict None strategy the SEL rate seems to have more influence than the Permanent Link fault rate, although neither factor has a large amount of influence. This seems a little different than Figure 25, where the only apparent change in the box plots is the minimum for the highest Permanent Link fault rate.

4.5.2 Convict All (Standard) Strategy

Figure 27 shows the assumption reliability of the Convict All strategy vs. the number of nodes. The Y axis lists the number of maximum fault assumption violations per hour,

Table 26. TTP/C Membership: Factors, in Order of Influence

Convict All	Convict None	Convict Some
SEU	SEU	SEU
Nodes	Nodes	Nodes
Nodes*SEU	Nodes*SEU	Nodes*SEU
BER	SEL	BER
SEU*BER	SEL*SEU	SEU*BER
Nodes*BER	BER	Nodes*BER
Nodes*SEU*BER	SEU*BER	Nodes*SEU*BER
PermLink	PermLink	Prob. Conv. T.
PermLink*BER	PermLink*BER	SEU*Prob. Conv. T
SEL	Nodes*SEL	Nodes*SEU*Prob. Conv. T
Nodes*SEL	Nodes*SEL*SEU	PermLink
SEL*SEU	Nodes*BER	PermLink*BER
Nodes*SEL*SEU	Nodes*SEU*BER	SEU*BER*Prob. Conv. T
Nodes*PermLink	SEL*SEU*BER	BER*Prob. Conv. T
Nodes*PermLink*BER	SEL*BER	Nodes*BER*Prob. Conv. T
		SEL
		Nodes*SEL

on a log scale. As Figure 27 shows, adding nodes decreases the assumption failure rate. However, after about 11 nodes, the amount of improvement diminishes due to the effects of lightning. There is relatively steady improvement from the 4 node configurations through the 10 node configurations.

In Figure 27, it is apparent that other factors affect the assumption failure rate besides the number of nodes. From the previous section, the Single Event Upset rate and the Bit Error Rate were the two types of faults that the models were most sensitive to. There are three SEU rates and 3 BERs, for a total of nine combinations of fault rates. Figure 28 plots the average assumption failure rate per number of nodes for each of these nine combinations of fault rates. There were nine models for each of the nine combinations. The average assumption failure rate of these models was computed and graphed. Error bars at each point show the maximum and minimum values for the nine data points that were averaged.

Figure 28 shows that the assumption reliability for the Convict All strategy is almost entirely determined by the Single Event Upset rate and the number of nodes. There are

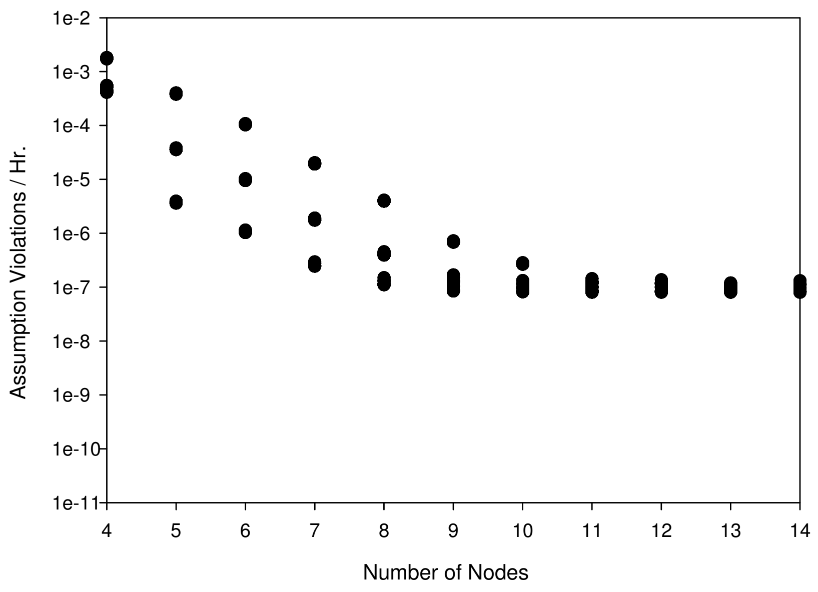


Figure 27. TTP/C Convict All: Assumption Reliability vs. Nodes

three lines in 28, with the topmost line corresponding to an SEU rate of 10^{-8}, the middle line corresponding to an SEU rate of 10^{-9}, and the bottom line corresponding to an SEU rate of 10^{-10}. Figure 27 and Figure 28 look nearly identical. The error bars on Figure 27 are barely perceptible, indicating that for each group of 9 data points with the same BER and SEU, the mean, minimum, and maximum assumption failure rate are nearly the same. Since the SEU is the dominant factor influencing the assumption failure rate, this shows that the Convict All strategy might have trouble tolerating transient faults.

The Convict All strategy also appears to have problems handling burst faults, especially transient burst faults. The assumption failure rate levels off at $1.6*10^{-7}$ due to lightning. While the way lightning was modeled is somewhat arbitrary, burst faults definitely do exist and will affect the system reliability. The duration of the lightning burst modeled is 5ms, or half of a 10ms round. Many burst faults are longer than this [64]. Because TTP/C tightly integrates agreement with fault diagnosis, losing agreement is equivalent to the entire group being declared faulty. System restart will be required if all of the nodes consider themselves to be faulty and cease sending frames.

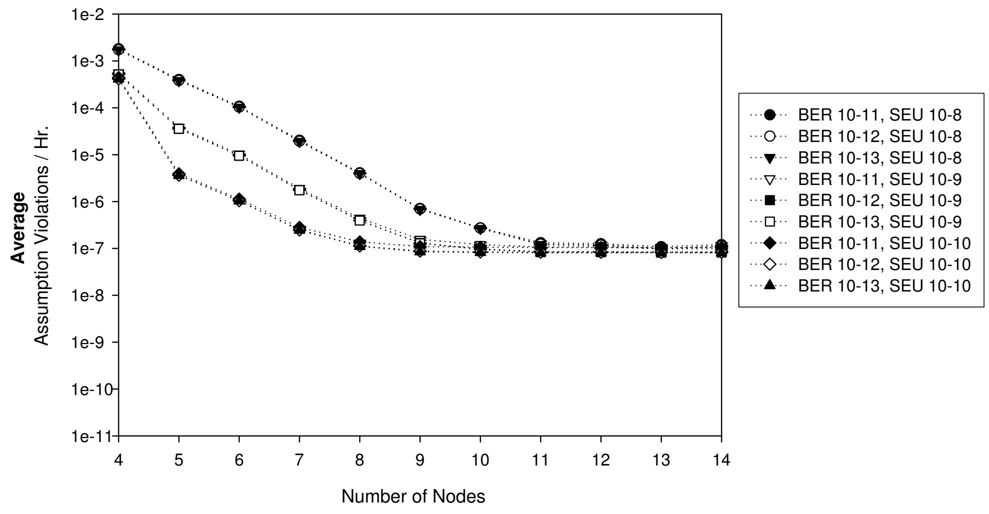


Figure 28. TTP/C Convict All: Average Assumption Reliability (SEU/BER) vs. Nodes

One technique for handling burst faults could be to design the diagnosis round duration with respect to the burst duration. No matter what, agreement cannot be preserved if all of the frames in the diagnosis round are lost or corrupted. The round length is usually determined by the shortest message period in the network, which is around 5-10ms for many common embedded systems [116]. Unfortunately, transient bursts of 10ms are not uncommon. For example, automotive electromagnetic testing standards include a number of transient pulses of 10ms duration or longer [116].

Burst faults might also be handled through special detection and diagnosis strategies, studied to some extent in the Convict Some strategy. The Convict Some strategy classifies transient and permanent faults separately, where burst faults would be included in the transient fault class. The transient fault class could be further refined into single-frame transient faults and multiple-frame transient faults. However, if a burst fault lasts for a number of diagnosis rounds, the system may be better off treating this as a permanent fault. Some provisions for long-duration burst faults are included in the protocols already. The latest version of the TTP/C spec (1.4.3) includes a blackout detection procedure for communica-

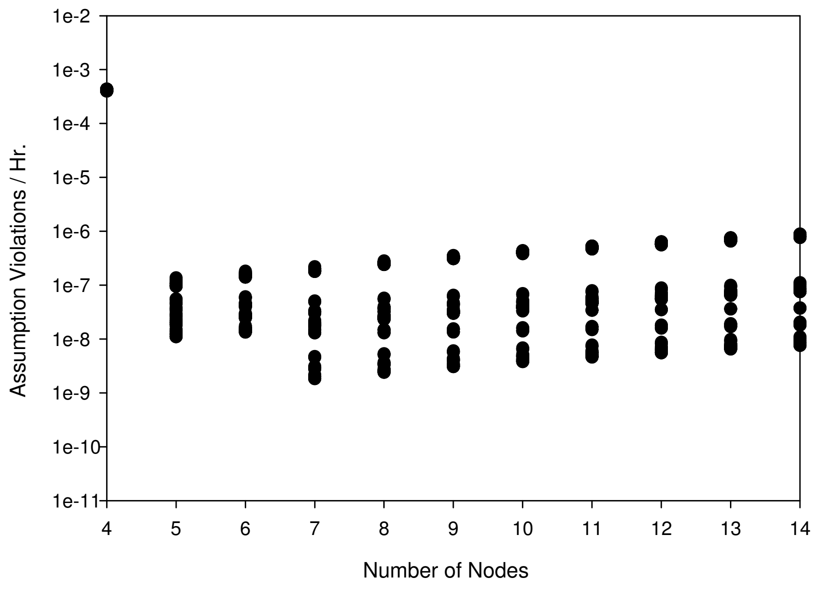


Figure 29. TTP/C Convict None: Assumption Reliability vs. Nodes

tion blackouts lasting equal to or longer than one TDMA round [118, p. 69].

4.5.3 Convict None (Do Nothing) Strategy

Unlike the other strategies, the Convict None strategy is somewhat sensitive to permanent faults as well as transient faults. As shown in the Figure 25 box plot, the Convict None strategy was sensitive to the Permanent Link Fault rate in addition to the Single Event Upset rate and the Bit Error Rate. Since the Convict None strategy never removes nodes, these permanent faults will remain in the system until the end of the mission. However, even this strategy was not sensitive to the permanent Single Event Latchup faults, which were modeled as symmetric faults for membership. This shows that fault diagnosis strategies should concentrate on removing the permanent asymmetric faults.

Because no faulty nodes are ever removed, the relationship between the assumption failure rate and the number of nodes is interesting. The assumption failure rate actually increases if more nodes are added, after about seven nodes. Figure 29 shows the assumption reliability of the Convict None (do nothing) strategy vs. the number of nodes. The Y axis

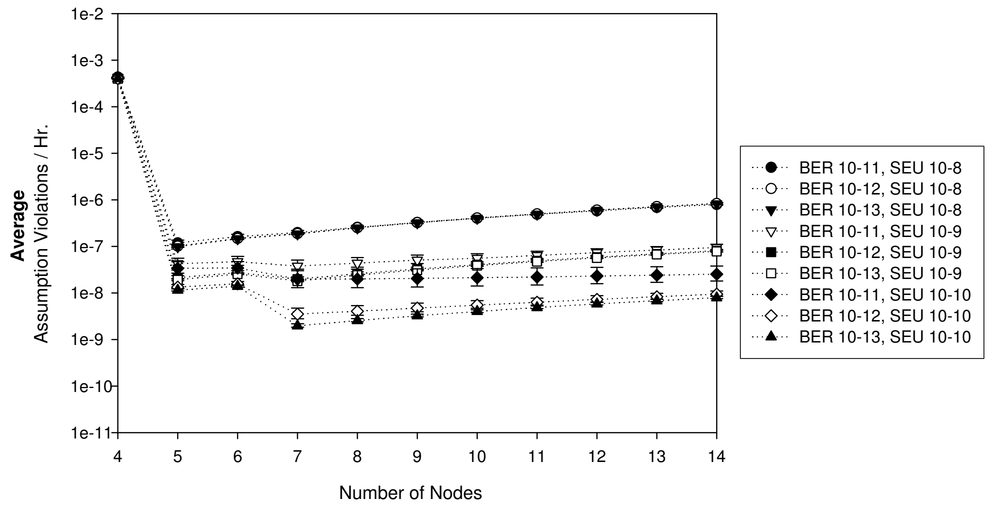


Figure 30. TTP/C Convict None: Average Assumption Reliability (SEU/BER) vs. Nodes

lists the number of maximum fault assumption violations per hour, on a log scale. Again, the four node configurations have a high assumption failure rate. But Figure 29 looks very different from the standard Convict All strategy in Figure 27, which removes nodes from the group after a single fault. There is a drastic decrease in the assumption failure rate between the 4 node configurations and the 5 and 6 node configurations. There is another jump between the 5 and 6 node configurations and the 7 or more node configurations. This is in contrast to the much more gradual decrease in the assumption failure rate for the standard strategy.

At seven nodes, the lowest assumption failure rate is about 10^{-9}. Since no faulty nodes are ever removed, permanent faulty nodes accumulate in the system and degrade the reliability. If the system is failing due to too many active faults, adding nodes will only increase the chance of failure by increasing the number of faulty components in the system. The exact tradeoff point for this particular strategy is explored further in the Death State Analysis section. This behavior is similar to the behavior described by Powell in his study of hypothetical failure rates [91]. Here, I show that this behavior can happen with failure rates expected in the real world.

Figure 30 describes the relationship between the Single Event Upset rate, Bit Error Rate, and assumption failure rate. There are three SEU rates and 3 BERs, for a total of nine combinations of fault rates. Figure 30 plots the average assumption failure rate per number of nodes for each of these nine combinations of fault rates. There were nine models for each of the nine combinations. The average assumption failure rate of these models was computed and graphed. Error bars at each point show the maximum and minimum values for the nine data points that were averaged.

The Single Event Upset rate appears to be the biggest factor influencing the assumption failure rate for the Convict None strategy, as Figure 30 shows. The three data sets for the SEU rate of 10^{-10} have the largest error bars. The effects of permanent link faults appear to stand out more for low SEUs. The contour lines in Figure 30 match Figure 29 fairly well.

It is useful to use the Convict None strategy results as a minimum level that new strategies must do better than. The Convict None strategy shows a lower variance than the standard strategy. Except for the four node configurations, the assumption failure rate was between 10^{-6} assumption violations/hr and 10^{-10} assumption violations/hr. This provides a helpful baseline for judging any other fault tolerance strategy. In the standard Convict All strategy, many of the configurations have an assumption failure rate of higher than 10^{-6}, and none of the configurations achieve a failure rate of lower than 10^{-8}. This could be unacceptable for safety-critical applications. Any new fault tolerance strategy should be required to outperform the baseline strategy in most cases. This shows the value of measuring assumption reliability. Hopefully, these results will encourage adoption of a new conviction strategy or development (and analysis) of a reintegration strategy that will make up for the shortcomings of the standard conviction strategy.

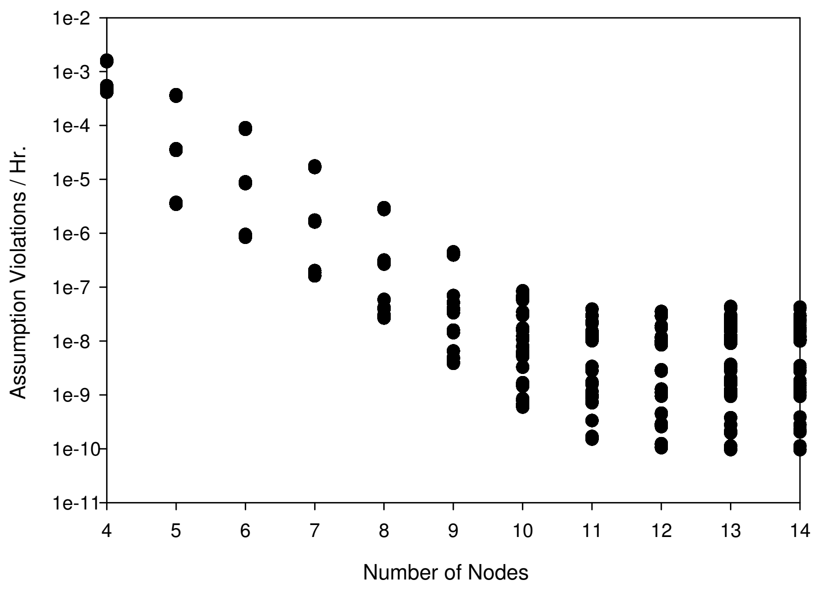


Figure 31. TTP/C Convict Some: Assumption Reliability vs. Nodes

4.5.4 Convict Some (Novel) Strategy

Figure 31 shows the assumption reliability of the Convict Some strategy vs. the number of nodes. The Y axis lists the number of maximum fault assumption violations per hour, on a log scale. The four node configurations have the highest assumption failure rate, and the twelve through fourteen node configurations have the lowest. Like the standard Convict All strategy, adding nodes decreases the assumption failure rate, and there is a relatively constant level of improvement from the four node configurations through the eight or nine node configurations.

Unlike the Convict All strategy, the Convict Some strategy does not appear to be limited by the effects of lightning. Because lightning strikes are treated as transient faults, the group can recover from temporarily losing half of it members for a round. Thus the Convict Some strategy would be expected to handle transient burst faults much better than the standard Convict All strategy.

After about eight or nine nodes in Figure 31, there appears to be some significant change

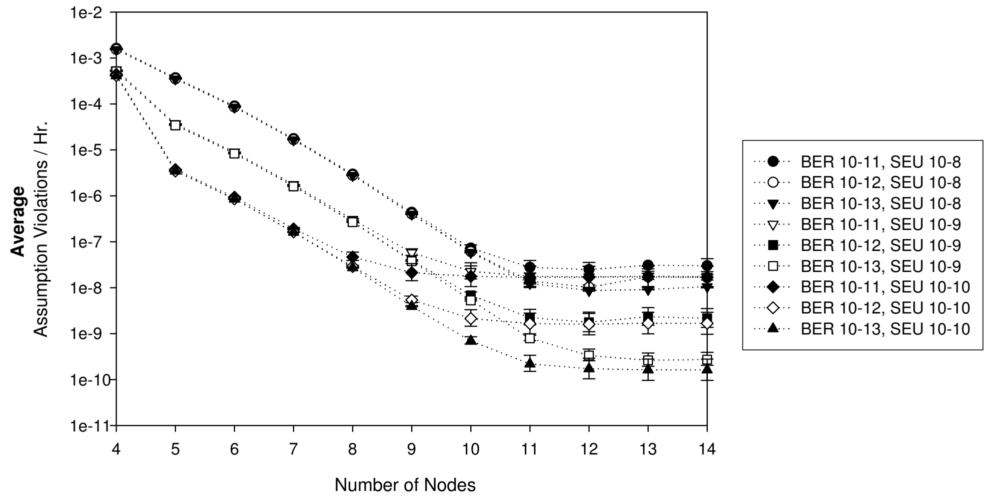


Figure 32. TTP/C Convict Some: Average Assumption Reliability (SEU/BER) vs. Nodes

in the factors affecting the assumption failure rate. Figure 32 shows the relationship between the Single Event Upset rate, Bit Error Rate, number of nodes, and assumption failure rate. Figure 32 plots the mean of each set of nine configurations that have the same SEU rate and BER. Up until about eight or nine nodes, the Single Event Upset rate appears to be the dominant factor influencing the assumption failure rate, similar to the Convict All strategy in Figure 28. After about nine nodes, the Bit Error Rate appears to be the dominant factor.

Even though the Convict Some strategy is the best overall, for the highest SEU rate of 10^{-8} the assumption failure rate does not go much below 10^{-8}. In these models, some Single Event Upset faults are asymmetric. Since the TTP/C maximum fault assumption only allows one asymmetric fault per consensus period (two TDMA rounds maximum), it is doubtful that a different diagnosis policy could improve the assumption failure rate much. Even if the asymmetric faulty node is convicted right away, that fault would still count against the maximum fault assumption. Therefore, lowering the SEU rate may be an important consideration.

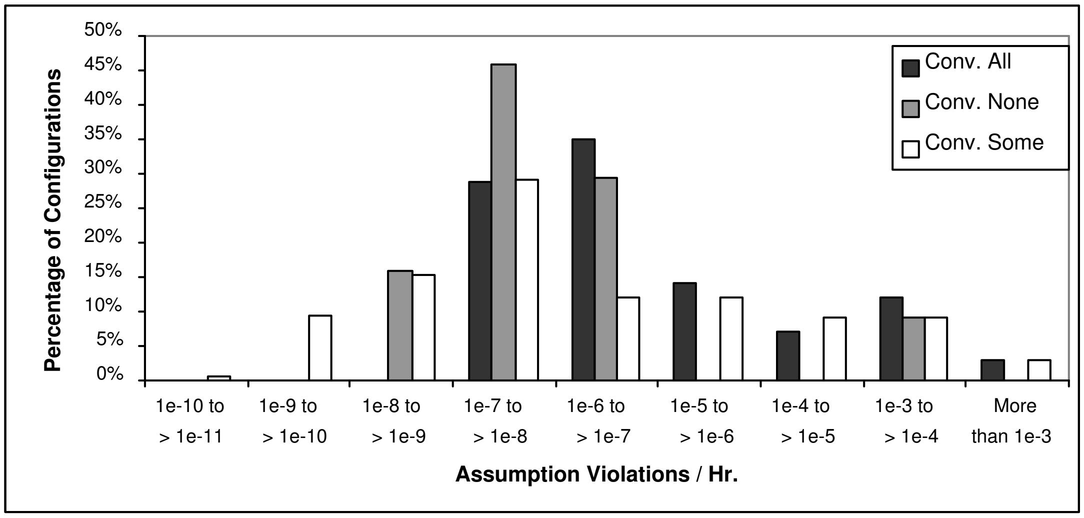


Figure 33. TTP/C Membership: Assumption Failure Rate Comparison

4.5.5 Comparison of Three TTP/C Diagnosis Strategies

Table 27 summarizes the assumption failure rate for all of the configurations for the three diagnosis strategies studied. Figure 33 plots the percentage of configurations that fall into each assumption failure rate bin.

Overall, the Convict Some strategy had the lowest assumption failure rate, and the standard Convict All strategy had the highest assumption failure rate, as shown in Figure 33 and Table 27. There were Convict Some configurations that achieved a three orders of magnitude decrease in assumption failure rate compared to the other two strategies (10^{-10} to $< 10^{-11}$ in Table 27). The 4, 5, and 6 node configurations all had high failure rates as compared to the 7 node and above configurations. The assumption failure rates for the Convict All and Convict Some strategies show more of a spread than the Convict None strategy, in Figure 33. This could be due to the conviction of good nodes (which does not occur in the Convict None strategy since no nodes are convicted), or the loss of redundancy due to convicted misdiagnosed transient faulty nodes.

In addition to comparing the overall performance of each strategy, it is interesting to

Table 27. TTP/C Membership Summary

Lists the number of configurations in each assumption failure rate bin

Assumption Violations/Hr.	Conv. All	Conv. None	Conv. Some
More than 10^{-3}	27 (3.0%)	81 (9.1%)	243 (3.1%)
10^{-3} to > 10^{-4}	108 (12.1%)	0	729 (9.1%)
10^{-4} to > 10^{-5}	63 (7.1%)	0	729 (9.1%)
10^{-5} to > 10^{-6}	126 (14.2%)	261 (29.3%)	972 (12.1%)
10^{-6} to > 10^{-7}	312 (35.0%)	408 (45.8%)	972 (12.1%)
10^{-7} to > 10^{-8}	255 (28.6%)	141 (15.8%)	2328 (29.0%)
10^{-8} to > 10^{-9}	0	0	1236 (15.4%)
10^{-9} to > 10^{-10}	0	0	756 (9.4%)
10^{-10} to > 10^{-11}	0	0	54 (0.7%)
Fewer than 10^{-11}	0	0	0

investigate the change in the assumption failure rate for each individual configuration. For example, the Convict Some strategy has a lower assumption failure rate than the Convict All strategy overall, but is it better for all of the configurations, or is it better for some configurations and worse for others?

Figure 34, Figure 35, and Figure 36, compare each pair of configurations with the same system and fault rate parameters for each of the three strategies. For example, a configuration might have 6 nodes, an SEL rate of 10^{-8}, an SEU rate of 10^{-10}, a Permanent Link fault rate of 10^{-7}, and a Bit Error Rate of 10^{-12}. The assumption reliability for the configuration with these parameters under (for example) the Convict Some strategy would be compared to the assumption reliability under the Convict All strategy.

There are a few details on how the amount of 'improvement' is measured. To measure improvement, first the error and the difference are computed. The assumption reliability measurements were made with the SURE tool, which gives an upper and lower bound on the assumption failure rate measurement. The *error* is defined as the upper bound minus the lower bound. There will be two error measurements, one for each of two configurations being compared. The *difference* is defined as the upper bound for the configuration using Strategy One minus the upper bound for the configuration using Strategy Two. The two upper bounds are compared (instead of the lower bounds) to be conservative. If the absolute value of the difference is less than either error measurement, then the amount of *improvement* is defined as zero.

The difference would be misleading as an ‘improvement’ measure, since what one really wants to measure is how many orders of magnitude the two configurations differ by. For example, the difference between a configuration with an upper bound of 10^{-3} assumption violations/hr compared to a configuration with an upper bound of 10^{-4} assumption violations/hr would be $9*10^{-4}$. In contrast, the difference between a configuration with an upper bound of 10^{-6} assumption violations/hr and a configuration with an upper bound of 10^{-9} assumption violations/hr would be $9.99*10^{-7}$. Although the first case has the larger difference, the second case shows more improvement. Therefore, the *improvement* is defined as (Upper bound Strategy Two)/(Upper bound Strategy One) if the upper bound for Strategy Two is the larger number, and (- Upper bound Strategy One)/(Upper bound Strategy Two) if the upper bound for Strategy One is the larger number. Therefore, the improvement is negative if Strategy Two has a lower assumption failure rate. Improvement is a unit-less quantity since it is the ratio of two assumption failure rate measurements.

Since negative numbers cannot be plotted on a log scale, Figure 34, Figure 35, and Figure 36 use the same linear scale for the Y axis. The number of nodes is listed on the X axis. These figures include all improvement measurements (i.e., there were not any measurements greater than 5000 or less than -5000).

For all pairs of configurations, the Convict Some strategy had either equal or better assumption reliability than the standard Convict All strategy, as shown in Figure 34. For four through seven nodes, there was zero improvement since all of the difference measurements were less than the error measurements for each of the configurations. From eight nodes on, the Convict Some strategy shows increasing improvement over the Convict All strategy (up to about three orders of magnitude). This shows that a smarter diagnosis strategy can improve assumption reliability without any negative effects, compared to the existing

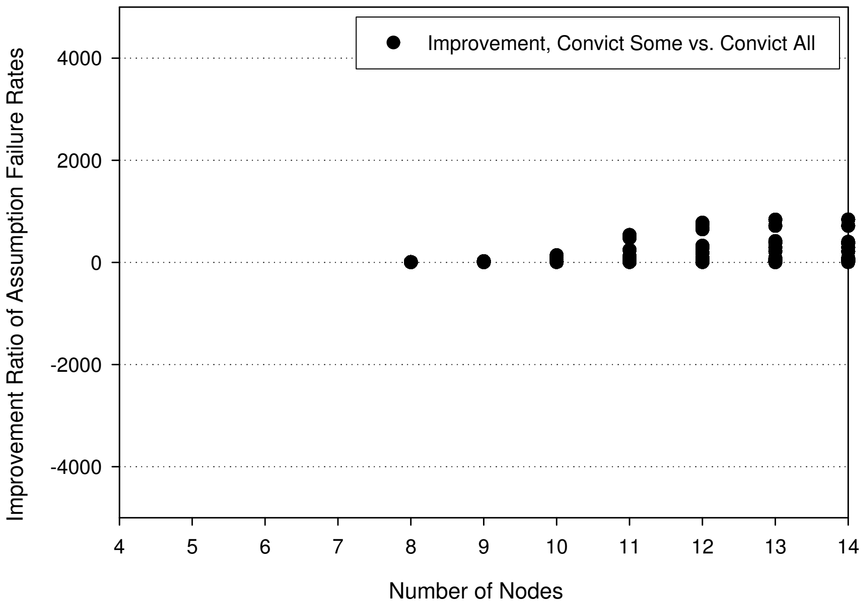


Figure 34. TTP/C: Improvement, Convict Some vs. Convict All

standard strategy.

The standard Convict All strategy was, for the most part, either about the same as or worse than the do nothing Convict None strategy, as shown in Figure 35. Especially for configurations with few nodes, the Convict All strategy actually decreased reliability. This shows that configurations with few nodes require a lenient (or extremely accurate) diagnosis algorithm. After about eight nodes or so, the two strategies have similar performance. For nine nodes or fewer, the Convict None strategy is best. For ten nodes and up, in some cases the Convict All strategy was better and in other cases the Convict None strategy was better. In other words, for ten nodes and up there are some positive points on Figure 35 and some negative points on Figure 35, although the magnitude of the improvement is small.

The comparison between the Convict Some strategy and the Convict None strategy is interesting. As Figure 36 shows, the Convict None strategy is better for four through eight nodes (especially for the five node configurations). At nine nodes, the improvement was zero for all pairs of configurations. For ten nodes and up, the Convict Some strategy is

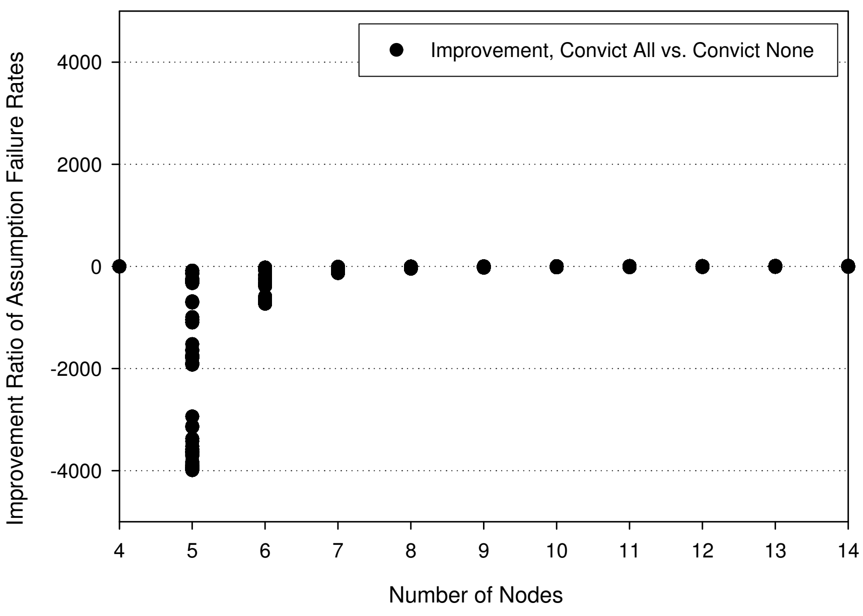


Figure 35. TTP/C: Improvement, Convict All vs. Convict None

Table 28. TTP/C Membership Dominant Failure

Lists the number of configurations according to which condition of the maximum fault assumption fails first

	Conv. All	Conv. None	Conv. Some
Active Faults (MFA.1)	0	747 (83.8%)	3159 (39.4%)
Too Few Nodes (MFA.2, MFA.3)	All	144 (16.2%)	4860 (60.6%)

better. Since the Convict Some strategy could misclassify nodes, this indicates that the penalty for mistakenly convicting transient faulty nodes can be high when there is little spare redundancy available. Also, Figure 36 correlates with the information in the Death State analysis section. For the Convict Some strategy, the dominant cause of assumption failure before about eight nodes is running out of redundancy, and after about eleven nodes the dominant cause of failure is too many active faults, with a mix for numbers of nodes in between.

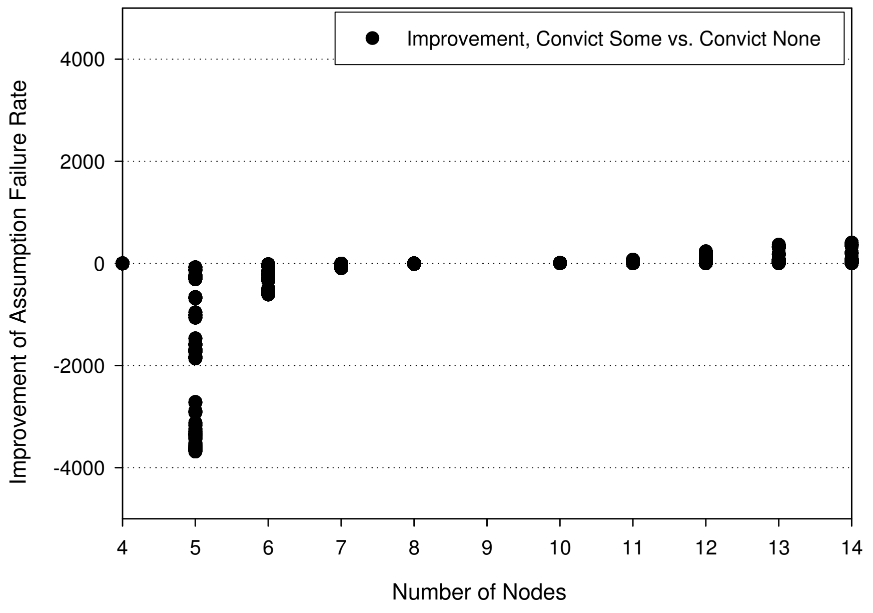


Figure 36. TTP/C: Improvement, Convict Some vs. Convict None

4.5.6 Death State Analysis

In all cases studied, the Convict All strategy failed by running out of redundancy, as shown in Table 28. In contrast, the Convict None strategy failed primarily due to too many active faults. The Convict Some strategy balanced the two risks best. The models were most sensitive to the transient faults (BER and SEU), with the Convict All and Convict Some strategies sensitive to permanent link faults and the Convict None strategy less so. All three strategies were insensitive to the SEL rate.

Further investigation shows that adding nodes might not improve reliability if the dominant cause of failure is too many active faults. In other words, adding a node means adding another potential faulty component. In the Convict None strategy, running out of redundancy was the dominant cause of failure for configurations with 4 to 6 nodes, but configurations with 9 or more nodes failed due to too many active faults. The configurations with the lowest failure rates had 9 or 10 nodes — configurations with more nodes had higher failure rates. In the Convict Some strategy, running out of redundancy was the

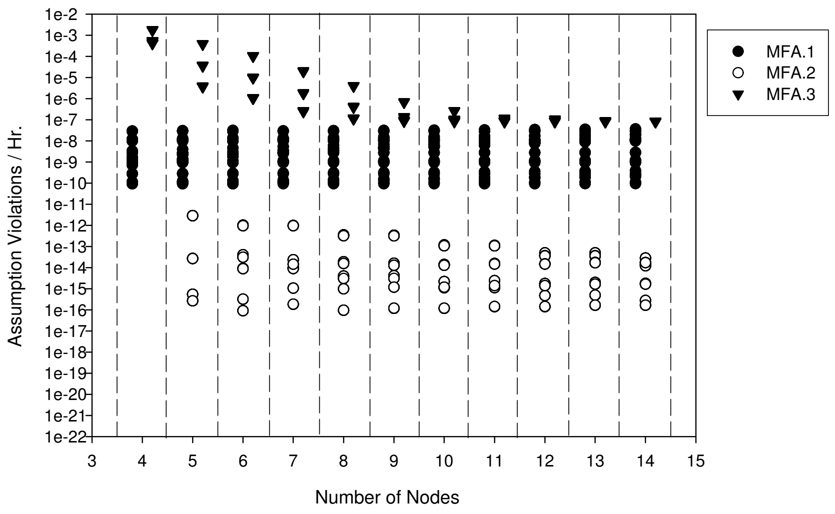


Figure 37. TTP/C Convict All: Dominant Assumption Failure Scatter Plot

dominant cause of failure for configurations with 4 to 10 nodes, but configurations with 13 or more nodes failed due to too many active faults. The configurations with the lowest failure rates had 13 or 14 nodes (the greatest number tested).

I hypothesize that adding nodes will eventually decrease reliability for algorithms whose maximum fault assumption includes a fixed term. For the TTP/C MFA.1, an asymmetric fault plus any other fault might cause the guarantees to be violated. Adding nodes will only increase the chance that a pair of faults occurs, so if MFA.1 is the dominant assumption violated, adding nodes is expected to decrease reliability. Many Byzantine fault tolerant algorithms are expected to include a fixed term in their MFAs, because for a round-based algorithm there must be at least $f + 1$ rounds to tolerate f Byzantine faults [35], and the total number of rounds is usually fixed.

Usually, one piece of the maximum fault assumption is the primary cause of failure. A successful strategy will mitigate faults that violate this piece of the maximum fault assumption. The SURE tools allow each piece of the maximum fault assumption to be tested separately. Table 25 lists the conditions tested. The conditions are tested in order, so states that fail to satisfy MFA.1 will not be checked for MFA.2 and MFA.3, for example.

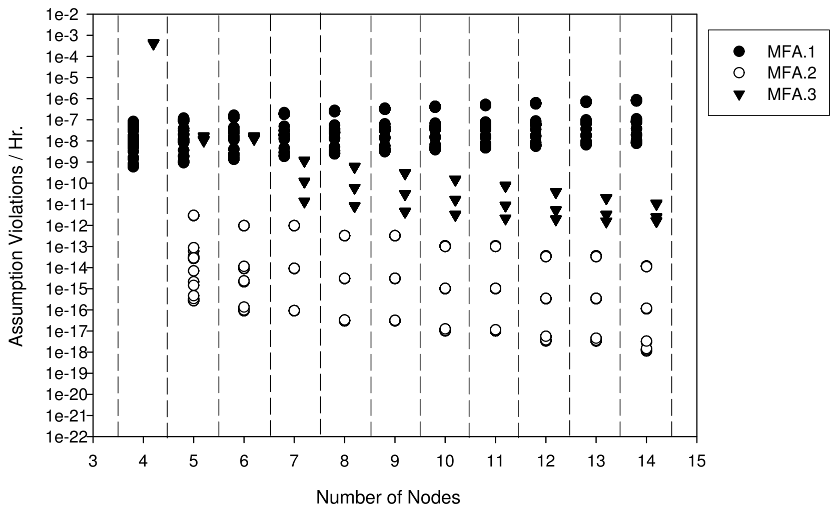


Figure 38. TTP/C Convict None: Dominant Assumption Failure Scatter Plot

For all of the diagnosis strategies, the assumption failure rate due to asymmetric faults (MFA.1) stays fairly constant in the range of about 10^{-7} to 10^{-10}. In the Convict None strategy, the assumption failure rate due to MFA.1 being violated actually increases slightly with the number of nodes, since no nodes are ever convicted. Figure 37, Figure 38, and Figure 39 show scatter plots that depict the probability of failure for each MFA condition compared to the number of nodes. The assumption failure rate in assumption violations/hr is listed on the Y axis, and the number of nodes is listed on the X axis. Symmetric faults (MFA.2) do not appear to be much of a threat by themselves. The failure rate of MFA.2 is 10^{-11} or below for all of the configurations of all of the strategies. (Note that MFA.1 would count the case where there is at least one symmetric fault plus at least one asymmetric fault.)

In Figure 37, the failure rate of MFA.3 appears to track the failure rate of lightning-induced faults for the Convict All strategy. The failure rate of MFA.3 improves some initially, but stops improving once it reaches about 10^{-7}. In contrast, in Figure 38 for the Convict None strategy, MFA.3 is the dominant cause of failure only for the four node

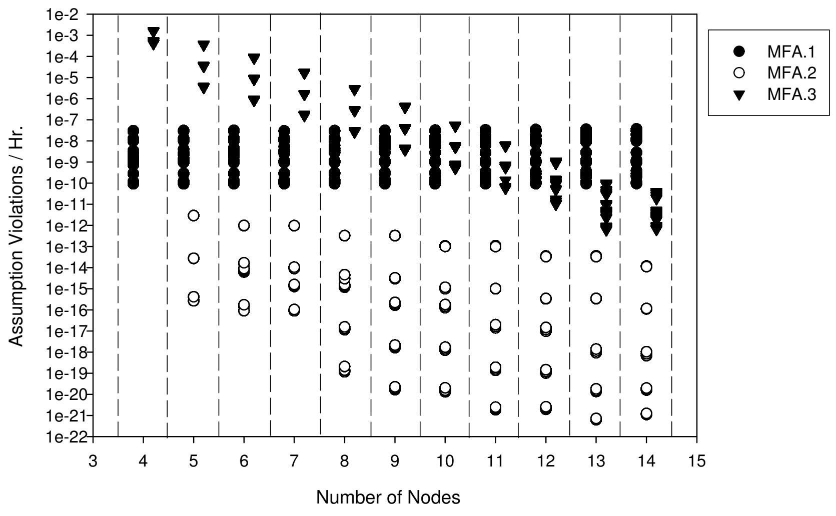


Figure 39. TTP/C Convict Some: Dominant Assumption Failure Scatter Plot

configurations and some of the five and six node configurations.

Figure 39 portrays the Convict Some strategy, where there is a trade-off between MFA.1 and MFA.3. Initially, adding nodes helps reduce the chance of failure due to not enough redundancy (MFA.3). Eventually, however, the chance of an asymmetric fault plus another simultaneous fault catches up (MFA.1). This confirms the suspicion from Figure 32 that Single Event Upset faults might be a limiting factor to how low the assumption failure rate can go.

4.5.7 Sensitivity Analysis

This section explores sensitivity to some of the system parameters assumed, and to the ability of the diagnosis strategy to handle asymmetric faults, lightning, and to handle losing some good nodes due to conviction. To keep the amount of work feasible, and to examine the strategy with the biggest response to lightning, the sensitivity analysis is performed on the Convict All strategy. Of the physical fault types, Single Event Upset faults had the most influence on the assumption failure rate of the TTP/C diagnosis strategies. The size and capacity of the integrated circuit will influence the number of upsets/device-hr, since larger circuits may have more bits that SEU faults can affect. Also, the chance of an SEU causing an asymmetric fault is an important parameter, since asymmetric faults are the most difficult to tolerate.

The first part of sensitivity analysis studies two different memory sizes (64K and 256K) and four different percentages of memory affected by asymmetric SEUs (0, 15%, 50%, and 100%). Also, sensitivity analysis investigates three probabilities of convicting asymmetric faulty nodes (1.0, 0.95, and 0.90) and three probabilities of good node conviction in the event of an asymmetric fault ($1/G_N$, 0.25, and 0.50). At maximum, half of the good nodes in the group could be convicted in the event of an asymmetric fault, if the asymmetric faulty node is in the majority clique. The SEL rate was kept constant at the highest studied (10^{-6}). There were (3 SEUs * 3 Perm. Link fault rates * 3 BERs * 8 memory combinations * 3 Prob. Conv. Asym * 3 Prob. Conv. Good) = 1944 configurations for membership.

The second part of sensitivity analysis examines sensitivity to lightning. In the original model, two lightning strikes would cause the assumptions to fail for the standard Convict All strategy, since each strike half of the group will be convicted and removed. This section repeats the sensitivity analysis tests for the Convict All strategy, without lightning.

Figure 40 shows that the TTP/C Convict All strategy is sensitive to both the number of bits susceptible to SEU faults and to the percentage of SEU faults that are asymmetric. The eight boxes at the far right of Figure 40 show the eight combinations of bytes per node and asymmetric SEU percentage. Interaction effects are expected between these two new parameters, so a separate box plot was generated for each combination of the parameters. This strategy appears more sensitive to the percentage of SEU faults that are asymmetric vs. the total number of bytes in the node. Therefore integrated circuit size could increase, as long as the percentage of SEU faults that have an asymmetric manifestation is kept small.

The Convict All strategy appears relatively insensitive to the probability of correctly convicting an asymmetric faulty node. This may be because the primary cause of assumption

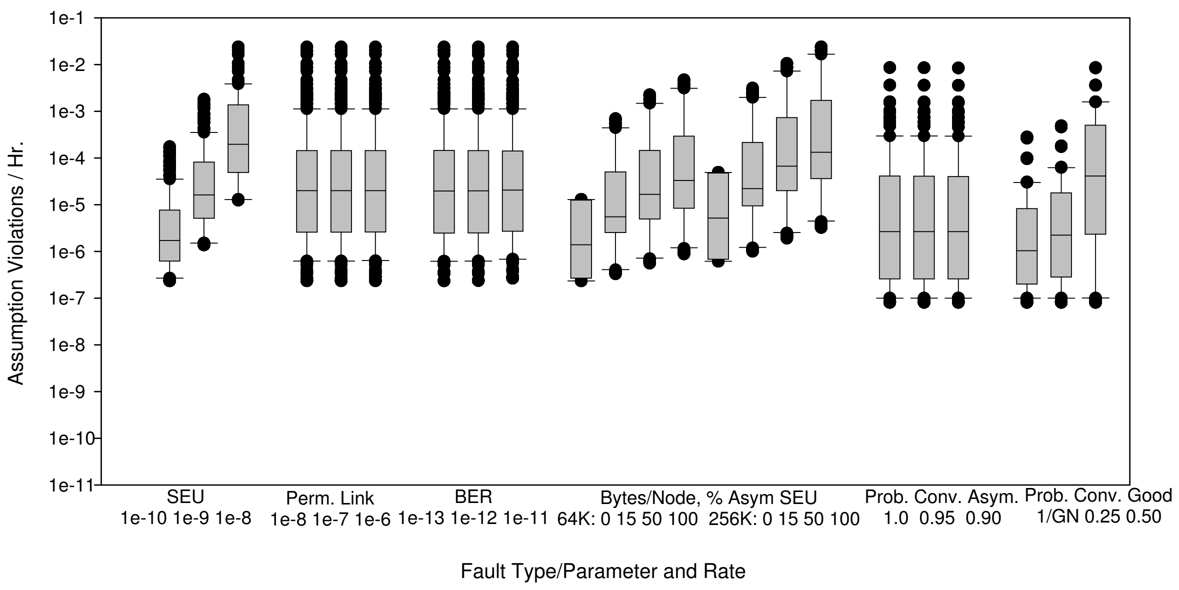


Figure 40. TTP/C Convict All: Sensitivity Analysis Box Plot

failure for the Convict All strategy is running out of redundancy (see the Death State Analysis section for more details). However, this also indicates that a practical fault diagnosis algorithm performs fairly well with respect to an ideal algorithm.

The assumption failure rate is sensitive to the probability of convicting good nodes. The chance of convicting good nodes depends on the asymmetric fault manifestation, i.e. how many receivers are in the minority clique. In the worst case an asymmetric fault will divide the current group in half. The probability of this occurring in practice is not known, but it is important to point this out as a concern. This could also be a design issue — the effects of an asymmetric fault may depend on the spatial distribution of the nodes. For example, a weak bus driver may be able to transmit to nearby nodes but not faraway nodes. If a star coupler is used, this could be a star coupler issue.

Without lightning, the Convict All strategy performs better, as shown in Figure 41. Figure 40 and 41 have the same Y axis scale so that the figures may be easily compared. The best configurations have an assumption failure rate of around 10^{-10} assumption viola-

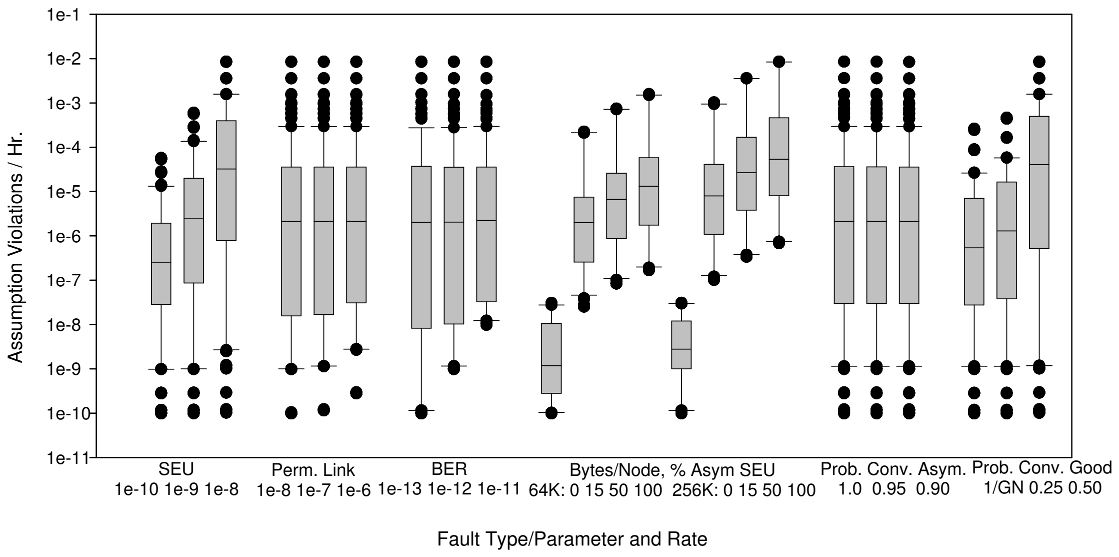


Figure 41. TTP/C Convict All: Sensitivity Analysis, No Lightning, Box Plot

tions/hr instead of 10^{-7} assumption violations/hr. There is a large improvement for configurations where there are no asymmetric SEU faults (the 64K / 0% Asym. SEU box plot and the 256K / 0% Asym. SEU box plot). However, it seems there is less improvement for configurations that experience asymmetric SEU faults. The best of these configurations have an assumption failure rate around 10^{-7} assumption violations/hr.

Without lightning, the Convict All strategy is still most sensitive to the same faults as before (the amount of asymmetric SEU faults, the SEU rate and the probability of convicting good nodes). Also, Figure 41 shows that without lightning there is some sensitivity to the Bit Error Rate and the Permanent Link fault rate.

4.5.8 Other Modeling Information

Since the Convict Some strategy had the largest models, a summary of execution time information for this strategy is given here. The execution time depends on the size of the model and (for SURE) the amount of model pruning. In SURE, the designer can specify the pruning level (using the PRUNE constant) and the loop truncation level (using the TRUNC constant). For the pruning level, any path with probability less than PRUNE will be removed from the model. For truncation, any looped path that includes more than TRUNC times through the loop is removed from the model. Pruning and loop truncation can greatly reduce the model solution time. However, if the pruning level or loop truncation level is too high, the SURE tool will be unable to calculate reliability bounds. A pruning level of 10^{-16} was used for configurations with four through twelve nodes, and a pruning level of 10^{-17} was used for the thirteen and fourteen node configurations and for a handful of twelve node configurations for which the 10^{-16} pruning level was too severe. The SURE tool aggregates death states according to which piece of the maximum fault assumption was violated first. This greatly reduces the number of death states, since there are only three death states after aggregation. For a few of the models it was also necessary to prune some of the states at the ASSIST level, which requires more manual work but can reduce the solution time for the SURE tool and was needed for some cases where the Markov model generated by the ASSIST tool was too large to be read by the SURE tool.

Table 29 summarizes some of the performance statistics for the TTP/C Convict Some strategy. Solution times are given in CPU seconds. For some of the executions, the computer clock rolled over, giving a negative solution time. While the exact value of clock rollover is unknown, from the data it appears the clock rolls over at about 2400 CPU seconds (forty minutes). These configurations were excluded from the average, and the number of these configurations is listed in the table. While the average execution time was about 120 seconds (two minutes), there was quite a bit of variation. Most models were solved in under 10 seconds, but some models took over twenty minutes. Models with higher BERs (10^{-11}) overall took the longest to solve. Starting at thirteen nodes, the models were additionally pruned at the ASSIST stage, so that is why the number of states and transitions decrease from the twelve node to thirteen node models. The pruning level was

Table 29. TTP/C Convict Some Model Size and Execution Time

Nodes	States	Transitions	Avg. Execution Time (CPU sec.)	Time Rollovers
4	128	1121	5	0
5	523	4974	11	0
6	1533	15434	15	0
7	3648	38471	17	0
8	7668	83615	22	0
9	14568	163226	22	0
10	25813	295434	27	0
11	43148	502870	26	0
12	69018	816621	28	9
13	55946	662142	129	45
14	90486	1090615	123	54

not optimized per model, so shorter times may be possible.

In addition, there is a certain amount of model 'read time' required. The ASSIST tool outputs the Markov model to a file and the SURE tool must read this file back in before it can solve the model. The read time is not included in the calculations, but increases with the number of states and transitions (and thus increases with the number of nodes). The fourteen node configurations had a read time of about 550 seconds, or about 9 minutes. The amount of time the ASSIST tool needed to generate the model was not recorded for each configuration, but was generally not more than a few seconds.

4.6 SPIDER Membership Results

Table 30 lists the SPIDER maximum fault assumption. SPIDER has two types of components, Bus Interface Units (BIUs) and Redundancy Management Units (RMUs). The SPIDER maximum fault assumption (MFA) is stated in terms of the sets G, B, S, and A, which denote the sets of good, benign, symmetric, and asymmetric nodes respectively [75]. E_{BIU} and E_{RMU} refer to the sets of eligible voters for BIUs and RMUs, respectively.

Treating the Redundancy Management Units as separate entities is one of SPIDER's greatest strengths. If at least three RMUs are used, the receiving Bus Interface Units can vote the received frame values, unlike TTP/C and FlexRay nodes which will accept any cor-

Table 30. SPIDER Maximum Fault Assumption [75]

SPIDER MFA.1. $2\|G \cap E_{RMU}\| > \|E_{RMU} \backslash B\|$ for all RMUs RMU, and
SPIDER MFA.2. $2\|G \cap E_{BIU}\| > \|E_{BIU} \backslash B\|$ for all BIUs BIU, and
SPIDER MFA.3. $\|A \cap E_{RMU}\| = 0$ for all RMUs RMU, or $\|A \cap E_{BIU}\| = 0$ for all BIUs BIU

rectly formatted frame. SPIDER guarantees Byzantine fault tolerant agreement on frame contents, in addition to agreement on the set of members. Also, the RMUs aid in the fault detection process, since RMUs can pass on special frames indicating that the BIU sender failed to correctly send a frame during its time slot.

However, the results show that elevating the RMUs to first class entities changes the assumption reliability quite a bit. There are a limited number of RMUs since installing another RMU requires costly wiring. From the results, the assumption failure rate appears to be dominated by the failure rate of the RMUs. Unlike TTP/C and FlexRay nodes, adding SPIDER BIUs has almost no impact on the assumption failure rate. Also, the penalty for misdiagnosing a permanent fault is high, even though the models are most sensitive to transient faults. This indicates that transient faults have a greater impact if the amount of redundancy is already compromised by the undetected permanent fault.

This section examines the three strategies individually, then compares them to each other. The Convict None strategy performed the best overall, which was surprising. The Convict Some strategy was second best overall, and the Convict All strategy had the poorest assumption reliability overall. The models were most sensitive to transient faults, although configurations for the Convict None strategy were also sensitive to the permanent fault rates. The misclassification rate has a rather large effect on the assumption reliability for the Convict Some configurations, and sensitivity analysis is done comparing those results to a perfect classification scheme.

4.6.1 Physical Fault Sensitivity

Which faults are the models most sensitive to? This section examines box plots for each fault type, for each diagnosis strategy. For the SPIDER Convict All and Convict None strategies, there are five primary factors studied: the number of nodes, Single Event Latchup rate, Single Event Upset rate, Bit Error Rate, and Permanent Link fault rate. For the Convict Some strategy, two additional factors are investigated: the probability of a convicting a permanent faulty node, and the probability of convicting transient faulty nodes. Since SPIDER assures that no good node will ever be convicted, it is possible that a permanently faulty node might never be convicted if there is not enough evidence available to the group. In other words, a node is innocent until proven faulty by the whole group - if some of the group thinks the node is good and others think it is faulty, then in some cases the node must be considered good. These probabilities can be thought of as the (1 - false negative) rate and the false positive rate, respectively. Box plots are created for each primary factor investigated. For more detail on box plots and the statistics involved, please see the previous discussion in the Statistical Measures section. Sensitivity to the number of nodes is investigated in the next section, and additional factors are considered in the sensitivity analysis.

There is one set of box plots for each strategy: Figure 42 shows the standard Convict All strategy, Figure 43 shows the Convict None strategy, and Figure 44 shows the Convict Some strategy. The X axis lists the different levels of each primary factor. The assumption failure rate is listed on the Y axis in terms of assumption violations/hr using a log scale. The Y axis scales are the same for all three figures, so that the figures may be compared with each other.

The three strategies appear to have quite different assumption failure rates overall. Configurations with the Convict All strategy in Figure 42 show a very high assumption failure rate, from about 10^{-3} to 10^{-6}. In contrast, the configurations for the Convict None strategy in 43 have much lower assumption failure rates, ranging from about 10^{-6} to 10^{-9}. Config-

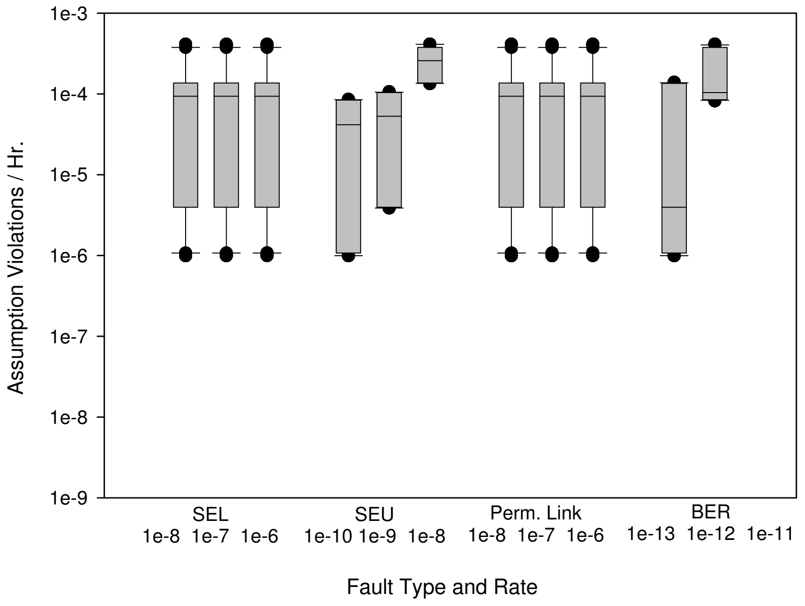


Figure 42. SPIDER Convict All, Box Plot

urations for the Convict Some strategy seem to be mixed, with failure rates ranging from 10^{-4} to about 10^{-8}. It is interesting that the Convict None strategy outperforms the Convict All strategy by such a large amount, and appears to outperform the Convict Some strategy in a many cases as well. This will be investigated further throughout the SPIDER analysis.

All of the diagnosis strategies seem most sensitive to the Single Event Upset rate and Bit Error Rate. The Convict All and Convict some strategies were especially sensitive to these two parameters, with about two orders of magnitude difference in assumption reliability between the 10^{-7} and 10^{-6} SEU rates and between the 10^{-13} and 10^{-12} BER rates. The assumption failure rate increase was not as pronounced for the Convict None strategy. This suggests that the Convict All and Convict Some strategies have difficulty handling transient faults; perhaps too many transient faulty nodes are being convicted. The Convict None strategy was also sensitive to the Single Event Latchup rate and the Permanent Link fault rate.

There are two additional factors for the SPIDER Convict Some model - the permanent

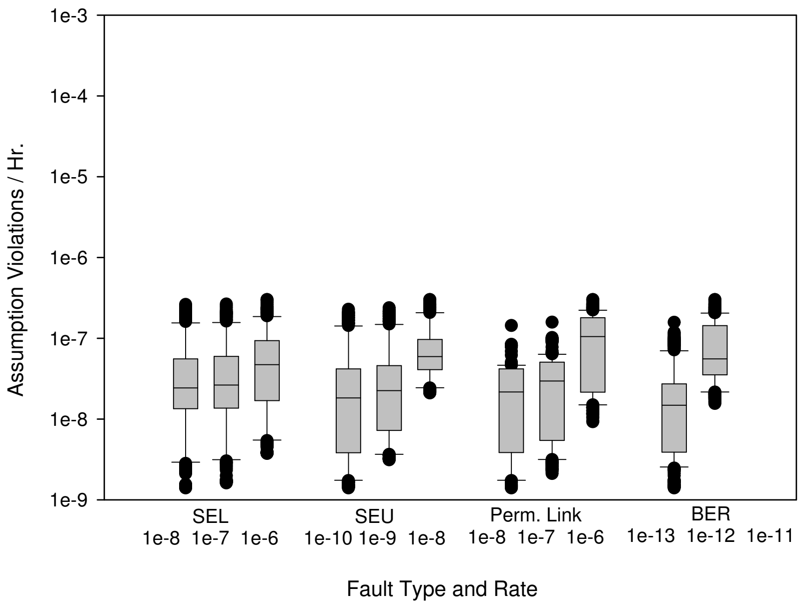


Figure 43. SPIDER Convict None, Box Plot

fault misclassification rate and the transient fault misclassification rate. These are represented in Figure 26 as the Probability of Convicting a Permanent faulty node and the Probability of Convicting a Transient faulty node. Convicting a permanent faulty node can be though of as a 'false negative', and convicting a transient faulty node can be though of as a 'false positive'. Ideally, the probability of convicting a permanent faulty node would be 100%, since the reliability will improve if permanent faulty nodes are removed. The probability of convicting a transient faulty node would be ideally be zero, since transient faults will expire and the affected node will return to normal the next communication round. Removing transient faulty nodes would reduce available redundancy.

As Figure 44 shows, the models are sensitive to the probability of convicting a permanent faulty node, but seem insensitive to the probability of convicting a transient faulty node as a primary factor. There is about half an order of magnitude decrease in the assumption failure rate for configurations with a 0.95 probability of convicting a permanent faulty node compared to a 0.99 probability. The same is true for configurations with a 0.90 probability

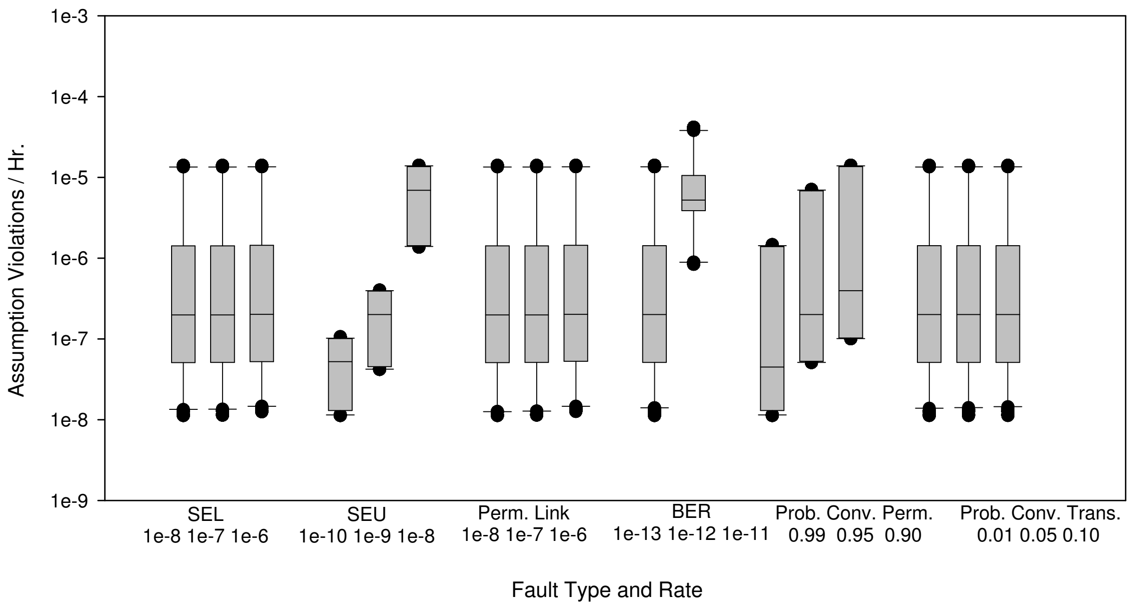


Figure 44. SPIDER Convict Some, Box Plot

compared to a 0.95 probability.

Table 31 lists factors and combinations of factors that influenced the assumption failure rate, from greatest influence to lowest influence. Effects due to a single factor are called ‘main effects’ and effects due to a combination of factors are called ‘interaction effects’. While the ordering is interesting, the actual magnitude of the effect is not very useful in this case since the assumption failure rate measurements had to be scaled up to perform the analysis. Scaling preserves the order, however. For the SPIDER studies, only two factor interaction effects were studied, since some of the three factor interaction effects were too small for the analysis tools to create confidence intervals for. If some of the three factor interaction effects are too small to analyze, then all of the three factor effects are excluded from the analysis since the statistical model becomes unbalanced if only some three factor effects are considered.

For all three strategies, the Bit Error Rate had the largest effect on the assumption failure rate, followed by the Single Event Upset rate for the Convict All and Convict None strate-

Table 31. SPIDER Membership: Factors, in Order of Influence

Convict All	Convict None	Convict Some
BER	BER	BER
SEU	PermLink	SEU
SEU*BER	PermLink*BER	Prob. Conv. Perm.
Nodes*SEU	SEU	BER*Prob. Conv. Perm.
Nodes*BER	SEL	SEU*Prob. Conv. Perm.
Nodes	SEU*PermLink	SEU*BER
	SEL*BER	Nodes*SEU
	Nodes	Nodes*BER
	SEL*SEU	
	Nodes*BER	
	Nodes*SEL	
	Nodes*SEU	

gies and the Permanent Link fault rate for the Convict None strategy. The Convict Some factors are interesting. The probability of convicting a transient faulty node did not appear to affect the assumption failure rate, even through interaction effects with the fault rates. However, the probability of convicting a permanent faulty node had some interesting interaction effects with the transient fault rates (BER and SEU). One might expect interaction effects between the probability of convicting permanent faulty nodes and the permanent fault arrival rates (SEL and PermLink). It seems that if a permanent fault is present in the system, then the SPIDER system becomes much more sensitive to transient faults.

The next sections discuss each of the strategies in more detail, but one interesting observation is that all of the configurations from all strategies are fairly insensitive to the number of Bus Interface Units. A SPIDER Bus Interface Unit is similar to a node in the FlexRay or TTP/C. This is very different from FlexRay and TTP/C, which were both sensitive to the number of nodes. This is due to the fact that SPIDER treats its Redundancy Management Units (similar to a star coupler) as first class entities, whereas FlexRay and TTP/C do not treat star couplers as first class entities and instead map star coupler faults back to nodes. As the Death State Analysis section will explore, faults in the Redundancy Management Units are a prime contributor to assumption violations, so adding more Bus Interface Units

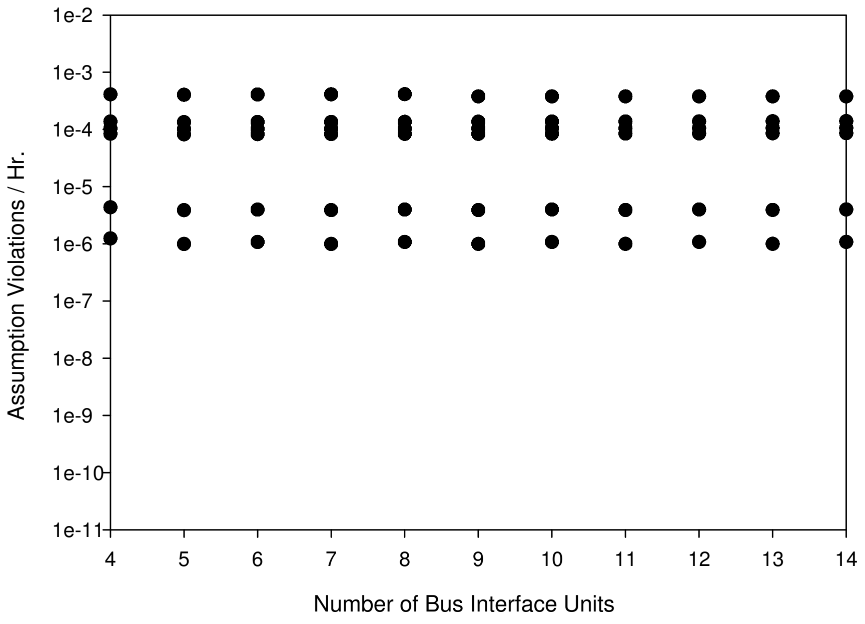


Figure 45. SPIDER Convict All: Assumption Reliability vs. BIUs

may not change the assumption failure rate much.

4.6.2 Convict All (Standard) Strategy

Figure 45 plots the assumption failure rate of the Convict All strategy vs. the number of Bus Interfaces Units (BIUs). The Y axis lists the number of maximum fault assumption violations per hour, on a log scale. The configurations appear to fall along five or so horizontal bands, ranging from an assumption failure rate of about $5*10^{-4}$ to 10^{-6}. This indicates that the assumption reliability depends heavily on one or two of the fault types. From the factors discussed in Table 31, the Single Event Upset rate and Bit Error Rate are the two fault types of interest.

The scatter plot of all configurations in Figure 45 maps fairly well to a plot of the average of configurations with the same Single Event Upset rate and Bit Error Rate. Figure 46 plots the average assumption failure rate for groups of configurations with the same Single Event Upset rate and Bit Error Rate. There are three SEU rates and two BERs, for a total of six

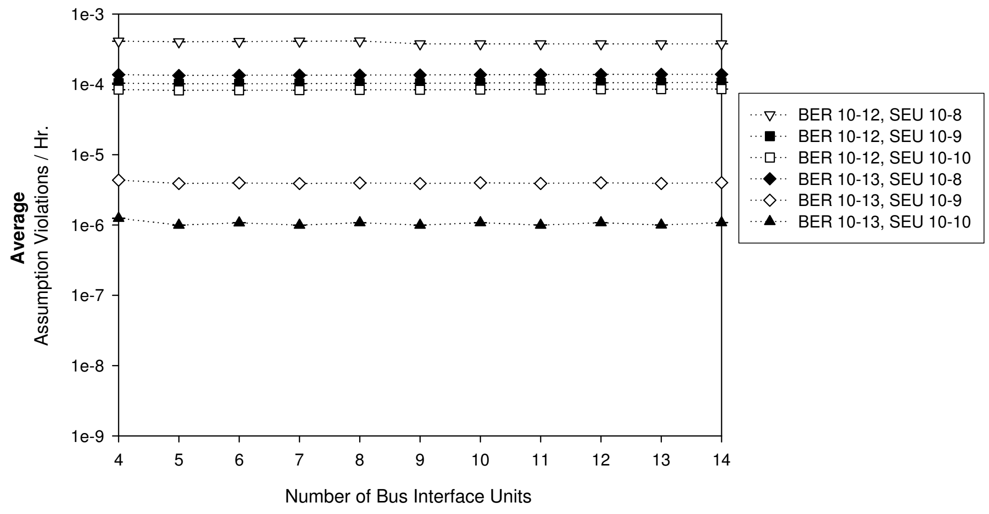


Figure 46. SPIDER Convict All: Average Assumption Reliability (SEU/BER) vs. BIUs

combinations of fault rates. There were nine models for each of the six combinations. The average assumption failure rate of these models is computed and graphed in Figure 46. Error bars at each point show the maximum and minimum values for the nine data points that were averaged. Since the error bars are extremely small (they hardly show on the graph, except as a line through the center of the symbols) , this indicates that the Single Event Upset rate and Bit Error Rate predict the assumption failure rate well. A low BER (10^{-13}) only appears beneficial for lower SEU rates, since configurations with an SEU rate of 10^{-8} had a high assumption failure rate regardless.

4.6.3 Convict None (Do Nothing) Strategy

Unlike the other strategies, the Convict None strategy is sensitive to all of the types of faults. As shown in the Figure 43 box plot, the Convict None strategy was sensitive to the Permanent Link Fault rate and the Single Event Latchup rate in addition to the Single Event Upset rate and the Bit Error Rate. On the positive side, this means that lowering any of the

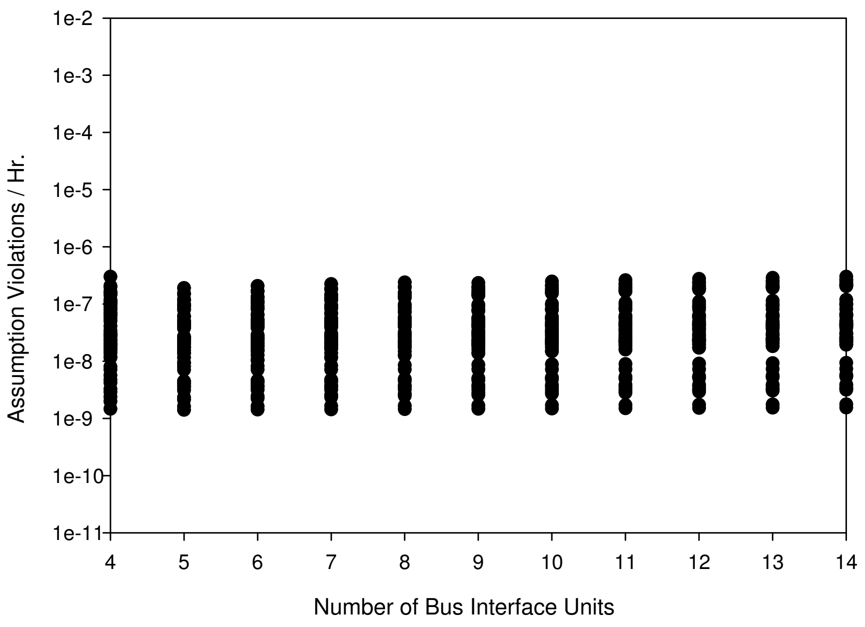


Figure 47. SPIDER Convict None: Assumption Reliability vs. BIUs

fault rates is likely to reduce the assumption failure rate some. On the downside, since there seems to be no dominant fault type, reducing one of the fault rates might only give a small reduction in the assumption failure rate.

The Convict None strategy appears fairly insensitive to the number of Bus Interface Units. Figure 47 shows the assumption failure rate of the Convict None (do nothing) strategy vs. the number of Bus Interface Units (BIUs). The Y axis lists the number of maximum fault assumption violations per hour, on a log scale. The assumption failure rate appears evenly spread between about $5*10^{-7}$ assumption violations/hr. and 10^{-9} assumption violations/hr.

Figure 48 plots the average assumption failure rate for groups of configurations with the same Single Event Upset rate and Bit Error Rate. There are three SEU rates and two BERs, for a total of six combinations of fault rates. Figure 48 plots the average assumption failure rate per number of BIUs for each of these six combinations of fault rates. There are nine models for each of the six combinations. The average assumption failure rate of

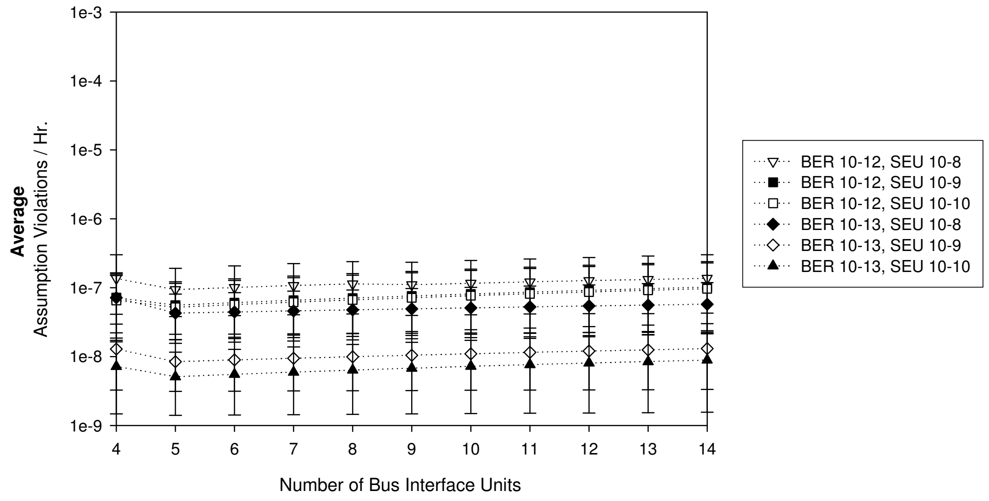


Figure 48. SPIDER Convict None: Average Assumption Reliability (SEU/BER) vs. BIUs

these models is computed and graphed. Error bars at each point show the maximum and minimum values for the nine data points that were averaged.

Unlike the other strategies, the influence of the Permanent Link fault rate and Single Event Latchup rate appears to be significant, in addition to the Bit Error Rate and the Single Event Upset rate. The error bars in Figure 48 span about an order of magnitude or more. Therefore, predicting the assumption failure rate from just the Bit Error Rate and Single Event Upset rate will probably give an inaccurate estimate. Since Table 31 shows that the Permanent Link fault rate was a stronger influence on the assumption failure rate than the SEU rate, it might be beneficial to graph this strategy in terms of the BER and Permanent Link fault rate combinations. Here, the graphs for all three strategies were plotted the same way. The Bit Error Rate appears to have a somewhat stronger influence on the assumption failure rate than the Single Event Upset rate for cases where the SEU rate is 10^{-10} or 10^{-9}.

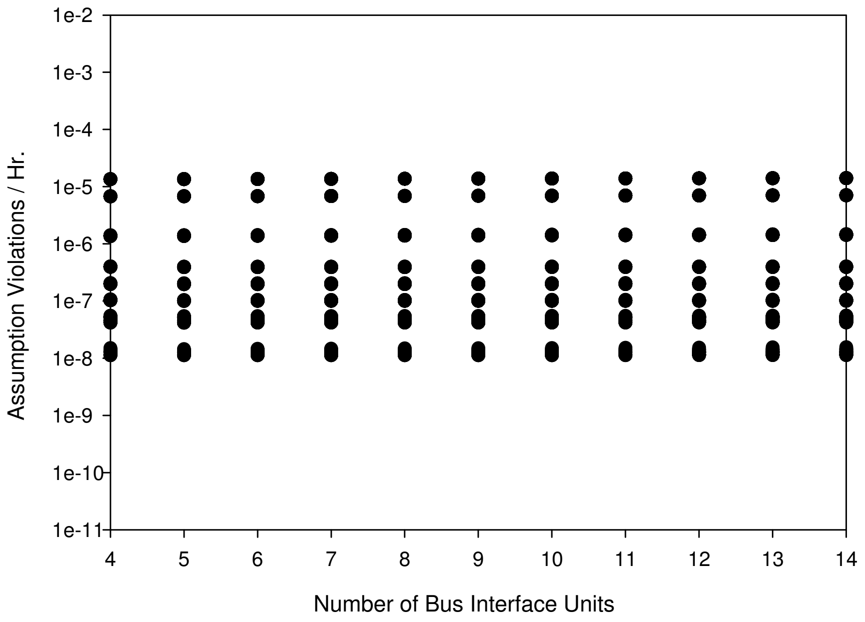


Figure 49. SPIDER Convict Some: Assumption Reliability vs. BIUs

4.6.4 Convict Some (Novel) Strategy

The Convict Some strategy had the widest range of assumption failure rates, as shown in Figure 49. Figure 49 shows the assumption failure rate of the Convict None (do nothing) strategy vs. the number of Bus Interface Units (BIUs). The Y axis lists the number of maximum fault assumption violations per hour, on a log scale.

Figure 50 plots the average assumption failure rate for groups of configurations with the same Single Event Upset rate and Bit Error Rate. There are three SEU rates and two BERs, for a total of six combinations of fault rates. Figure 50 plots the average assumption failure rate per number of BIUs for each of these nine combinations of fault rates. Since the Convict Some strategy also models two additional misclassification probabilities, there were 81 models for each combination. The average assumption failure rate of these models was computed and graphed. Error bars at each point show the maximum and minimum values for the 81 points.

For the Convict Some strategy, the probability of convicting permanent faulty nodes

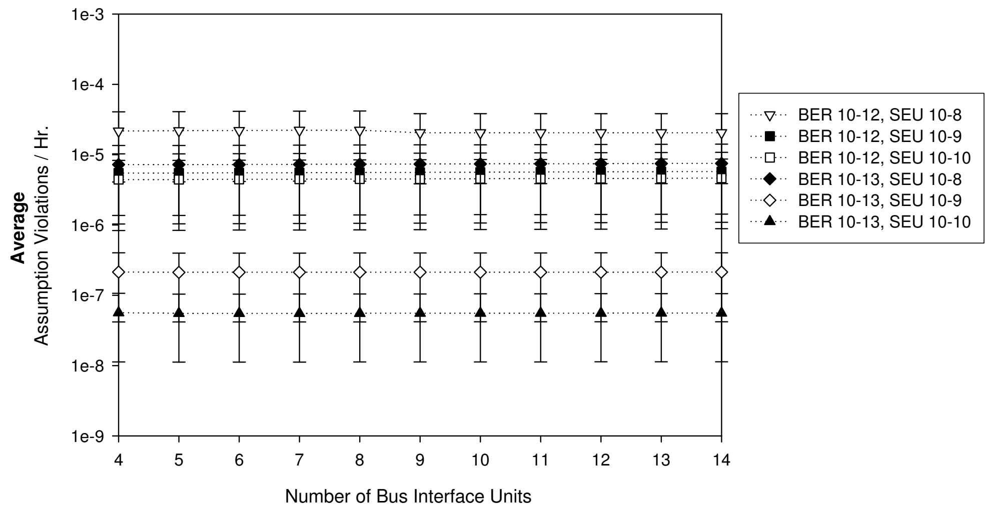


Figure 50. SPIDER Convict Some: Average Assumption Reliability (SEU/BER) vs. BIUs

might also be important in addition to the BER and SEU rates. From Table 31, the BER and SEU rates are the two most influential factors, followed by the probability of convicting a permanent faulty node. The error bars in Figure 50 are still fairly large, about an order of magnitude or so. For example, configurations with the lowest assumption failure rates had an SEU rate of 10^{-10}, a BER of 10^{-13}, and a 0.99 probability of convicting permanent faulty nodes. There was about an order of magnitude difference between the configurations with the same fault arrival rates, but a 0.90 probability of convicting permanent faulty nodes.

4.6.5 Comparison of Three SPIDER Diagnosis Strategies

Figure 51 and Table 32 compare all three strategies according to the number of configurations from each strategy aggregated into assumption failure rate bins. Figure 51 and Table 32 include all of the configurations; there are no configurations with an assumption failure rate greater than 10^{-3} or less than 10^{-9}. In Figure 51, the assumption failure rate is listed along the bottom on the X axis, and the percentage of configurations that fall within each

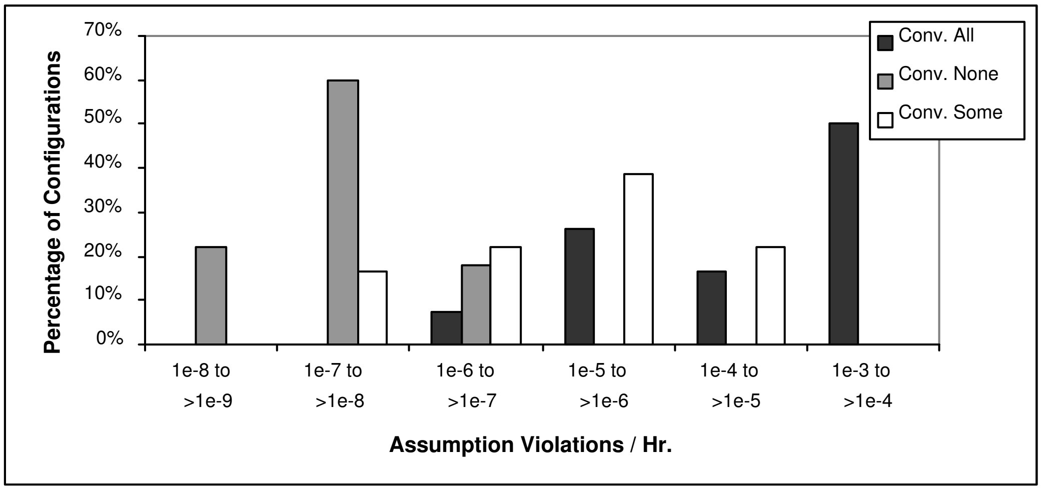


Figure 51. SPIDER Membership: Assumption Failure Rate Comparison

Table 32. SPIDER Membership Summary

Lists the number of configurations in each assumption failure rate bin

Assumption Violations/Hr.	Conv. All	Conv. None	Conv. Some
10^{-3} to > 10^{-4}	297 (50.0%)	0	0
10^{-4} to > 10^{-5}	99 (16.7%)	0	1188 (22.2%)
10^{-5} to > 10^{-6}	155 (26.1%)	0	2079 (38.9%)
10^{-6} to > 10^{-7}	43 (7.2%)	106 (17.9%)	1188 (22.2%)
10^{-7} to > 10^{-8}	0	356 (59.9%)	891 (16.7%)
10^{-8} to > 10^{-9}	0	132 (22.2%)	0

assumption failure rate range is listed along the Y axis. The percentage of configurations is used since the Convict Some strategy has more total configurations, due to the additional probabilities of misclassification.

The SPIDER strategies are interesting since, overall, the Convict None strategy is the best performer. This is in contrast with TTP/C where the Convict Some strategy was best. Ultimately, SPIDER may be limited by the number of Redundancy Management Units. All configurations here had three RMUs. Unfortunately, adding an RMU is more expensive than adding a Bus Interface Unit, because of the additional wiring that is needed. Adding an RMU in SPIDER would be similar to adding another star coupler in TTPC or FlexRay. The assumption failure rate also seems to be very dependent on the probability of convicting a permanent faulty node. Otherwise, all other parameters being equal, the Convict Some strategy should at least have the same assumption failure rate as the Convict None strategy, if not a lower assumption failure rate.

How do the three strategies compare per configuration? For example, the Convict None strategy seems the best overall, but is it the best for each configuration too? Figure 52, Figure 53, and Figure 54 compare each pair of configurations with the same system and fault rate parameters for each of the three strategies. For example, a configuration might have 6 nodes, a SEL rate of 10^{-8}, an SEU rate of 10^{-10}, a Permanent Link fault rate of 10^{-7}, and a Bit Error Rate of 10^{-12}. The assumption failure rate for the configuration with these parameters under (for example) the Convict Some strategy would be compared to the assumption failure rate under the Convict All strategy. Each of these figures use the same Y axis and a linear Y axis scale, since negative numbers cannot be plotted on a log scale. The number of Bus Interface Units is listed on the X axis. For SPIDER, changing the number of Bus Interface Units does not have much effect on the improvement metric, unlike FlexRay and TTP/C where the improvement varied according to the number of nodes.

A special calculation for 'improvement' was used here. To measure improvement, first the error and the difference were computed. The assumption failure rate measurements were made with the SURE tool, which gives an upper and lower bound on the reliability. The *error* is defined as the upper bound minus the lower bound. There will be two error measurements, one for each of two configurations being compared. The *difference* is defined as the upper bound for the configuration using Strategy One minus the upper bound for the configuration using Strategy Two. The two upper bounds are compared (instead of the lower bounds) to be conservative. If the absolute value of the difference is less than either error measurement, then the amount of *improvement* is defined as zero.

The difference is misleading as an 'improvement' measure, since what one really wants to measure is how many orders of magnitude the two configurations differ by. Therefore,

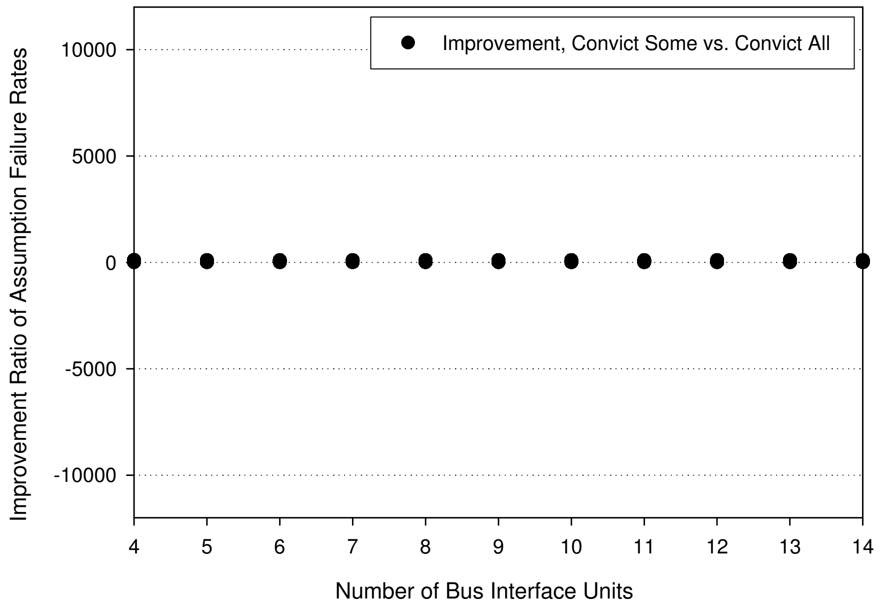


Figure 52. SPIDER: Improvement, Convict Some vs. Convict All

the *improvement* is defined as (Upper bound Strategy Two)/(Upper bound Strategy One) if the upper bound for Strategy Two is the larger number, and (- Upper bound Strategy One)/(Upper bound Strategy Two) if the upper bound for Strategy One is the larger number. Therefore, the improvement is negative if Strategy Two has a lower assumption failure rate. Improvement is a unit-less quantity since it is the ratio of two assumption failure rate measurements. Figures 52, 53, and 54 include all configurations (there were no improvement values greater than 12,000 or less than -12,000). For all of the SPIDER configurations, all improvement measurements were greater than the error.

The Convict Some strategy had a lower assumption failure rate than the Convict All strategy when all pairs of configurations from both strategies were compared. Figure 52 illustrates the improvement of the Convict Some strategy over the Convict All strategy. Since the Convict All strategy is the standard, this shows that a smart fault diagnosis strategy can improve assumption reliability. The maximum amount of improvement was about one order of magnitude (10^1).

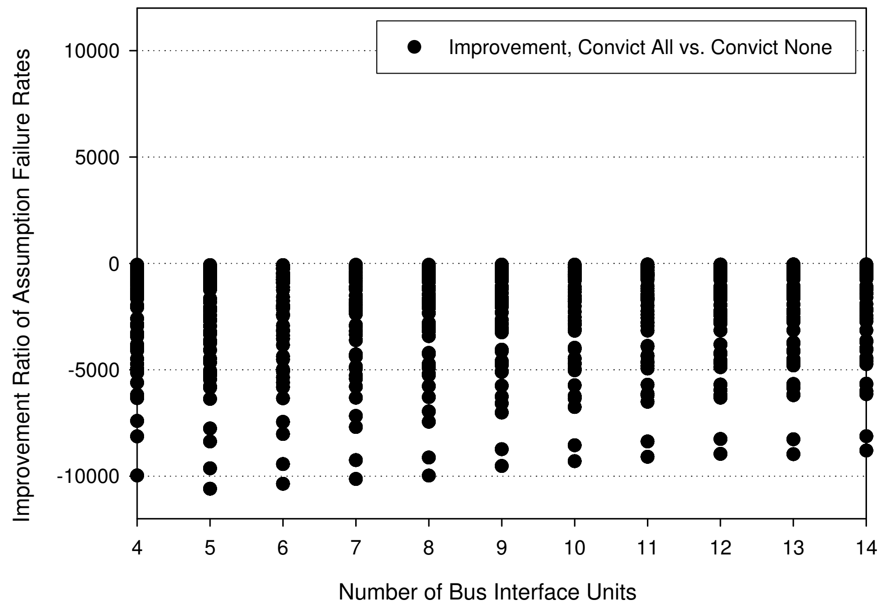


Figure 53. SPIDER: Improvement, Convict All vs. Convict None

The standard Convict All strategy had a higher assumption failure rate than the Convict None strategy for all pairs of configurations. Figure 53 shows how the assumption failure rates of Convict All configurations compare to the assumption failure rates of Convict None configurations. It appears that the Convict All strategy is too aggressive, and the assumptions fail due to too many inappropriate convictions of transient faulty nodes. Further investigation of the data confirms this. Figure 53 shows that some Convict All strategy configurations are only a little worse than in the Convict None strategy, and that some are quite a bit worse. The Convict All configurations with the worst difference had the highest transient fault arrival rates - a Single Event Upset rate of 10^{-8} and a Bit Error Rate of 10^{-12}.

The comparison between the Convict Some strategy and the Convict None strategy is interesting. For almost all pairs of configurations, the Convict Some strategy had a higher assumption failure rate than the Convict None strategy, shown in Figure 54. In these cases it was better to do nothing than to try to convict faulty nodes, due to the chance of misclassification. There were a few configurations where the Convict Some strategy was better

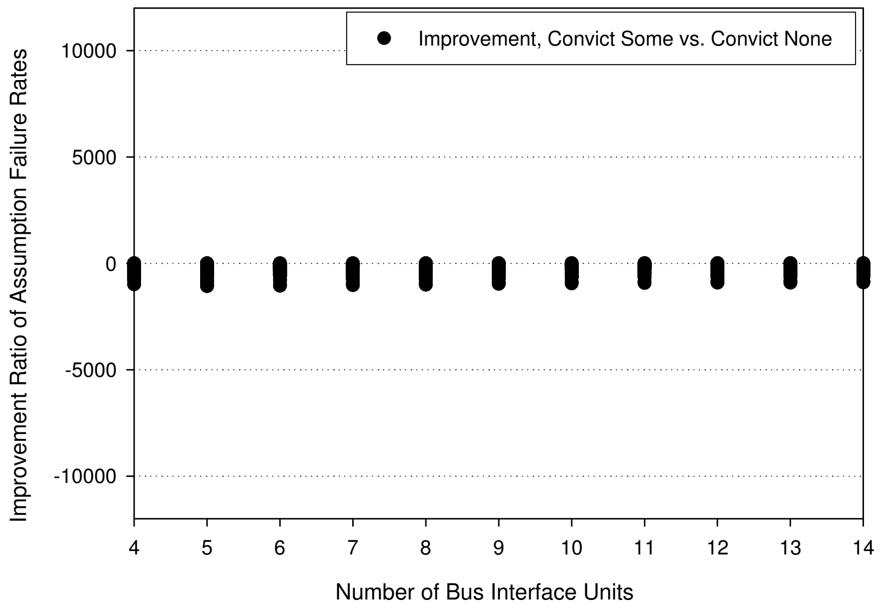


Figure 54. SPIDER: Improvement, Convict Some vs. Convict None

Table 33. SPIDER Membership Dominant Failure
Lists the number of configurations according to which condition of the maximum fault assumption fails first

	Conv. All	Conv. None	Conv. Some
Not Enough RMUs (MFA.1)	All	420 (70.7%)	All
Not Enough BIUs (MFA.2)	0	8 (1.3%)	0
Asymmetric RMU plus Asymmetric BIU (MFA.3)	0	166 (28.0%)	0

than the Convict None strategy. These configurations had the lowest transient fault arrival rates tested - an SEU rate of 10^{-10} and a BER of 10^{-11}. This indicates that inappropriate convictions can make the system vulnerable as the transient fault rates increase.

4.6.6 Death State Analysis

In all configurations studied, the Convict All and Convict Some strategy failed by running out of Redundancy Management Units (RMUs), as shown in Table 33. The Convict None strategy failed due to running out of RMUs in most cases. In some configurations with high

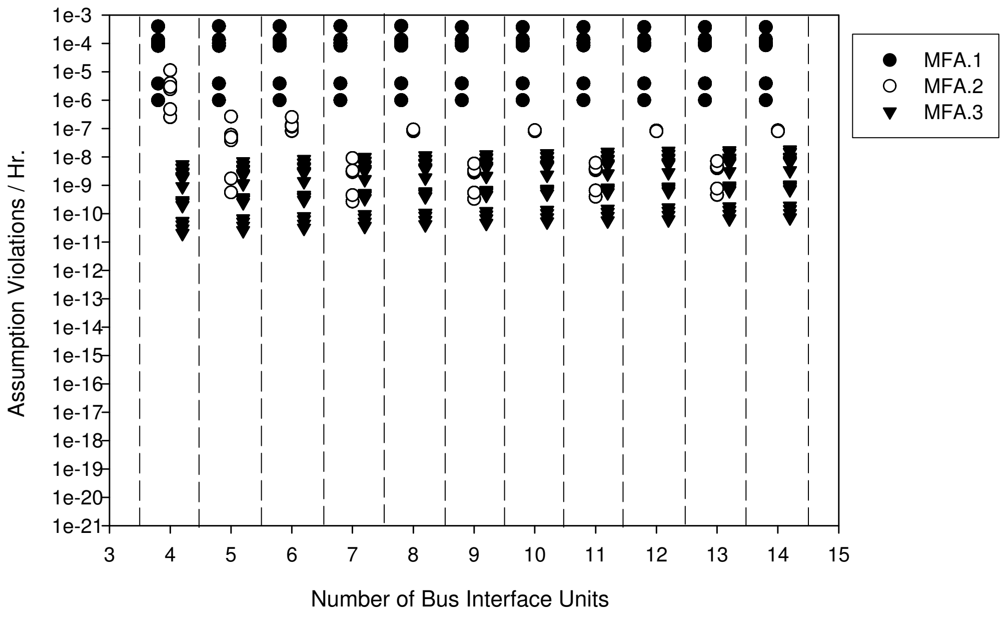


Figure 55. SPIDER Convict All: Dominant Assumption Failure Scatter Plot

fault rates, the Convict None strategy failed due to a critical pair of asymmetric faults (an asymmetric faulty RMU plus an asymmetric faulty Bus Interface Unit). In a handful of the four BIU configurations, the Convict None strategy failed due to running out of BIUs.

Usually, one piece of the maximum fault assumption is the primary cause of failure. A successful strategy will mitigate faults that violate this piece of the maximum fault assumption. The SURE tools allow each piece of the maximum fault assumption to be tested separately. Table 30 lists the conditions tested. The conditions are tested in order, so states that fail to satisfy MFA.1 will not be checked for MFA.2 and MFA.3, for example.

Figures 55, 56, and 57 portray the assumption failure rate due to each maximum fault assumption condition for the three strategies. The Number of Bus Interface Units is listed along the X axis, although for SPIDER changing the number of BIUs does not affect the assumption failure rate much. The Y axis shows the assumption failure rate in assumption violations/hr.

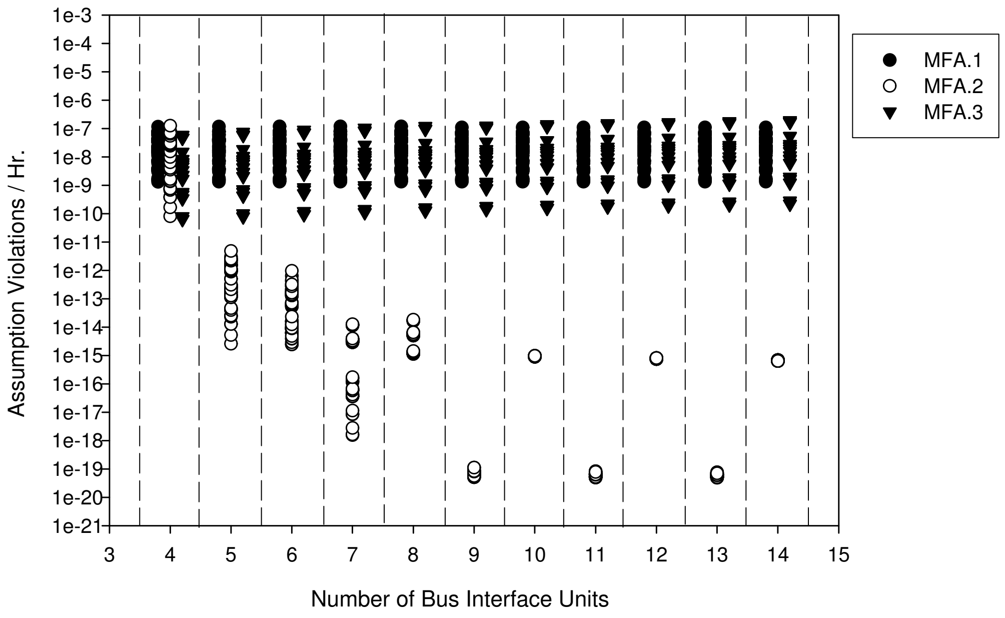


Figure 56. SPIDER Convict None: Dominant Assumption Failure Scatter Plot

For the Convict All and Convict Some strategies, MFA.1 was by far the most likely piece of the maximum fault assumption to fail. A violation of MFA.1 occurs when there are not enough Redundancy Management Units. For the Convict None strategy, MFA.1 and MFA.3 are about equal causes of failure. MFA.3 is violated when there is (at least) one asymmetric Bus Interface Unit plus (at least) one asymmetric Redundancy Management Unit. In other words, if MFA.3 is violated, the system is compromised due to too many active faults.

The number of Bus Interface Units looks adequate for all three strategies, since the failure rate due to MFA.2 (running out of BIUs) is lower than MFA.1 and is also lower than MFA.3 for the Convict None and Convict Some strategies. (The four BIU configurations are an exception, where MFA.2 has a similar failure rate in some cases). It seems that if more RMUs are added to the system, the Convict Some strategy might improve.

Convicting nodes might have some benefit with respect to MFA.3, where there is at least

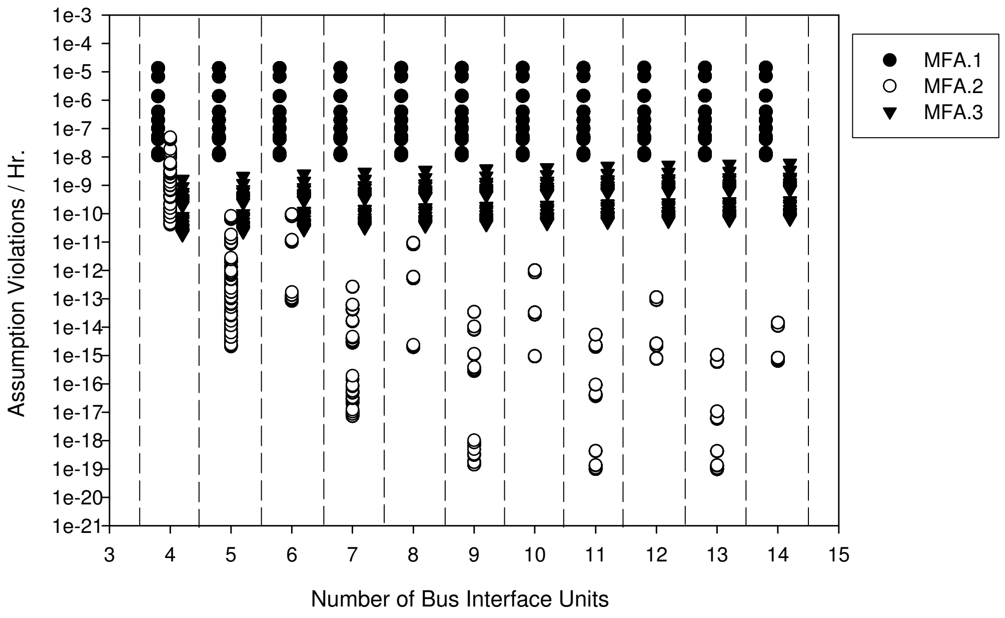


Figure 57. SPIDER Convict Some: Dominant Assumption Failure Scatter Plot

one asymmetric faulty BIU and at least one asymmetric faulty RMU. For the Convict All (Figure 55) and Convict Some (Figure 57) strategies, the failure rate of MFA.3 is in the range of about 10^{-8} to 10^{-11}. In the Convict None strategy (Figure 56), the failure rate of MFA.3 is in the range of about 10^{-7} to 10^{-10}. However, since MFA.3 is checked last, this could also be due to configurations having a failed MFA.1 first.

4.6.7 Sensitivity Analysis

For SPIDER, the assumption failure rate for the Convict Some strategy was heavily dependent on the probability of convicting permanent faulty nodes. Previous work I did with an earlier SPIDER model suggested that the Convict Some strategy performed much better if there was perfect permanent fault conviction [67]. The sensitivity analysis investigates the current SPIDER model, with perfect permanent fault conviction. There were 1782 configurations total: (11 BIU) * (3 SEL) * (3 SEU) * (3 PermLink) * (2 BER) * (3 Probabilities of

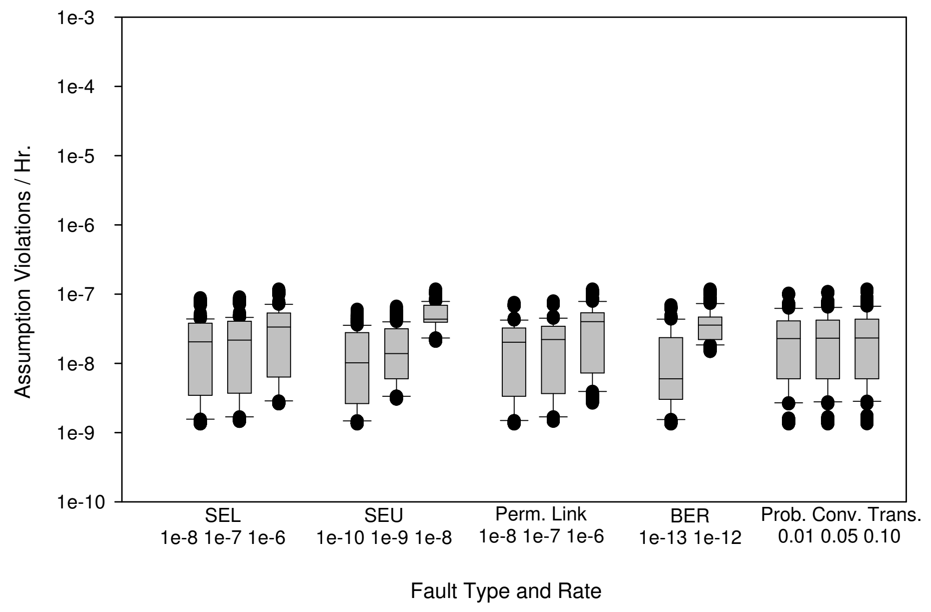


Figure 58. SPIDER Sensitivity Analysis - Perfect Permanent Fault Conviction, Box Plot

convicting a transient faulty node). The probability of convicting a permanent faulty node was set to 1.

With perfect conviction, configurations from the SPIDER Convict Some strategy have a much lower assumption failure rate. Figure 58 illustrates the box plots for the SPIDER Convict Some strategy with perfect conviction. The assumption failure rate ranges from about 10^{-9} to 10^{-7}, which is much lower than the range for imperfect conviction (about 10^{-5} to 10^{-8}). The new assumption failure rate range is close to the assumption failure rate range of the Convict None strategy, in Figure 43 in the Physical Fault Sensitivity section. This shows that the penalty for misclassification is high for SPIDER.

4.7 Protocol Comparison

While different protocols are not directly comparable, it is helpful to summarize the distinctions between them. Table 34 summarizes some notable features and design choices

Table 34. Protocol Feature Comparison

Features	FlexRay	TTP/C	SPIDER
Clock synchronization	Yes	Yes	Yes
Agreement on group members	No	Yes	Yes
Agreement on data values	No	No	Yes
Tolerates truly Byzantine faulty star coupler	N/A	No	Yes
Good nodes never convicted	N/A	No	Yes
Channels	2	2	3

for each of the protocols. In this table, 'No' means this service is not offered at the protocol level, although optional application level services might be possible. 'N/A' means that this concern is not applicable based on the guarantees the service offers (i.e. clock synchronization or membership).

All three protocols provide distributed clock synchronization. Clock synchronization is a form of approximate agreement. The TTP/C and SPIDER protocols use formally proven clock synchronization algorithms. FlexRay clock synchronization is based on the formally proven Welch and Lynch clock synchronization algorithm, although the final FlexRay algorithm differs somewhat from the Welch and Lynch proof. The SPIDER clock synchronization algorithm is described and proved by Torres-Pomales, Malekpour, and Miner [117]. The TTP/C clock synchronization algorithm is described in the specification [118] and is formally proven by Pfeifer, Schwier and von Henke in [89]. The FlexRay clock synchronization algorithm is described in the FlexRay specification [33], and the Welch and Lynch algorithm is described in [120].

Guaranteeing agreement on the actual data values is worthwhile, since even if there is consensus on the group members, these group members may still act on different information. Both TTP/C and SPIDER guarantee agreement on the group members, but SPIDER additionally guarantees agreement on the data values transmitted. For example, if a sender sends a correctly formatted $frame_a$ to some receivers and a correctly formatted $frame_b$ to other receivers, there will be consensus on the group members even though the group members received different data values. This scenario could occur with a single fault. In TTP/C, for example, each node has two bus drivers (one for each channel) [118]. In TTP/C, since any correctly formatted frame may be accepted from either channel, this scenario could occur if the sender's bus driver on channel A sends a different (but correctly formatted) frame than the sender's bus driver on channel B.

Guaranteeing agreement on data values in the presence of Byzantine faults requires three channels, however, so that the receiver can perform a majority vote of the values received. Therefore, guaranteeing agreement on data values comes with a cost. The SPIDER topology elevates the star couplers to first-class status, called Redundancy Management Units (RMUs) in this protocol. Three RMUs are needed to guarantee consensus on the received values. (The $n > 3f + 1$ restriction applies to both the Bus Interface Units and Redundancy Management Units combined - the RMUs are the second stage in the two-round SPIDER diagnosis protocol.)

There is a related issue regarding what type of fault behavior the star coupler can exhibit if it is not treated as a first-class entity. It is tempting to add more complexity into the star coupler to reduce other failure modes. For example, recent work has proposed a star coupler that can alter the timing of frames, to reduce the probability of a slightly-off-specification timing fault from the sender [78]. Reintegration is also easier if the star coupler has certain abilities, such as the ability to provide the current membership vector to reintegrating nodes [78]. However, Morris and Koopman demonstrated that some restrictions must be placed on a star coupler's authority. For example, a star coupler cannot be permitted to store a complete frame, because it could then it could succumb to a store-and-forward fault [78]. This issue does not apply in the same manner to clock synchronization, since frames are valid if they arrive within the time slot. A related issue might be slightly fast or slightly slow clocks that eventually change to the rate so much that the physical clock cannot compute at that rate.

Since the SPIDER star couplers (RMUs) have first-class status in the membership proofs, SPIDER can tolerate truly Byzantine faulty RMUs. One advantage for diagnosis is that the SPIDER RMUs can pass on diagnostic information [117]. The RMUs are permitted to alter the frame. The cost is that the fault rates for the RMUs must be kept low. Since there are few RMUs and it is expensive to add RMUs due to wiring costs, high RMU fault rates may cause the system to run out of redundancy.

For good node conviction, the majority clique in TTP/C is not necessarily fault-free. An asymmetric faulty sender will belong the majority clique if most of the receivers obtain a correctly formatted frame. A node with a weak bus driver or faulty clock might cause this kind of behavior (redundant clocks and bus drivers with separate power sources can help address this problem). A malicious faulty node might take advantage of this behavior, but malicious faulty nodes are not considered here.

Finally, adding a channel is much more expensive than adding an individual node due to the cost of the wiring harness. This is especially true for automotive applications. In addition to the cost of the wiring itself, adding an extra wiring harness can affect other components in ways that require more expensive components. Extra wiring adds weight and can increase the electromagnetic interference (which may create a need for more shielding, which may add more weight). Weight is an important consideration for both automotive and aviation. In automotive systems, other parts will need to be reinforced to carry the weight, and this may impact emissions ratings. In aircraft, extra weight represents an increased fuel cost per flight, which can dramatically increase operating costs for a large number of flights. Installing wiring harnesses is difficult, and a common cause of problems in vehicles is incorrectly connected harnesses, especially if there are many connections involved. However, good wiring harness design can reduce this risk somewhat (for example, using different shapes for connectors so mismatch is physically impossible). Connectors are a typical point of failure, since they may come loose or crack over time.

4.8 Usage Guidelines and Alternative Calculations

As with any methodology, there is potential for misuse. It is helpful to summarize limitations of the methodology, and ways to address those limitations. First, this section summarizes some usage guidelines for the methodology.

Simpler mathematical calculations can be used as a sanity check for the more complex Markov model mathematics. The Physical Fault Model section gives an example of this, by calculating the assumption failure rate for SPIDER in the case where just fail-silent permanent hardware faults are present. The assumption failure rate due to lightning strikes for the TTP/C Convict All strategy is another example where a simpler calculation has been used to explain the results of a more complicated model. Two more examples are given here: calculating the probability of both channels failing, and calculating the probability of receiving two invalid frames within two rounds.

Technique best at relative comparisons. The methodology is best for making relative comparisons, instead of generating a single reliability number. Since there can be orders of magnitude uncertainty in the input fault rates, the modeling tools will not be able to generate a high-precision reliability number as output. However, the technique is very useful for comparing two different design choices to see which one is better.

Be careful when comparing two protocols. Assumption reliability measurements from two different protocols are not directly comparable, since the guarantees might be different. The methodology cannot tell the designer which guarantees are important; it can only evaluate the ability of a protocol to provide the service it claims to provide. For example, which is better, guaranteed agreement on group members with a 10^{-5} probability of failure per hour or guaranteed agreement on data values with a 10^{-4} probability of failure per hour? This depends on the application.

Results depend on the fault model used. As with any methodology, the output depends on the input. If the fault profile input is not representative of the faults encountered during operation, then the actual assumption reliability will be different. One of the advantages of this methodology is that a wide range of fault rates can be tested. However, the number of reliability models increases each time a new rate parameter is tested in a full factorial experiment setup. A designer might need to select a subset. The amount of work can be mitigated some by sensitivity analysis, examining selected values of fault arrival rates and system parameters before choosing a subset to use for a full factorial experiment.

Other important considerations exist. There are other important considerations in the development of X-by-Wire systems, such as data availability and faults outside the scope of the protocol. A group membership service typically treats all nodes as equal members. In an embedded system, however, the processing element associated with each network node will have a distinct function in the system. By convicting nodes, a membership service may compromise data availability. For example, a group membership service might convict and remove all four braking nodes (one braking node at each wheel). The maximum fault assumption may not have been violated, but the vehicle will not be able to brake. Also, there will always be faults outside of the ability of the protocol to handle. If the applications using the protocol are unreliable, then the system will be unreliable.

Methodology false positives. If the methodology incorrectly identifies an area of concern, one possible outcome is that a lot of time will be spent on a portion of the design that does not matter. However, if no assumption reliability estimation is done, the chance of spending time on unimportant design features is much more likely. In fact, the comparisons between the standard Convict All membership strategies and the Convict None membership strategies show that investing more effort into a design does not ensure a better design. The standard Convict All membership strategy are the result of numerous proofs and much effort targeted at reducing the risk posed by Byzantine faulty nodes. Unfortunately, for many configurations the Convict All strategy had a higher assumption failure rate than doing nothing (Convict None), due to inappropriate convictions. Good process is correlated with, but does not ensure, good product. This is where a measurement methodology can have great benefit.

Methodology false negatives. The methodology might also miss something that would decrease the assumption reliability, giving an overly optimistic estimate. However, the baseline case of no evaluation provides no guidance at all. Since this methodology is a just a piece of the system development lifecycle, implementation testing will still be done. The methodology could be used to focus the implementation testing. The risk is that randomly targeted implementation testing might produce a more reliable system than targeted implementation testing, when some of the testing is targeted at a fault that does not matter.

4.8.1 Channel Failures

One simple bound on the assumption reliability of the service is the probability of all channels permanently failing. If all channels are permanently failed, then no frames will be transmitted. The service will surely fail in this case, and will not be able to recover until at least one of the channels has been repaired.

Assuming independent faults (the most optimistic model), then the service failure rate due to channel failures is easy to calculate. The physical fault rate for a bus or star coupler was cited as 10^{-6} faults/hr in the Methodology chapter. The clock synchronization algorithms and the TTP/C membership algorithms both use a dual-channel design. The service failure rate due to both channels becoming permanently faulty is 10^{-6}/hr * 10^{-6}/hr = 10^{-12}/hr. A typical SPIDER design uses three Redundancy Management Units (similar to star couplers). The probability of all three RMUs becoming permanently faulty is 10^{-6}/hr * 10^{-6}/hr * 10^{-6}/hr = 10^{-18}/hr. This calculation assumes that the RMUs fail in such a way that the SPIDER service is able to diagnose at least two of them as benign faulty. In other words, the service might fail for some combinations of just two faulty RMUs, if the RMUs fail at about the same time.

4.8.2 Bit Error Rate Calculations

While the example above could be extended to other types of faults that affect the nodes, the bit error rate category is a little more difficult to compute. Here, the dual-channel TTP/C frame correctness rules are used as an example. For a receiver to notice a faulty frame, an invalid frame must be received on both channels. Then, to violate the TTP/C maximum fault assumption, two invalid frames must be received within two TDMA rounds. (The reliability models use a slightly more conservative maximum fault assumption, since the reliability modeling tools do not explicitly include the concept of a round).

This section presents equations for computing the probability that two frames will be lost on both channels within two rounds. The probability calculations were validated through simulation. The probability of having two faults occur in the same message will depend on the frame size. I assume that each node transmits one frame per round, completing an entire round in 5 ms. The system has two channels, with an independent single bit corruption fault model. The probability of a bit being corrupted is given as a Bit Error Rate, defined as Errors/Bits. I assume that all frames are the same size. The calculations use the following parameters:

N: Number of nodes

BER: Bit Error Rate in errors/bits

PERM: Permanent fault arrival rate in faults/round

Framesize: Number of bits in a frame

Roundtime: Number of seconds in a round

In order for a frame in TTP/C to be considered invalid, the same frame must be invalid on both channels [118], pg. 59. It is possible that a single node transmits its frame at different times on the two channels, as a form of temporal redundancy. Even if the frame is sent at different times on each channel, the calculations apply as long as a node sends only one frame per channel per round, as the fault model is composed of uncorrelated single bit errors. The calculations first compute the probability of having a pair of frames, where each frame has one or more faulty bits.

The probability that a frame has one or more faulty bits equals 1 minus the probability of no faulty bits. Since there are Framesize bits in a frame, and each bit has a BER chance of being faulty, this equals:

$P\_invalid\_frame : 1 - (1 - BER)Framesize$

Since the two channels are independent, the chance of an invalid frame pair on both channels is:

$P\_invalid\_frame\_pair : (1 - (1 - BER)Framesize)^2$

Consider two consecutive rounds, Round A and Round B. There are two ways to violate the Single Fault Hypothesis; by having two or more pairs of invalid frames in Round A (CaseAA), or one invalid frame pair in Round A and one or more pairs of invalid frames in Round B (CaseAB). If three (or more) consecutive rounds contain pairs of invalid frames, this is counted as a CaseAB, so the round preceding a CaseAA must contain no invalid frame pairs. The first step is to calculate the probability of having no invalid frame pairs in a round and the probability of having exactly one invalid frame pair in a round. Since each node transmits once per round, there are N independent frame pairs, so the probability of zero invalid frame pairs in a round is:

$P\_zero\_invalid\_frame\_pairs\_in\_round : (1 - P\_invalid\_frame\_pair)^N$

If one frame pair is invalid, and the rest are valid, there are N choose 1 places in the round that the invalid frame may be. The probability of exactly one invalid frame in a round is:

$P\_exactly\_one\_invalid\_frame\_pair\_in\_round$:

$(N!/(1! * (N - 1)!)) * (P\_invalid\_frame\_pair) * (1 - P\_invalid\_frame\_pair)(N - 1)$

The probability of a CaseAA occurring equals the chance of having a round with zero invalid frame pairs followed by a round with two or more invalid frame pairs.

$P\_CaseAA : (P\_zero) * (1 - (P\_zero + P\_exactly\_one))$

The probability of a CaseAB occurring equals the chance of two consecutive rounds with one or more invalid frame pairs. (If the first round has two or more invalid frame pairs and qualifies as a CaseAA, this situation is counted as two Single Fault Hypothesis violations, since a round with two invalid frame pairs might instead be a CaseAB if the prior round had an invalid frame pair.)

$P\_CaseAB : (1 - P\_zero) * (1 - P\_zero)$

To get the number of Single Fault Hypothesis violations expected per hour, we multiply by the number of rounds in an hour:

$Expected\_Violations : (P\_CaseAA + P\_CaseAB) * (3600 seconds/Roundtime)$

Table 35 lists the expected number of Single Fault Hypothesis violations per hour for some of the combinations of number of nodes, frame sizes, and BERs studied. Since the bandwidth was kept at a constant 1 MBit/second and divided equally among all nodes, the frame size is shorter for configurations with more nodes. These calculations are for transient faults with a single bit duration; longer faults could possibly corrupt multiple frames. These transient fault calculations were validated using a fault injection simulation.

Compared to the number of assumption violations expected for permanent faults, the impact of transient faults can be significant. Two (or more) permanent faults must occur within two rounds to violate the Single Fault Hypothesis. Nodes that have already failed will have been removed from the group, and are no longer a detriment to reliability after consensus on the group has been reached. The expected number of Single Fault Hypothesis violations due to permanent faults can be computed in a similar manner. The probability of a node failing per round is PERM. The probability of no permanent faults in a round is:

$P\_zero\_perm : (1 - PERM)^N$

For exactly one permanent fault, there are N choose 1 nodes that may be faulty:

$P\_exactly\_one\_perm : (N!/(1! * (N - 1)!)) * (1 - PERM)(N - 1) * (PERM)$

The calculations only include permanent node faults, and do not include permanent channel faults. CaseAA_*Perm*, where there are two permanent faults in one round and zero in

Table 35. Probability of Two Invalid Frames Within Two Rounds, by BER

	4 Nodes	10 Nodes	20 Nodes
BER	**1250 Bit Frame**	**500 Bit Frame**	**250 Bit Frame**
10^{-10}	$3.9*10^{-21}$	$6.6*10^{-22}$	$1.8*10^{-22}$
10^{-9}	$3.9*10^{-17}$	$6.5*10^{-18}$	$1.7*10^{-18}$
10^{-8}	$3.9*10^{-13}$	$6.5*10^{-14}$	$1.7*10^{-14}$
10^{-7}	$3.9*10^{-9}$	$6.5*10^{-10}$	$1.7*10^{-10}$
10^{-6}	$3.9*10^{-5}$	$6.5*10^{-6}$	$1.7*10^{-6}$
10^{-5}	$3.8*10^{-1}$	$6.5*10^{-2}$	$1.7*10^{-2}$

the prior round, is:

$P\_CaseAA\_Perm : (P\_zero\_perm) * (1 - (P\_zero\_perm + P\_exactly\_one\_perm))$

CaseAB_*Perm*, where there is at least one permanent fault in Round A and in Round B, is:

$P\_CaseAB\_Perm : (1 - P\_zero) * (1 - P\_zero)$

The expected number of Single Fault Hypothesis Violations per hour due to permanent faults is:

$Expected\_Violations\_Perm$:

$(P\_CaseAA\_Perm + P_{C}aseAB\_Perm) * (3600 seconds/Roundtime)$

For a permanent fault arrival rate of 10^{-5} faults/hour, we would expect the number of Single Fault Hypothesis assumption violations per hour to be approximately $3.1 * 10^{-15}$ for 4 nodes, $2.0 * 10^{-14}$ for 10 nodes, and $8.2 * 10^{-14}$ for 20 nodes. From Table 35, this number of assumption violations per hour corresponds to a bit error rate of between 10^{-9} and 10^{-8} errors/bits. Transient faults are worthy of study, especially for domains with higher bit error rates such as the automotive domain.

4.9 Future Work

Future work could use the information and methodology presented to test new kinds of fault tolerance strategies and other types of faults. First, this dissertation paves the way for adaptive fault tolerance strategies. Now that designers can identify trade-off points, the fault tolerance strategies can be customized to these trade-off points. Second, while one category of burst faults is covered in the dissertation (lightning), there are other burst fault sources that could be investigated. Finally, there are a few additional experiments that this methodology could be used to execute.

4.9.1 Adaptive Group Membership

One exciting category of future research is the development of adaptive fault tolerance algorithms. This dissertation demonstrated how to find trade-off points in the design space, and demonstrated that trade-off points do exist for most of the protocols studied. For example, in TTP/C, sometimes adding nodes improved the assumption reliability and sometimes it did not, depending on what the dominant cause of assumption failure was. The same was true with respect to different fault types. An adaptive fault tolerance algorithm could be customized to these trade-off points, providing superior service when compared to a static fault tolerance algorithm.

Group membership services provide a great platform for building dynamically adaptive fault tolerance strategies (assuming the proofs are constructed to allow adaptive strategies, such as in SPIDER). Since TTP/C and SPIDER guarantee agreement on which nodes belong to the active group, one could tailor the fault tolerance strategy to the current number of group members (all good nodes in the group are guaranteed to agree on this number). If agreement is guaranteed on the data values as well, the fault tolerance strategy could adapt to other parameters, such as the measured fault rate for a certain fault type instead of the average fault rate. Proactive strategies could even be implemented. For example, if an airplane is flying into a storm and navigation data is broadcast on the network, the fault tolerance strategy could be changed ahead of time to better tolerate lightning strikes. The set of possible fault tolerance strategies would still to be programmed into the nodes at design time.

4.9.2 Burst Faults and Modeling

Electromagnetic interference includes a wide range of possible burst faults. Also, X-by-Wire systems will involve the development of new electromechanical actuators (such as local braking actuators at each wheel). These new components could be new sources of electromagnetic interference. Thus, it is important for a service to be able to handle these burst faults.

The relationship between the burst fault duration and round duration could be interesting to investigate. The designer has some control over what the round duration should be. Some of my early preliminary work with the TTP/C protocol suggests that there is a sharp decrease in assumption reliability soon after burst faults exceed the round duration. The decrease is less steep for single round duration burst faults compared to multiple round duration burst faults. However, these observations were from very early TTP/C models, which did not include all the features of the protocol and did not include a comprehensive fault model test. The electromagnetic compatibility standards cited in the Related Work chapter would provide a good source of data for a researcher interested in this topic.

Another research area could be to determine an appropriate metric for measuring system response to burst faults. The assumption reliability metric is limited, since if a burst fault immediately violates the maximum fault assumption, the assumption reliability will be bound by that fault rate. A better measure might reflect the ability of the service to recover quickly after a fault. Theoretically, as long as the clocks stay synchronized, the clock synchronization or membership service might be able to resume operation immediately after the burst fault is over. This would require the service to correctly identify and ignore a burst fault. Temporal redundancy techniques that have been proposed might also help with respect to burst faults. For example, a node in TTP/C is permitted to send its frame in different time slots on different channels. A burst fault might only affect one of the replicated frames.

Finally, a system designer might want to know what burst fault rate is acceptable. It is difficult to find data on actual burst fault rates, since the electromagnetic compatibility standards are optimized for minimum testing time. So, the burst fault rate will be the fastest rate that the hardware component can handle, in order to complete the testing as quickly as possible. Modeling these rates will not accurately reflect real-world fault rates. Therefore a designer might take the reverse approach, and try to determine the highest burst fault rate that the service can handle.

4.9.3 Future Experiments

Any set of results always brings out new questions. There are a few direct extensions of this methodology that seem to be good avenues for research.

It would be interesting to investigate the effects of adding more Redundancy Management Units to SPIDER. While the cost would probably be too high for automotive, a four RMU SPIDER might be feasible for aviation. Preliminary investigations done with earlier SPIDER models suggest that adding more RMUs does increase assumption reliability quite a bit. However, these investigations used a simpler form of the physical fault model and did not model misclassified faults.

Also, it would be an interesting follow-on study to run a complete set of models for automotive fault arrival rates. This research has identified a set of fault arrival rates for aviation as well as automotive. There is some difficulty creating and solving models for automotive systems, since the transient fault arrival rates are so high. However, the models can be made smaller if some of the physical fault types are omitted (this leads to fewer transitions and possibly a smaller state space). The aviation results can guide the choice of which faults to omit from the analysis, since the sensitivity to each of the fault types is known for algorithms tested with the aviation data. Fault types that the models were insensitive to could be omitted from the automotive studies. For example, almost every protocol/diagnosis strategy was insensitive to the Single Event Latchup rate. Since the Single Event Latchup rate is even lower for automotive systems, it is unlikely that SEL faults would have any effect on the assumption reliability for automotive systems. This fault type could be removed from the analysis.

5 Conclusions

Ultra-reliable systems are difficult to design and even more difficult to test. New X-by-Wire network protocols are being developed for ultra-reliable aviation and automotive applications. The goal of an X-by-Wire protocol is to electronically provide safety-critical functionality (such as steering or braking) with no mechanical backup. The acceptable failure rate of these functions is on the order of 10^{-9} failures per hour, according to guidelines from the Federal Aviation Administration and the Motor Industry Software Reliability Association. Since exhaustive testing is infeasible at this level of reliability, designers rely on carefully constructed specifications, where some portion of the specification often involves a formal proof of correctness. The specifications for these protocols guarantee that certain properties hold as long as a given maximum number of faults is not exceeded (the maximum fault assumption). If the maximum fault assumption is violated, the guarantees might not be provided, and the system may fail.

I introduce a methodology to estimate the reliability of a service, at design time, by measuring the probability that the service's maximum fault assumption will be violated for a realistic physical fault model. The key idea is that the probability of violating the maximum fault assumption can be used as a (conservative) estimate of the service's failure rate. I provide a realistic reusable physical fault model, reliability modeling templates for the three protocols, and describe the measurement process. My methodology can accommodate uncertainty in fault arrival rates, since order of magnitude approximations are common at the design stage. This methodology is built around Markov modeling tools from NASA Langley Research Center, although the concepts are not limited to this particular tool suite. My research focuses on the distributed clock synchronization and distributed group membership services for three next-generation X-by-Wire protocols: FlexRay, the Time Triggered Protocol Class C (TTP/C), and the NASA Langley Scalable Processor Independent Design for Electromagnetic Resilience (SPIDER). Two of these protocols are slated for use in production aviation and automotive systems.

Next, I apply my methodology to these three safety-critical X-by-Wire protocols, producing a number of interesting conclusions. First, I show that it is important to test systems with a realistic fault model that includes transient faults. In a study of the SPIDER protocol, a simple fail-silent fault model overestimated assumption reliability by over ten orders of magnitude. Next, I show that measurement is crucial to creating a more reliable service. Investing more effort into a design does not ensure a more reliable service. To emphasize this point, I show that the standard fault removal strategy for both TTP/C and SPIDER is, overall, worse than no fault removal strategy. With assumption reliability testing, the fault tolerance strategy can be improved. Finally, since improvement is possible, the next question is which enhancement is expected to provide the most benefit. My research demonstrates how to identify trade-off points in the design space. Two common enhancements are adding more redundant components or installing higher quality components. Will either action help? I show that the estimated reliability depends on the dominant cause of assumption violation, which may change throughout the design space. Successful enhancements address this dominant cause.

5.1 Methodology Contributions

- **Reliability modeling process**

  The reliability modeling process starts with the key idea that the reliability of a service can be estimated by testing the probability that the service's maximum fault assumption holds for the duration of the mission. Steps in the process include identifying design choices, defining a physical fault model, creating a Markov model template from the protocol specification and its maximum fault assumption, then testing and comparing the design choices with the physical fault model. This process uses Markov modeling tools from NASA Langley Research Center, although the methodology is not limited to this tool suite.

- **Realistic, reusable physical fault model**

I define a realistic, reusable physical fault model for both the aviation and automotive domains. I then test a wide variety of configurations using the aviation fault profile. The physical fault model is reusable for testing other services in the same domain. By testing a large number of combinations of fault rates, the methodology can produce meaningful results despite uncertainty in the input parameters.

- **Reliability modeling templates for the three protocols**

  A reliability modeling template describes the states, transitions, and death conditions for each study. To create the template, a designer must map the physical faults onto the fault terms used in the service's maximum fault assumption. I give full mappings and transition tables for each of the three protocols. A template can serve as a baseline for others, and new types of faults (and the corresponding transitions) could be added to these templates.

5.2 Observations

- **Transient faults are important**

  The Physical Fault Model section illustrates how a simple permanent physical fault model can underestimate the assumption failure rate compared to a more extensive physical fault model that included transient faults. The permanent fault model with fail-silent only faults underestimated the assumption failure rate by over ten orders of magnitude compared to the extensive physical fault model, in tests of the SPIDER membership service. The permanent fault model with two additional types of permanent faults still underestimated the assumption failure rate by about five orders of magnitude compared to the extensive physical fault model.

  The Physical Fault Sensitivity section, for each of the three protocols, shows that the transient faults studied had a larger effect on the assumption failure rate estimate than the permanent faults studied. The sensitivity to a particular fault is depicted using a set of box plots, with one box plot for a set of data with a given fault rate and type. If the box plot changes as the fault rate changes, then the assumption failure rate estimates are sensitive to this fault. The Clock Synchronization services were both most sensitive to the (transient) Bit Error Rate. The TTP/C and SPIDER membership services were both most sensitive to the (transient) Single Event Upset rate. In contrast, the models were less sensitive to the Permanent Link fault rate, and only the SPIDER Convict None strategy showed sensitivity to the (permanent) Single Event Latchup rate.

- **Unmeasured designs can lead to a less reliable service**

  In my dissertation, I investigate two types of design decisions. For the FlexRay protocol, I examine the clock synchronization service, based on the Welch and Lynch clock synchronization algorithm. A new algorithm, the Strictly Omissive Asymmetric algorithm, proposes a new maximum fault assumption in terms of a revised hybrid fault model. I compare the Welch and Lynch maximum fault assumption against the Strictly Omissive maximum fault assumption. For the TTP/C and SPIDER, I examine the fault removal strategies for the membership algorithms. The standard fault removal strategy, Convict All, removes nodes after a single fault. I compare this strategy to never removing any nodes (Convict None) and removing permanent faulty nodes while letting transient faults expire (Convict Some).

  For all three protocols, I discuss both overall and per-configuration comparisons. To get a view of the overall assumption failure rate of a design decision, I construct a histogram for each of the three protocols. Here, 'better' means that the strategy had more configurations with lower assumption failure rates. For clock synchronization, overall, the Strictly Omissive Asymmetric algorithm was better than the Welch and Lynch algorithm. For membership, the conclusions were surprising. Overall, for both TTP/C and SPIDER the do-nothing Convict None strategy was better than the standard Convict All strategy. For TTP/C, the Convict Some strategy was better overall than both the standard and do-nothing strategy. For SPIDER, however, the do-nothing strategy was better overall than the Convict Some strategy. I show that for SPIDER, this is due to misclassifications by the Convict Some strategy.

In addition to overall comparisons, I investigate the ramifications of changing the maximum fault assumption or fault tolerance strategy for each individual configuration. This is important, since a given strategy X might look better overall than strategy Y, but strategy X might have a higher assumption failure rate than strategy Y for certain configurations. I define an improvement metric for a pair of configurations. Each pair of configurations compared has the same set of fault arrival rates; the configurations only differ in their maximum fault assumption (for clock synchronization) or fault tolerance strategy (for membership). For clock synchronization, all of the Strictly Omissive Asymmetric configurations had a lower assumption failure rate than their Welch and Lynch algorithm counterparts. For TTP/C, the amount of improvement depends on the number of nodes in the system. For example, the Convict Some strategy did not show any improvement over the Convict None strategy until about ten nodes and up. For SPIDER, the amount of improvement was independent of the number of nodes, depending instead on the combination of fault rates and fault diagnosis accuracy for a particular configuration.

- **Trade-off points can be identified and used to choose enhancements**

  There are two types of common component enhancements: more redundant components can be added, or higher quality components can be used. The Death State Analysis sections, for each of the protocols, address the question of whether or not adding redundant nodes will help. Each protocol has a maximum fault assumption split into three pieces, which are translated into death conditions (sink states) in the reliability models. Typically, one piece of the maximum fault assumption is violated more often than the others. If the dominant assumption violation is too many asymmetric faults, adding nodes is not expected to improve reliability. This is especially true for the TTP/C and SPIDER membership services, since the maximum fault assumption allows only a fixed number of asymmetric faults at a time. Adding redundant nodes will only increase the chance of a fixed number of nodes being asymmetric faulty. If the dominant assumption violation is not having enough good nodes (for example, TTP/C assumes that there are at least three good nodes), then adding redundant nodes should improve the reliability. The TTP/C Convict Some strategy is a good example of trade-off analysis. Before about nine nodes, the assumption failure rate due to not enough nodes is high. This assumption failure rate decreases as nodes are added, until it drops below the assumption failure rate due to too many active faults (which stays relatively constant for TTP/C).

If higher quality components are added, the key choice is which fault type to mitigate. For each protocol, the assumption failure rates of configurations are graphed according to the two dominant types of faults. In most cases, there is a good fit. Higher quality components that lessen the arrival rates of the dominant faults should improve service reliability. In some cases (for example, the SPIDER Convict None strategy), the top two faults do not describe the variability in the data well. If this is so, higher quality components might or might not help, since it might be necessary to reduce the rates of all fault types to improve service reliability.

- **Foundation for adaptive group membership strategies**

  Because trade-off points can be identified, this dissertation paves the way for adaptive group membership strategies. A group membership strategy could adapt, at run time, using any information that all members agree upon. For example, in TTP/C, I identified a trade-off point at around nine nodes for the Convict Some strategy. Below about nine nodes, the consequences of incorrectly convicting a node are greater than the consequences of leaving in a faulty node (for the physical fault profile studied). Above about nine nodes, the situation is reversed. Since all good nodes in TTP/C agree on the size of the group, the group members could tailor their local fault removal scheme to the current group size.

5.3 Publications

Publications related to this dissertation:

- E. Latronico and P. Koopman. Design Time Reliability Analysis of Distributed Fault Tolerance Algorithms. *2005 International Conference on Dependable Systems and Networks (DSN '05)*, June 2005.
- E. Latronico, P. Miner, and P. Koopman. Quantifying the Reliability of Proven SPIDER Group Membership Service Guarantees. *2004 International Conference on Dependable Systems and Networks (DSN '04)*, June 2004.
- E. Latronico and P. Koopman. A Period-Based Group Membership Strategy for Nodes of TDMA Networks. *Fifth IFAC International Conference on Fieldbus Systems and Their Applications (FeT '03)*, July 2003.
- E. Latronico. Problems Facing Group Membership Specifications for X-by-Wire Protocols. *2003 International Conference on Dependable Systems and Networks (DSN '03)*, Student Paper, June 2003.

Additional publications:

- E. Latronico and P. Koopman. Representing Embedded System Sequence Diagrams As A Formal Language. *Fourth International Conference on the Unified Modeling Language (UML '01)*, October 2001.
- B. Latronico, C. Martin, and P. Koopman. Analyzing Dependability of Embedded Systems from the User Perspective. *Workshop on Reliability in Embedded Systems*

(in conjunction with Symposium on Reliable Distributed Systems/SRDS-2001), October 2001.

- M. Paulitsch, J. Morris, B. Hall, K. Driscoll, E. Latronico, and P. Koopman. Coverage and the Use of Cyclic Redundancy Codes in Ultra-Dependable Systems. *2005 International Conference on Dependable Systems and Networks (DSN '05)*, June 2005.

References

[1] A. Ademaj, H. Sivencrona, G. Bauer, and J. Torin. Evaluation of Fault-Handling of the Time-Triggered Architecture with Bus and Star Topology. *Proc. of the 2003 Intl. Conf. on Dependable Systems and Networks (DNS '03)*, June 2003.

[2] Allen-Bradley. Tech Talk. *Sensors Today*, June 1999, p. 12-13.

[3] Agilent Technologies. Measuring Jitter in Digital Systems, Application Note 1448-1. June 2003.

[4] austriamicrosystems AG. AS8202NF TTP-C2NF Communication Controller Data Sheet Rev.1.2, Nov. 2003.

[5] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, Vol. 1, No. 1, Jan.-Mar. 2004.

[6] M. Azadmanesh and R. Kieckhafer. Exploiting Omissive Faults in Synchronous Approximate Agreement. *IEEE Transactions on Computers*, Vol. 49, No. 10, Oct. 2000, p. 1031-42.

[7] G. Bauer and M. Paulitsch. An Investigation of Membership and Clique Avoidance in TTP/C. *Proceedings of 19th IEEE Symposium on Reliable Distributed Systems SRDS-2000*, p. 118-24.

[8] G. Bauer, H. Kopetz, and P. Puschner. Assumption Coverage under Different Failure Modes in the Time-Triggered Architecture. *8th IEEE Intl. Conf. on Emerging Technologies and Factory Automation*, Oct. 2001.

[9] G. Bauer, H. Kopetz and W. Steiner. The Central Guardian Approach to Enforce Fault Isolation in the Time-Triggered Architecture. *Proceedings of the Sixth IEEE International Symposium on Autonomous Decentralized Systems (ISADS '03)*, April 2003, p. 37-44.

[10] E. Borgstrom. EMC Requirements for Avionics: RTCA/DO-160D, Change No. 3. *Interference Technology Annual Guide 2003*, 2003, p. 34-7.

[11] A. Bouajjani and A. Merceron. Parametric Verification of a Group Membership Algorithm. *Proceedings of the 7th Intl. Symposium on Formal Techniques in Real-Time and Fault Tolerant Systems*, Sept. 2002, p. 311-30.

[12] J. Burch, E. Clarke, and K. McMillan. Symbolic Model Checking: 10^{20} States and Beyond. *Information and Computation*, Vol. 98, No. 2, June 1992.

[13] R. Butler. The SURE Approach to Reliability Analysis. *IEEE Trans. on Reliability*, Vol. 41, No. 2, June 1992, p. 210-18.

[14] R. Butler and S. Johnson. Techniques for Modeling the Reliability of Fault-Tolerant Systems With the Markov State-Space Approach. NASA RP-1348, Sept. 1995.

[15] R. Butler and G. Finelli. The Infeasibility of Experimental Quantification of Life-Critical Software Reliability. *Proc. of the ACM SIGSOFT '91 Conf. on Software for Critical Systems*, Dec. 1991, p. 66-75.

[16] P. Caspi, A. Curic, A. Maignan, C. Sofronis, S. Tripakis, and P. Niebert. From Simulink to SCADE / Lustre to TTA: A Layered Approach for Distributed Embedded Systems. *Submitted to the 2002 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES 2002)*, Oct. 2002.

[17] E. Chan, Q. Le, and M. Beranek. High Performance, Low-Cost Chip-on-Board (COB) FDDI Transmitter and Receiver for Avionics Applications. *Proc. of the 1998 Electronic Components and Tech. Conf.*, 1998, p. 410-17.

[18] T. Chandra and S. Toueg. Unreliable Failure Detectors for Reliable Distributed Systems. *Journal of the ACM*, Vol. 43, Issue 2, March 1996.

[19] G. Chockler, I. Keidar, and R. Vitenberg. Group Communication Specifications: A Comprehensive Survey. *ACM Computing Surveys*, Vol. 33, No. 4, Dec. 2001, p. 427-69.

[20] G. Ciardo, J. Muppula, and K. Trivedi. SPNP: Stochastic Petri Net Package. *Proceedings of the Third International Workshop on Petri Nets and Performance Models (PNPM '89)*, 1989, p. 142-51.

[21] J. Clark and D. Pradhan. Fault Injection: A Method for Validating Computer-System Dependability. *Computer*. Vol. 28, Issue 6, June 1995, p. 47-56.

[22] C. Constantinescu. Impact of Deep Submicron Technology on Dependability of VLSI Circuits. *Proc. of the Intl. Conf. on Dependable Systems and Networks, DSN'02*, June 2002.

[23] M. Cukier and D. Powell. Coverage Estimation Methods for Stratified Fault Injection. *IEEE Transactions on Computers*, Vol. 48, No. 7, July 1999.

[24] D. Davies and J. Wakerly. Synchronization and Matching in Redundant Systems. *IEEE Transactions on Computers*, Vol. c-27, No. 6, June 1978.

[25] DBench Project. Fault Representativeness. Deliverable ETIE2. IST 2000-25425. June 2000.

[26] Y. Deswarte, K. Kanoun, and J.-C. Laprie, Diversity Against Accidental and Deliberate Faults. *Proc. of Computer Security, Dependability and Assurance: From Needs to Solutions*, 1998.

[27] P. Dodd and L. Massengill. Basic Mechanisms and Modeling of Single-Event Upset in Digital Microelectronics. *IEEE Trans. on Nuclear Science*, Vol. 50, No. 3, June 2003, p. 583-602.

[28] D. Dolev, C. Dwork, and L. Stockmeyer. On the Minimal Synchronism Needed for Distributed Consensus. *Journal of the Association for Computing Machinery*, Vol. 34, No. 1, January 1987, p. 77-97.

[29] D. Dolev, N. Lynch, S. Pinter, E. Stark, and W. Weihl. Reaching Approximate Agreement in the Presence of Faults. *Journal of the Association for Computing Machinery*, Vol. 22, No. 3, July 1986, p. 499-516.

[30] K. Driscoll, B. Hall, H. Sivencrona, and P. Zumsteg. Byzantine Fault Tolerance, from Theory to Reality. *Proc. of the 2003 Intl. Conf. on Computer Safety, Reliability, and Security (SAFECOMP 2003)*, Sept. 2003, p. 235-48.

[31] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the Presence of Partial Synchrony. *Journal of the Association for Computing Machinery*, Vol. 35, No. 2, April 1988, p. 288-23.

[32] Federal Aviation Administration. Instructions for Continued Airworthiness: Maintenance Tasks for High Integrity Radio Frequency (HIRF)/Electromagentic Interference (EMI)/Lightning Protection Features. Advisory Circular 33.4.3 [Draft].

[33] FlexRay Consortium. FlexRay Communications System Protocol Specification, Version 2.0. June 2004.

[34] F. Fich and E. Ruppert. Hundreds of Impossibility Results for Distributed Computing. *Distributed Computing* 2003, 16:121-63.

[35] M. Fischer and N. Lynch. A Lower Bound for the Time to Assure Interactive Consistency. *Information Processing Letters*, 14(4):183-86, June 1982.

[36] M. Fischer, N. Lynch and M. Paterson. Impossibility of Distributed Consensus with One Faulty Process. *Journal of the Assocation for Computing Machinery*, Vol. 32, No. 2, April 1985, p. 374-82.

[37] Ford Motor Company. ES-XW7T-1A278-AC: Component and Subsystem Electromagenetic Compatibility. Worldwide Requirements and Test Procedures. Oct. 10 2003.

[38] E. Fuchs. Software Implemented Fault Injection (A Literature Survey). Research Report, Technische Universität Wien, Institut für Technische Informatik. Nov. 1994.

[39] A. Galleni and D. Powell. Consensus and Membership in Synchronous and Asynchronous Distributed Systems. Technical Report. LAAS-CNRS No. 96104. 1995.

[40] A. Geser and P. Miner. A Formal Correctness Proof of the SPIDER Diagnosis Protocol. *Proc. of the 15th International Conference on Theorem Proving in Higher Order Logics (TPHOLS)*, Aug. 2002.

[41] A. Geser and P. Miner. A New On-Line Diagnosis Protocol for the SPIDER Family of Byzantine Fault Tolerant Architectures. NASA/TM-2003-212432, April 2003.

[42] P. Goddard. Validating the Safety of Embedded Real-Time Control Systems Using FMEA. *Proceedings of the 1993 Annual Reliability and Maintainability Symposium*, 1993, p. 227-30.

[43] P. Goddard. A Combined Analysis Approach to Assesing Requirements for Safety Critical Real-Time Control Systems. *Proceedings of the 1996 Annual Reliability and Maintainability Symposium*, 1996, p. 110-15.

[44] P. Goddard. Software FMEA Techniques. *Proceedings of the 2000 Annual Reliability and Maintainability Symposium*, 2000, p. 118-23.

[45] S. Gokhale, W. Wong, K. Trivedi, and J. Horgan. An Analytical Approach to Architecture-Based Software Reliability Prediction. *Proceedings of the 1998 International Performance and Dependability Symposium (IPDS)*, 1998.

[46] P. Herout, S. Racek, and J. Hlavička. Model-Based Dependability Evaluation Method for TTP/C Based Systems. *4th European Dependable Computing Conference (EDCC 2002)*, LNCS 2485, 2002, p. 271-82.

[47] G. Holzmann. The Model Checker SPIN. *IEEE Transactions on Software Engineering*, Vol. 23, No. 5, May 1997.

[48] G. Huffines and R. Orville. Lightning Ground Flash Density and Thunderstorm Duration in the Continental United States: 1989-96. *Journal of Applied Meteorology 38*, No. 7, July 1999, p. 1013-19.

[49] R. Hyle, Jr. Fiber Optics - Failure Modes and Mechanisms. *Proc. of the Reliability and Maintainability Symp.*, 1992, p. 379-88.

[50] IABG. V-Model: Development Standard for IT-Systems of the Federal Republic of Germany. Lifecylce Process Model. 1997.

[51] Institute of Electrical and Electronics Engineers. IEEE Draft P802.3ah/D1.1, Amendment to IEEE Std 802.3-2002, Ethernet in the First Mile, Oct. 2002.

[52] International Electrotechnical Commission. IEC 61000-4-4. Electromagnetic Compatibility (EMC) - Part 4-4: Testing and Measurement Techniques - Electrical Fast Transient/Burst Immunity Test. July 2004.

[53] International Electrotechnical Commission. IEC 61375-7, Annex B: Guidelines for Conformance Testing. May 1998.

[54] International Organization for Standardization. ISO 7637-1, ISO 7637-2, ISO 7637-3. Road Vehicles – Electrical Disturbances from Conduction and Coupling. March 2002, June 2004, Nov. 1995.

[55] International Organization for Standardization and International Electrotechnical Commission. ISO/IEC 7498-1:1994. Information Technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. 1994.

[56] R. Jain. Error Characteristics of the Fiber Distributed Data Interface (FDDI). *IEEE Transactions on Communications*, Vol. 38, No. 8, Aug. 1990, p. 1244-52.

[57] M. Kaaniche, J.-C. Laprie, and J.-P. Blanquart. Dependability Engineering of Complex Computing Systems. *6th IEEE International Conference on Complex Computer Systems (ICECCS '00)*. Sept. 2000.

[58] R. Kieckhafer, M. Azadmanesh, and Y. Hui. On the Sensitivity of NMR Unreliability to Non-Exponential Repair Distributions. *Fifth IEEE Symposium on High Assurance Systems Engineering (HASE 2000)*, Nov. 2000, p. 293-300.

[59] R. Kieckhafer and M. Azadmanesh. Reaching Approximate Agreement with Mixed-Mode Faults. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 5, No. 1, Jan. 1994.

[60] R. Kieckhafer, C. Walter, A. Finn, and P. Thambidurai. The MAFT Architecture for Distributed Fault Tolerance. *IEEE Transactions on Computers*, Vol. 37, No. 4, April 1988.

[61] K. Kim and E. Shokri. Minimal-Delay Decentralized Maintenance of Processor-Group Membership in TDMA-Bus LAN Systems. *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS '93)*, p. 410-19.

[62] H. Kopetz. Fault Containment and Error Detection in the Time-Triggered Architecture. *Proc. of the 6th Intl. Symp. on Autonomous Decentralized Systems (ISADS '03)*, Apr. 2003.

[63] M. Kull, K. Feser, and W. Köhler. A New Test System for Measurements of Fast Transients in Passenger Cars. *2003 IEEE Symposium on EMC*, Symposium Record Vol. 1, Aug. 2003, p. 425-28.

[64] H. Kunkel and T. Moyer. Update on IEC 61000-4-4 (EFT/Burst Test). *Interference Technology*, Annual Guide 2003, p. 24-28.

[65] M. Kwiatkowska, G. Norman and D. Parker. Controller Dependability Analysis By Probabilistic Model Checking. *Proc. of the 11th IFAC Symp. on Information Control Problems in Manufacturing (INCOM '04)*, Apr. 2004.

[66] L. Lamport, R. Shostak, and M. Pease. The Byzantine Generals Problem. *ACM Transactions on Programming Language Systems*, Vol. 4, No. 3, July 1982, p. 382-401.

[67] E. Latronico, P. Miner, and P. Koopman. Quantifying the Reliability of Proven SPIDER Group Membership Service Gaurantees. *Proc. of the Intl. Conf. on Dependable Systems and Networks (DSN '04)*, June 2004.

[68] E. Latronico. Problems Facing Group Membership Specifications for X-by-Wire Protocols. Student Paper. *Proc. of the Intl. Conf. on Dependable Systems and Networks (DSN '03)*, June 2003.

[69] N. Leveson. Safeware: System Safety and Computers. Addison Wesley, 1995.

[70] C. Liu and J. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, Vol. 20, Issue 1, Jan. 1973.

[71] M. Lutz, R. Casanova, and T. Revesz. Induced Lightning Testing of Avionics - With Single Stroke, Multiple Stroke, and Multiple Burst. *Proc. of the 8th International Conference on Electromagnetic Interference and Compatibility*, 2003.

[72] N. Lynch. A Hundred Impossibility Proofs for Distributed Computing. *Proceedings of the Eighth Annual ACM Symposium on Principles of Distributed Computing*, 1989.

[73] MAXIM Integrated Products. Accurately Estimating Optical Receiver Sensitivity. Application Note HFAN-3.0.0. Oct. 2001.

[74] F. Meyer and D. Pradhan. Consensus with Dual Failure Modes. *Proc. of the 17th Fault-Tolerant Computing Symp.*, July 1987, p. 48-54.

[75] P. Miner, A. Geser, L. Pike, and J. Maddalon. A Unified Fault-Tolerance Protocol. *Formal Techniques in Fault-Tolerance and Real-Time Systems*, 2004.

[76] Motor Industry Software Reliability Association. Development Guidelines for Vehicle Based Software. November 1994. PDF version 1.1, January 2001.

[77] D. Montgomery and G. Runger. Applied Statistics and Probabilty for Engineers. Second Edition. John Wiley and Sons, Inc., 1999.

[78] J. Morris, D. Kroening, and P. Koopman. Fault Tolerance Tradeoffs in Moving from Decentralized to Centralized Embedded Systems. *Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN 04)*, June 04.

[79] E. Normand. Single-Event Effects in Avionics. *IEEE Trans. on Nuclear Science*, Vol. 43, No. 2, April 1996, p. 461-74.

[80] E. Normand. Single Event Upset at Ground Level. *IEEE Trans. on Nuclear Science*, No. 6, Ser. 1, Dec. 1996, p. 2742-50.

[81] E. Normand, J. Wert, P. Majewski, D. Oberg, W. Bartholet, S. Davis, S. Wender, and A. Gavron. Single Event Upset and Latchup Measurements in Avionics Devices Using the WNR Neutron Beam and a New Neutron-Induced Latchup Model. *Radiation Effects Data Workshop*, NSREC '95, IEEE, July 1995.

[82] M. O'Bryan, K. LaBel, J. Howard Jr., C. Poivey, R. Ladbury, S. Kniffin, S. Buchner, M. Xapsos, R. Reed, A. Sanders, C. Seidleck, C. Marshall, P. Marshall, J. Titus, K. Li, J. Gambles, R. Stone, J. Patterson, H. Kim, D. Hawkins, M. Cearts, J. Forney, T. Irwin, Z. Kahric, S. Cox, and C. Palor. Single Event Effects Results for Candidate Spacecraft Electronics for NASA. *Radiation Effects Data Workshop*. July 2003, p. 65-76.

[83] T. J. O'Gorman. The Effects of Cosmic Rays on the Soft Error Rate of a DRAM at Ground Level. *IEEE Trans. on Electron Devices*, Vol. 41, No. 4, April 1994.

[84] F. Ortmeier and W. Reif. Safety Optimization: A Combination of Fault Tree Analsis and Optimization Techniques. *Proc. of the 2004 Intl. Conf. on Dependable Systems and Networks (DSN '04)*, Jun. 2004.

[85] N. Ou, T. Farahmand, A. Kuo, S. Tabatabaei, and A. Ivanov. Jitter Models for the Design and Test of Gbps-Speed Serial Interconnects. *IEEE Design and Test of Computers*, July 2003, P. 302-13.

[86] Analysis and Test of Bus Systems. PALBUS Task 10.2 and 10.3. SP Swedish National Testing and Research Institute, 2001.

[87] M. Pease, R. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *Journal of the Association for Computing Machinery*, Vol. 27, No. 2, April 1980, p. 228-34.

[88] H. Pfeifer. Formal Verification of the TTP Group Membership Algorithm. *Proc. of FORTE XIII / PSTV XX*, Oct. 2000, p. 3-18.

[89] H. Pfeifer, D. Schwier, and F. W. von Henke. Formal Verification for Time-Triggered Clock Sychronization. *7th IFIP Intl. Working Conf. on Dependable Computing for Critical Applications (DCCA-7)*, Jan. 99, p. 207-26.

[90] B. Portwood. Current Government and Industry Developments in the Area of System Safety Assessment. *Proceedings of the 17th Digital Avionics Systems Conference*, Nov. 1998, Vol. 1, p. B31 1-6.

[91] D. Powell. Failure Mode Assumptions and Assumption Coverage. *Proc. of the 22nd Intl. Symp. on Fault-Tolerant Computing (FTCS '92)*, 1992, p. 386-95.

[92] D. Powell. Failure Mode Assumptions and Assumption Coverage. Research Report 91462. Revised March 30, 1995.

[93] Robert Bosch GmbH. Controller Area Network Specification. Version 2.0. Sept. 1991.

[94] RTCA, Inc. Environmental Conditions and Test Procedures for Airborne Equipment. Document No. RTCA/Do-160D, July 29, 1997.

[95] J. Rushby and D. Stringer-Calvert. A Less Elementary Tutorial for the (PVS) Specification and Verification System. SRI Tech Report, CSL-95-10. June 1995, Revised July 1996.

[96] J. Rushby. A Comparison of Bus Architectures for Safety-Critical Embedded Systems. NASA CR-2003-212161. March 2003.

[97] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems. P. Panangaden and F. van Breugel, editors. American Mathematical Society. 2000.

[98] U. Schmid, B. Weiss, and J. Rushby. Formally Verified Byzantine Agreement in Presence of Link Faults. *Proc. of the 22nd Intl. Conf. on Distributed Computing Systems, ICDCS'02*, July 2002.

[99] F. Sexton. Destructive Single-Event Effects in Semiconductor Devices and ICs. *IEEE Trans. on Nuclear Science*, Vol. 50, No. 3, June 2003, p. 603-21.

[100] I. Shake, H. Takara, K. Uchiyama, Y. Yamabayashi. Quality Monitoring of Optical Signals Influence by Chromatic Dispersion in a Transmission Fiber Using Averaged Q-Factor Evaluation. *IEEE Photonics Technology Letters*, Vol. 13, No. 4, April 2001, p. 385-87.

[101] J. Shandle. CAN: Network for Thousands of Applications Outside Automotive. *TechOnLine*, June 26, 2003.

[102] K. Shin and P. Ramanthan. Diagnosis of processors with Byzantine faults in a distributed computing system. *17th Fault Tolerant Computing Symposium (FTCS 17)*, 1987, p. 55-60.

[103] D. Sieworek and R. Swarz. Reliable Computer Systems: Design and Evaluation. Second edition. Digital Press, Digital Equipment Corporation. 1992.

[104] Sigmaplot 2002 for Windows Version 8.0. SPSS Inc. 2002.

[105] H. Sivencrona. On the Design and Validation of Fault Containment Regions in Distributed Communication Systems. Dissertation. Chalmers University of Technology, 2004.

[106] H. Sivencrona, J. Hedberg, and H. Röcklinger. Comparative Analysis of Dependability Properties of Communication Protocols in Distributed Control Systems. PALBUS Task 10.2. Apr. 2001.

[107] I. Sommerville. Software Engineering. Fifth Edition. Addison-Wesley Publishers Ltd. 1995.

[108] A. Spillner. From V-model to W-model - Establishing the Whole Test Process. *Proceedings of the 4th Conference on Quality Engineering in Software Technology and VDE-ITG Workshop on Testing Non-Functional Software Requirements*, 2000, p. 222-31.

[109] W. Steiner, J. Rushby, M. Sorea, and H. Pfeifer. Model Checking a Fault-Tolerant Startup Algorithm: From Design Exploration to Exhaustive Fault Simulation. *Proceedings of the 2004 International Conference on Dependable Systems and Networks*, July 2004.

[110] R. Stephens. Analyzing Jitter at High Data Rates. *IEEE Optical Communications*, Feb. 2004, p. S6-10.

[111] W. Stephens, T. Banwell, G. Lalk, T. Robe, and K. Young. Transmission of STS-3c (115 Mbit/sec) SONET/ATM Signals over Unshielded and Shielded Twisted Pair Copper Wire, *Proc. of the GLOBECOM'92*, IEEE, Vol.1., Dec. 1992, p. 170-74.

[112] N. Storey. Safety-Critical Computer Systems. Addison Wesley Longman, 1996.

[113] K. Sullivan, J. Bechta Dugan, and D. Coppit. The Galileo Fault Tree Analysis Tool. *Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing*, June 1999.

[114] J. Rushby. Reconfiguration and Transient Recovery in State Machine Architectures. *Proc. of the 26th International Symposium on Fault-Tolerant Computing, FTCS-26*, 1996, p. 8-15.

[115] P. Thambidurai and Y.-K. Park. Interactive Consistency With Multiple Failure Modes. *Proc. of the Seventh Reliable Distributed Systems Symp.*, Oct. 1998.

[116] K. Tindell and A. Burns. Guaranteeing Message Latencies on Control Area Network (CAN). *Proceedings of the 1st International CAN Conference*, 1994.

[117] W. Torres-Pomales, M. Malekpour, and P. Miner. ROBUS-2: A Fault-Tolerant Broadcast Communication System. NASA/TM-2005-213540. March 2005.

[118] TTTech Computertechnik AG. Time Triggered Protocol TTP/C High-Level Specification Document, Protocol Version 1.1. Specification Edition 1.4.3. Nov. 2003.

[119] United States Department of Defense. MIL-HDBK-217F. Reliability Prediction of Electronic Equipment. Dec. 2, 1991.

[120] J. Lundelius Welch and N. Lynch. A New Fault-Tolerant Algorithm for Clock Synchronization. *Information and Computation*, Vol. 77, No. 1, Apr. 1988, p. 1-36.

[121] J. Wensley. SIFT Software Implemented Fault Tolerance. *Proceedings of the Fall Joint Computer Conference*, 1972, p. 243-53.

[122] J. Wensley, M. Green, K. Levitt, and R. Shostak. The Design, Analysis, and Verification of the SIFT Fault Tolerant System. *Proceedings of the 2nd International Conference on Software Engineering*, 1976.

[123] A. White. Markov Reliability Models. *Prepared for the 20th International System Safety Coference*, Aug. 2002.

[124] A. White. Worst Pattern Analysis for Markov Reliability Models. *Proceedings of the 21st Digital Avionics Systems Conference*, Vol. 2, Oct. 2002.

[125] S. Wielandy, M. Fishteyn, and B. Zhu. Optical Performance Monitoring Using Non-linear Detection. *Journal of Lightwave Technology*, Vol. 22, No. 3, March 2004, p. 784-93.

[126] WLEX-TV. Facts About Lightning. 2005.

[127] J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin. Separating Agreement from Execution for Byzantine Fault Tolerant Services. *19th ACM Symp. on Operating Systems Principles (SOSP-2003)*, 2003.

[128] L. Yin, M. A. J. Smith, and K. Trivedi. Uncertainty Analysis in Reliability Modeling. *Proceedings of the 2001 Annual Reliability and Maintainability Symposium*, 2001, p. 229-34.

[129] H. Zimmerman. OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communications*, Vol. Com-28, No. 4, April 1980.