The Customizable Fault/Error Model for Dependable Distributed Systems*

C.J. Walter
WW Technology Group
4519 Mustering Drum
Ellicott City, MD 21042
cwalter@cs.jhu.edu

N. Suri
Chalmers University
Dept. of Computer Engineering
Goteborg, S 41296, Sweden
suri@ce.chalmers.se

Abstract

Dependability is a qualitative term referring to a system's ability to meet its service requirements in the presence of faults. The types and numbers of faults covered by a system play a primary role in determining the level of dependability which that system can provide. Given the variety and multiplicity of fault types, to simplify the design process, the system algorithm design often focuses on specific fault types, resulting in either over-optimistic (all fault permanent) or over-pessimistic (all faults malicious) dependable system designs.

A more practical and realistic approach is to recognize that faults of varied severity levels and of differing occurrence probabilities may appear as combinations *rather than the assumed single fault type occurrences. Allowing the user to select/customize a particular* combination *of fault types of varied severity characterizes the proposed Customizable Fault/Error Model (CFEM). The CFEM organizes diverse fault categories into a cohesive framework by classifying faults by the effect they have on the required system services rather than targeting the source of the fault condition. In this paper, we develop (a) the complete framework for the CFEM fault classification, (b) the voting functions applicable under the CFEM, and (c) the fundamental distributed services of consensus and convergence under the CFEM on which dependable distributed functionality can be supported.*

Keywords: fault modeling, error classification, dependability

1 Introduction

Dependability is a qualitative term describing the confidence that can be placed in a computing system's ability to deliver the expected service, even in the presence of faults [9], where the term expected service also includes the notion of timeliness. Dependability comprises the issues and techniques most commonly used to identify, implement, and measure system fault-tolerance and real-time performance.

The effectiveness of a fault-tolerance method always relies on the realism and accuracy of the assumed underlying fault and error models. The faults tolerated by a system play a primary role in the level of dependability that a system can achieve, as quantified, for example, by the reliability metric. Typically,

*Supported in part by ONR Grant #N00014-91-C-0014

a *fault* is defined as an anomalous physical condition, the identified or hypothesized cause of an *error*, which may eventually lead to a loss of service i.e., *failure*.

The fault models used in designing and analyzing dependable distributed systems typically make simplifying assumptions about the natures of faults[1] in the system. Often, the fault tolerance algorithms employed by a system treat all faults identically, ignoring the effects of any fault types the algorithm is not designed to distinguish or to tolerate. Such overly-optimistic single fault-type models assume a fixed number of benign permanent faults and perfect fault coverage. Or, the system model employs complex protocols that assume all faults to be malicious, even though only a small portion of the faults may actually require such protection. These more pessimistic Byzantine models assume all faults to be arbitrary. By distinguishing different fault types and considering varying probabilities of occurrence of each fault type, we can develop more realistic system models to design algorithms capable of handing the various fault types. In this paper, we utilize the rationale of [2, 13, 15, 16, 19] and develop the *Customizable Fault/Error Model*(CFEM), which considers mixed fault types (as selected/customized by the user) along with the algorithms needed to tolerate such faults.

Under the CFEM, the set of all faults is partitioned into three disjoint classes based on fault effects: *non-malicious*, *malicious symmetric*, and *malicious asymmetric*. Then, the type of algorithm required to detect or mask the subset of faults that is assumed to occur is indicated as a function of the fault type. This matching of fault type to algorithm is important in ensuring adequate, yet cost effective, system fault coverage. If the fault tolerance techniques implemented do not support segregation and handling of mixed faults, then the CFEM reverts to the single fault-type models, with no improvement. Thus, existing fault tolerance methods must be extended to properly utilize the proposed CFEM. The main difference between our CFEM based algorithms and existing fault models based algorithms is our use of dynamic fault tolerance, based on the ability to detect certain types of faults locally. We combine the benefits of the models assuming perfect fault detection with the increased coverage of the more realistic models. An important contribution is the development algorithms that achieve *consensus* (exact agreement) and *convergence* (approximate agreement) without the classically required condition that all participants have the same number of values on which to vote.

The CFEM provides a framework which can be used to enhance the dependability of processes and systems. The CFEM philosophy takes a practical view of faults and their effects on system operations, recognizing that a system cannot tolerate unlimited set of faults, and that the fault tolerance techniques implemented in a system play a large role in determining the covered fault-effects. Variations in the probabilities of occurrence of different faults, the severity of their effects, and the number of faulty units which can be tolerated during different phases of system operation are exploited to improve the coverage provided by CFEM. The classical **single** fault-type instantiation is replaced by a **fault-tuple**, chosen (or customized) by the user, representing the **set** of fault-effects of varied severity which may exist *concurrently* in the system, and are chosen to be protected against. The fault resiliency of critical system functions such as synchronization, data voting, and other consensus-based operations is also improved through the consideration of CFEM.

Contributions

The contributions of this paper are developed over the following Sections. In Section 2, we present the distributed system models, review existing fault taxonomies, and present the basis for the CFEM based fault/error classification. Sections 3 and 4 focus on developing the theoretical framework to support CFEM. Section 3 develops the varied fault scenarios valid under CFEM and also presents a comparative

[1] A *fault* is the identified or hypothesized cause of an error. An *error* is the manifestation of a fault, an undesired state either at the boundary or at an internal point in the system or process. A *failure* is the inability of the system or component to provide the specified service caused by an error.

analysis of the CFEM enhanced fault handling capabilities compared to classical fault models. In Section 4, the requisite fault tolerant voting functions supporting CFEM are developed. In Section 5 we present the formal framework and analysis in CFEM's providing the fundamental distributed services of consensus and convergence. We summarize and discuss the impact of our contributions in Section 6.

2 Fault/Error Models in Distributed Systems

System Models

For simplicity of discussion, we adopt a generalized system model, although the CFEM and the other fault models are applicable to a variety of system models such as synchronous, asynchronous, full/partial connectivity etc.. A *system* is comprised of a set of *nodes*. Nodes communicate by exchanging information (messages) across links, with a bounded delay assumed for message generation, delivery and processing. We use the terms "system," "nodes," "links," and "messages" in an abstract sense, because some type of information exchange often exists between a node and *its* components, or between a process and its subprocesses.

We assume a fully connected system consisting of *nodes* which communicate using synchronous message passing, with an upper bound on the time required for a node to generate and send a message. Individual nodes make decisions and compute values based on information received in messages from other nodes. The status of a node, faulty or good, is discerned by other nodes through the contents of messages originating from the target node, or through the lack of an expected message from that node, i.e., nodes monitor the messages from other nodes to infer the presence of a fault-effect. As in [16] and [8], a non-faulty node can always identify the sender of a message it receives and can detect the absence of an expected message. The system fails when consistent decisions or computations across the system are no longer possible.

In our assumed system model the "fault floor" is at the node level. That is, regardless of how many failed components exist in a given node, only the node's fault status, as viewed by other nodes, is of interest in determining the system fault resiliency.[2] Since the system design objective is to provide continual correct service, we focus on faults which may affect distributed computation of critical system functions. As described in subsequent sections, the CFEM framework is based on classification of faults based on each node's analysis of inter-node communications. Membership in the fault-effect classes is determined by the system topology and by the fault tolerance methods implemented in a specific system. This linking of fault-effects to the methods implemented (or required) to tolerate them is unique to the CFEM.

2.1 Existing Fault Classifications

A fault is classically defined as the cause, real or perceived, of errors which can lead to failure. Much effort has been expended and many taxonomies have been proposed to capture the important attributes of faults. Laprie [9] classifies faults according to the attributes of *nature (accidental or intentional)*; *cause (physical or human-made)*; *location (internal or external)*; *phase of creation (design or operational)*; and *persistence (permanent or temporary)*. The fault attributes of Laprie have been augmented to include *activity (active or dormant)* and *value (stationary, non-stationary)* [1]. This taxonomy is useful in identifying faults to be removed during the design and testing process.

[2] Link failures are not directly addressed in this model, but are ascribed to the perceived sender of the message. However, the fault floor could be extended to node and link failures if desired.

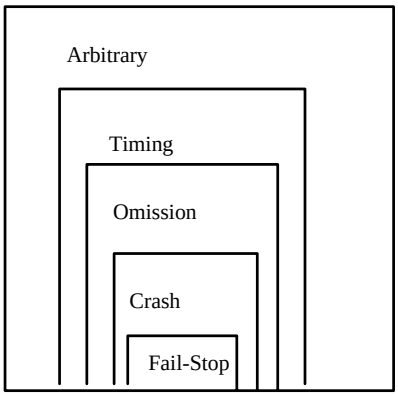


Figure 1 Onion Fault Model

While these are classifications and attributes are important and commonly used in the literature, they do not furnish all the information about faults that is required by the designers or assessors of a system. They provide insufficient indication of the types of faults that can occur in a given system, or how to avoid, detect, or tolerate the faults. It is assumed that the classes within each attribute are disjoint; so, for a given attribute, a specific fault *in a specific system* can belong to exactly one class. However, if the same type of fault occurs in two different systems, it is possible for the fault to be classified differently in the two systems with respect to a given attribute. Such fault classification ambiguity can arise because the systems in which the fault occurs can have vastly different requirements. For example, a fault lasting a few seconds might be sufficient for a fault handler in one system to diagnose a permanent fault and a temporary fault in another. Differences can also occur due to the location of the fault in the systems, how long the fault is active relative to the time scale (mission time) for each system, the technology used to construct each system, the assumed system environment, and other related factors.

A different approach based on the use of failure semantics for dependable systems with real-time constraints and responsive systems[12] is the onion model [4, 3]. Figure 1, taken from [4], shows the relationships among the five potential failure modes of the onion model, given in increasing order of severity: fail-stop, crash, omission, timing and arbitrary. Under the *fail-stop* assumption, a component[3] fails by ceasing execution; no incorrect state transition occurs, and all other good components can detect this failure. A *crash* or *fail-silent* mode is identical to fail-stop, except that detection by all good components is not guaranteed. An *omission* mode occurs when a component fails to respond to an input, perhaps undetectably to some components. A functionally correct, but untimely response corresponds to a *timing* fault. Any other mode of behavior is classified as an *arbitrary* fault. The more restricted the assumed fault mode, the stronger the assumed failure semantics. Thus, weak failure semantics correspond to little restriction of the behavior of a faulty component. Note also in Figure 1 that the containment relationship among the different classes is such that limiting the failure semantics of a host to, say, timing, means that the host can fail in any of the modes subsumed: fail-stop, crash, or omission. The onion fault model fails to capture the notion of a system comprised of nodes, and does not aid in ensuring that correct system operations are sustained in the presence of a node failure. The focus of this model on abstract and high level faults limits its coverage of data faults. It is to address such limitations of exiting fault models that we propose the CFEM.

[3] In this context, the term component may refer to hardware, software, or a combination of the two.

2.2 The Customizable Fault/Error Model Taxonomy

The CFEM is a model that focuses on runtime fault-effects and the methods associated with tolerating them. We do not attempt to deal with all possible fault occurrences specifically but rather form classes determined by the potential fault effects. Faults classified using other taxonomies can be mapped into the classes of the CFEM framework. The CFEM fault classes are determined by the fault-tolerance techniques (detection and masking) implemented in the system and the system topology. Under the CFEM, node faults are classified according to the ability of the system to tolerate their effects. At the highest level, the set of all faults is partitioned into *tolerable* and *intolerable* faults according to their effects.

Intolerable fault-effects are faults whose effects cannot be detected, masked, or otherwise tolerated by the system quickly enough to prevent them from causing a loss of service or improper service with consequences greatly exceeding the benefits of proper system service. Design faults, generic faults, common mode faults, physical damage faults and other catastrophic faults that immediately render much of a system useless would be categorized as intolerable. In some applications, it may be necessary to distinguish among different types of intolerable faults. Some fault effects are intolerable because no technique could possibly mask or detect them; faults that destroy substantial portions of the system fall into this category. Other fault effects are intolerable because they are uncovered by design or attrition. That is, the system fault model incorrectly assumes that such faults will not occur and the chosen system fault handling functions do not cover that type of fault; or, an otherwise covered fault is not handled because enough system resources are not available to mask or detect it. Since it is impossible to model or predict the occurrence of such faults, perfect fault coverage to a few non-coincident non-catastrophic faults is assumed, with the probability of catastrophic faults assumed to be very small.

Tolerable fault effects, or covered fault effects are the ones of direct interest to us in establishing the CFEM. We first present the detection setup in a node on which the CFEM is based. As shown in Figure 2, when an error from a faulty node is detected by some non-faulty node, the sending node is identified locally as *potentially* benign. All good nodes must agree on this local judgment for the fault effects to be classified as benign. The instantaneous classification of all faults requires the globally omniscient view in Figure 2. In practice, distributed information distribution and dissemination techniques are used to approximate the global view in the presence of potentially malicious faults [18, 19].

In the CFEM, all fault classification is based on **(1)** *Local-Classification* of fault-effects to the extent permitted by the fault-detection mechanisms built in at the node level, and **(2)** *Global-Classification* based on nodes exchanging their local-classification with other system nodes to develop a global opinion on the fault-effect. We now describe in greater detail the rationale behind the individual classifications.

The CFEM classification[19] begins with the examination of the possible fault states created by the dispersal of information. In the abstract, we can assume that when a node transmits information, two general cases are possible as illustrated in Figure 3: **(a)** all receivers obtain the same information i.e., $mess_j$ *(symmetric dispersal)*, or **(b)** receivers obtain different information i.e., $mess_j$ and $mess'_j$ (*asymmetric dispersal*). This abstraction is sufficient since we are primarily concerned with maintaining consistency and these two possibilities reflect whether it is preserved or not.

The second aspect of the CFEM is based on extent of the fault-effect as created by the faulty source and the way in which it has propagated its effects. If an individual node is sufficient to detect the error from an incoming message, we classify this case as a *benign sender fault* since the fault-effect is *locally detectable* at the receiver. On the other hand, there are situations that require multiple nodes to exchange their syndrome information with each other in order to provide accurate diagnosis, i.e., *globally detectable*. For these cases, the values in a message appear to be plausible locally at a node; however, they can only be verified with a multiple exchange of information. After the exchange is completed, the

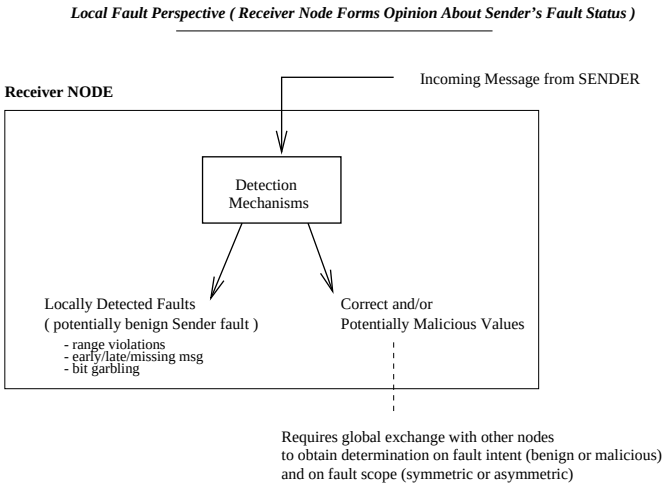
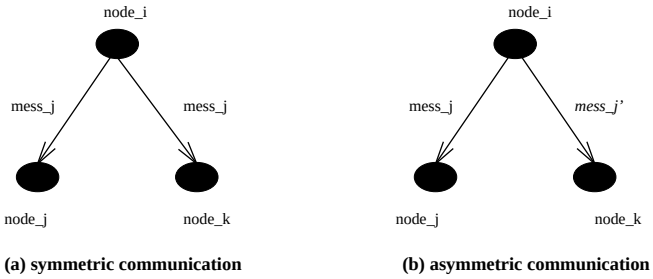


Figure 2 CFEM Fault Classification Basis



(a) symmetric communication **(b) asymmetric communication**

	locally detectable	*globally detectable*
symmetric communication	**Benign**	**Value**
asymmetric communication	**Benign**	**Value** (arbitrary faulty)

Figure 3: CFEM Fault Space (communication symmetry vs. detectability)

plausible value may be determined to be *value-faulty*.

As an example, consider the acceptable range of the value contained in a message to be [0,100], with the expected correct value of 50 in a message. A node accepts a message without registering any fault as long as the value is within [0,100]. If a faulty node sends values 30 and 70 as values in the messages (instead of the correct value of 50) to the other two nodes, both of these messages are within the acceptable range [0,100], thus these erroneous values are not locally detectable at the receiving nodes. Only when the two receiving nodes exchange (or distribute) their received message values of 30 and 70 respectively with each other, does the asymmetric nature of the faulty values comes across i.e., global detectability of value-faults.

Figure 3 summarizes the specific cases covered by the CFEM. The vertical axis reflects the possibilities for information dispersal and the horizontal axis represents the types of detectability. The fault-effects are represented by the table entries and consists of two major types covered in each vertical column: benign and value faults. The *benign* faults can be dispersed in a symmetric or asymmetric manner; however, these faults are detectable by all non-faulty nodes with fault detection mechanisms implemented locally. Any detectable fault in a message will result in the sender being classified as *locally benign faulty ($\mathcal{B}$),* by the receiver. The rightmost vertical column of the table covers *value* faults which are the locally undetectable class of faults. Messages which pass all data validity and range deviance checks, but provide a valid but *incorrect* value constitute value faults. Since the effect of these faults are dependent on the type of information dispersal, we have the sub-cases of *symmetric-value faults, ($\mathcal{S}$),* and *asymmetric-value faults, ($\mathcal{A}$),*. The asymmetric-value fault is equivalent to the case of a Byzantine fault and is also referred to as the *arbitrary fault* case.

The set of all tolerable faults, F, can be written as the union of three disjoint sets, giving $\mathcal{B} \cup \mathcal{S} \cup \mathcal{A}$, with the symbols corresponding respectively to benign, symmetric and asymmetric malicious faults. To accurately represent the single fault-type models, the set of benign faults, $\mathcal{B}$, can be split into two disjoint subsets: *benign symmetric faults* ($\mathcal{B}_\mathcal{S}$), and *benign asymmetric faults* ($\mathcal{B}_\mathcal{A}$).[4]

The CFEM reflects the behavior of the system in the presence of different node faults, when used in the system analysis phases, with each fault type mapped into one of the four disjoint classes. A *fault effect classification* captures the set of fault effects handled by the fault tolerance techniques implemented in the system. In the system design phase, the CFEM thus facilitates the selection of the fault tolerance methods to be used in the system. The *potential fault effect classes* are derived from alternative repartitionings of the total fault set, achieved by changing the set of fault detection and masking methods implemented in the system. The process of classifying a specific fault into a distinctive fault-effects category is termed as *fault transformation*, and it can enhance the system fault coverage by transforming initially intolerable faults into tolerable faults, or arbitrary faults into benign faults. The fault effect classification is then re-evaluated following the implementation of additional fault detection or masking techniques.

As mentioned earlier, the classification of non-catastrophic faults in the CFEM is a function of the fault tolerance techniques implemented in the system. If sufficient fault coverage and system reliability cannot be demonstrated by a given design, additional fault tolerance techniques can be used to increase the covered fault set or system reliability. For example, in a simplex system with no node or information redundancy, all faults are catastrophic. If information redundancy in the form of error detection and correction codes is used, then formerly catastrophic faults that cause errors which can be masked or detected by these coding procedures are transformed into benign faults.

[4]While crash faults are most often benign symmetric, with all non-faulty nodes able to detect that a node has crashed,range faults can be either symmetric or asymmetric. So, they are not included explicitly in this partitioning, but are restored when all benign faults are considered as a single class.

Similar transformations are also possible among the more severe fault types. For example, in a system of N nodes in which the hybrid symmetric scenario applies, suppose that the only benign faults which can be detected are checksum and missing messages. If a faulty node sends different values to different nodes, where each value is out of the range of correct values, the initial system may fail due to an uncovered asymmetric value fault, which is a catastrophic fault. However, if a range check is added to the fault detection techniques implemented in the node, that potentially uncovered catastrophic fault is converted to a benign fault. Thus, the covered fault set for any of the scenarios can often be extended by implementing additional fault detection methods which transform potentially catastrophic faults into benign ones. The addition of extra system nodes can also be used to transform some catastrophic faults into hybrid faults, in the case where the system fails due to resource exhaustion. The benefits of fault transformation are addressed in further detail in the next section.

3 CFEM: Implications and Comparisons With Classical Fixed Severity Fault Models

We demonstrate the benefits of the CFEM approach by examining possible solutions to a given design problem. We will present results for the general case, and then apply them to the following example. As discussed in the previous section, a system using only active redundancy techniques is capable of detecting benign faults from set $\mathcal{B}$.

However, if an (uncovered) malicious value fault from set $\mathcal{A} \cup \mathcal{S}$ occurs, system failure is likely to occur. When only non-iterative passive redundancy techniques, such as majority or fault-tolerant midpoint votes, are implemented, *symmetric faults* from the set $\mathcal{S} = \mathcal{B}_{\mathcal{S}} \cup \mathcal{S}$ are masked, but the occurrence of *asymmetric faults* from set $\mathcal{A} = \mathcal{B}_{\mathcal{A}} \cup \mathcal{A}$ can cause the system to fail. The use of interactive consistency and interactive convergence algorithms ensures all non-catastrophic fault types are covered, since such algorithms mask arbitrary faults, i.e. all faults in F.

We continue to assume a synchronous message passing system of N identical components or processes, called *nodes*, where the only evidence of a faulty node is an error in a message from that node. A good node is expected to collect information from other nodes and to arrive at a local decision that is consistent with the decisions of all other good nodes. A good node may also need to compute a local value within a prespecified range of the values of other good nodes. We address the issue of usage of the consistent local values computed by good nodes in the next section.

Based on the assumed system fault model, the fault handling techniques implemented in the system, and the resulting fault resiliency, we derive characteristics of system consensus operations. We discuss variations in the assumed fault models, coverage parameters, and the fault resiliency We assume a system of N nodes capable of sustaining f faults. The number of faults, f, can be written as $f = f_{\mathcal{A}} + f_{\mathcal{B}} + f_{\mathcal{S}}$ where $f_{\mathcal{A}}$ is the number of faults from $\mathcal{A}$, $f_{\mathcal{S}}$ is the number of faults from $\mathcal{S}$, and $f_{\mathcal{B}}$ is the number of faults from $\mathcal{B}$; *any of these parameters can be fixed at zero by the assumed system fault model.*

A system required to be Fail Op/Fail Op/Fail Safe, should remain operational after two non-coincident faults of any type, and should degrade to a predefined safe state following the third fault. Implicit in this specification is that the fourth fault, regardless of its scope, symmetry, or malice, leads to immediate system failure. While "perfect fault coverage" refers to coverage of non-catastrophic or hybrid faults, if a working (not failed) system contains too many faulty nodes, the next fault might cause system failure, no matter what effects it may have had in a system with fewer faulty nodes. So, the class of catastrophic faults also includes faults which cause system failure by exceeding the system fault resiliency. The hybrid fault model scenarios provide limits on the number and types of faults that can be tolerated by a given node set according to the fault tolerance techniques implemented in the

system. As mentioned earlier, if the available resources are not adequate to meet the system dependability requirements, additional fault tolerance techniques and resources can be used transform faults from one class of the CFEM to another, potentially improving the system reliability and fault coverage.

3.1 CFEM and Classical Fault Models: Fault Scenarios, Fault Coverage and Fault Resiliency Comparisons

Many system design approaches assume perfect coverage to a given fault set. Then, an algorithm is chosen which tolerates the worst case faults in that set. As we shall see, this results in either overly optimistic or pessimistic models when the perfect fault assumption is relaxed, as it must be when the system implementation is completed. Since anecdotal evidence suggests that faults in $\mathcal{B}$ are the most common, with faults in $\mathcal{S}$ less common than those in $\mathcal{B}$, and faults in $\mathcal{A}$ the least common of all, the fault assumptions made in a given system can be used to evaluate the impact of implementing the different CFEM fault scenarios presented below.

The key to the usage of the CFEM is to be able associate the proper CFEM supported algorithm with the assumed system or node fault set. For example, if the node set to be covered contains asymmetric faults, then a CFEM consensus or convergence algorithm should be implemented. The fault tolerance algorithms required by the CFEM along with the supporting voting functions are those developed in Sections 4 and 5. We do emphasize that the main difference between these scenarios and the single fault-type models is their treatment of benign faults. Unlike earlier models, the CFEM and its algorithms take advantage of a good node's ability to recognize missing or garbled messages, message tampering and other crash or range faulty behavior. The notation $CFEM_{\mathcal{X}}$ is used to indicate that the scenario assumes that the worst case faults are in set $\mathcal{X}$, where $\mathcal{X} \in \{\mathcal{B}, \mathcal{S}, \mathcal{A}\}$. The total number of faults which can be present in the system simultaneously is given by $f = f_{\mathcal{A}} + f_{\mathcal{S}} + f_{\mathcal{B}}$. The faults could have been sequential, near coincident, or coincident, but under static redundancy management, all the faulty nodes remain in the system. For the system to maintain correct operation, the number of nodes, N, and the number of faults f, must satisfy the conditions specified in the scenario below that corresponds to the system implementation. If the number of faults exceeds the limit of f, then the fault scenario in which f is exceeded is a catastrophic fault. Thus, the set of catastrophic faults includes those faults that cause system failure because they exceed the fault resiliency of the system.

We next present a brief overview of the classic single fault-type model scenarios, and the CFEM scenarios which supersede them. We present the resiliency of these scenarios in terms of the total number of faults, t, concurrently in the system. We do not address the various combinations of sequential and coincident node faults that could result in t faulty nodes. For simplicity, we present a limited version of the CFEM. The full CFEM scenarios are more flexible than those described below, because they also permit the minimum number of good nodes required for system operation to be specified. For example, as described below, both benign fault scenarios require $N \geq t + 1$ nodes to tolerate t benign faults. A more detailed treatment of the CFEM scenarios gives $N \geq t + \tau_{\mathcal{B}} + 1$, where $\tau_{\mathcal{B}}$ is fixed according to the minimum number of nodes permitted in an operating system. Comparable parameters $\tau_{\mathcal{S}}$ and $\tau_{\mathcal{A}}$ are defined for $CFEM_{\mathcal{S}}$ and $CFEM_{\mathcal{A}}$. However, the minimum values of these parameters are adopted in the remainder of this section. Having defined single fault-type and CFEM scenarios, we next compare the numbers and types of faults tolerated by each[5]. Since the classification of faults under the CFEM scenarios depends on the fault detection and masking methods implemented in the system, the potential for improved system reliability using fault transformation. Fault transformation can be done by adding detection methods, as well as by the addition of extra nodes. The minimum number of nodes required

[5]The combined results appear in Table 5

f_S	0	1	2	3	4	5
N_S	2	3,4	5,6	7,8	9, 10	11, 12

Table 1 Fault Resiliency Under the Symmetric Fault Scenario C_S

to maintain correct operation in the presence of a given number of faults is called the *resiliency* of the system. The fault set and detection algorithms corresponding to $CFEM_{\mathcal{B}}$ of the hybrid model are identical to those of the benign fault set $\mathcal{B}$; therefore, they share the same resiliency. However, both $CFEM_{\mathcal{S}}$ and $CFEM_{\mathcal{A}}$ differ in resiliency from $C_{\mathcal{S}}$ and $C_{\mathcal{A}}$, because their covered fault sets differ.

Classic Benign ($C_{\mathcal{B}}$) and CFEM Benign ($CFEM_{\mathcal{B}}$) Fault Scenarios

In these scenarios, the only fault-tolerance functions implemented are fault detection mechanisms. Thus, under the CFEM, the set of covered faults is $\mathcal{B}$, i.e., the set of faults which can be detected by each nodes. Any faults which can not be detected by those algorithms are, by definition, intolerable. Suppose all good nodes either detect an error in a message sent by a faulty node, or fail to receive an expected message. All nodes adopt the predefined default value as their local value for that node, and will thus agree on the fault status of the sender, without any further exchange of information. In this case, the fault is *benign*, because all good nodes can detect its occurrence. Thus, all local views are consistent with the global view. This is the behavior assumed in both the classic single fault-type scenario, $C_{\mathcal{B}}$, and the CFEM benign scenario, $CFEM_{\mathcal{B}}$. Both scenarios assume perfect coverage to all faults in $\mathcal{B}$, with $N \geq t + 1$ nodes required to detect t faults in $\mathcal{B}$. The only difference between these scenarios is in the default value adopted when a faulty node is detected. Under $CFEM_{\mathcal{B}}$, the adoption of a default value, $\mathcal{E}$, distinguishable from a correct value, when a message error is detected, allows that value to be ignored in future computations. The potentially correct or boundary default value adopted under $C_{\mathcal{B}}$ may skew future computations. Note that system failure may occur if a malicious value or a catastrophic fault occurs.

Under $CFEM_{\mathcal{B}}$, all faults in $\mathcal{B}$ are covered. CFEM based active redundancy algorithms and at least $N_{\mathcal{B}}$ nodes are required to tolerate $f_{\mathcal{B}}$ benign faults, where $N_{\mathcal{B}} = f_{\mathcal{B}} + (\tau_{\mathcal{B}} + 1)$. The parameter $\tau_{\mathcal{B}}$ is a fixed index, dependent upon the desired fault coverage, where $(1 + \tau_{\mathcal{B}})$ is the minimum number of nodes required for the system to remain operational.

Classic Symmetric ($C_{\mathcal{S}}$) and CFEM Symmetric ($CFEM_{\mathcal{S}}$) Scenarios

To handle symmetric malicious faults in $C_{\mathcal{S}}$, the established results appearing in literature necessitate $N_{\mathcal{S}} = 2f_{\mathcal{S}} + 1$ as the minimum number of nodes required to tolerate $f_{\mathcal{S}}$ symmetric faults. So, the resiliency of a system using this fault model (and an appropriate non-iterative passive redundancy algorithm) is given by $N_{\mathcal{S}}$, with all faults treated as if they were symmetric malicious faults. A system with three or four nodes can tolerate at most a single fault. A system with five or six nodes can tolerate at most two faults. Table 1 summarizes the resiliency for different values of $N_{\mathcal{S}}$.

For the CFEM, faults in $\mathcal{B} \bigcup \mathcal{S}$ are covered using hybrid non-iterative passive redundancy algorithms. At least $N_{\mathcal{S}} = (f_{\mathcal{B}} + f_{\mathcal{S}}) + (\tau_{\mathcal{S}} + 1)$ nodes are needed to tolerate $(f_{\mathcal{B}} + f_{\mathcal{S}})$ faults, where $\mathcal{S}_{\max} = \lfloor \frac{N_{\mathcal{S}} - 1}{2} \rfloor$ and $f_{\mathcal{S}} \leq \mathcal{S}_{\max}$. If operation in the presence of only one non-faulty node is possible, then $\tau_{\mathcal{S}} = \mathcal{S}_{\max}$. Otherwise, $\tau_{\mathcal{S}} \geq \mathcal{S}_{\max}$ if at least $(\tau_{\mathcal{S}} + 1)$ good nodes are required.

Under $CFEM_{\mathcal{S}}$, not all faults are assumed to be the worst case symmetric malicious faults; so, a system using this model and the appropriate CFEM fault-tolerant voting functions (Sec. 4) will tolerate more faults than the previous model. We have $N_{\mathcal{S}}$, is given by $N_{\mathcal{S}} = 2f_{\mathcal{S}} + f_{\mathcal{B}} + 1$. A set of three nodes

$f_{\mathcal{S}}$	0	1	2	3	4
$f_{\mathcal{B}}$					
0		3	5	7	9
1	2	4	6	8	10
2	3	5	7	9	11
3	4	6	8	10	12
4	5	7	9	11	13
5	6	8	10	12	14

Table 2 Fault Resiliency Under $CFEM_{\mathcal{S}}$.
Unlike Table 1, the combination of $f_{\mathcal{B}}$ and $f_{\mathcal{S}}$ tolerated by a given N (table entries) are indicated.

r	$f_{\mathcal{A}}$	$N_{\mathcal{A}}$
1	0	3
1	1	4,5,6
2	2	7,8,9
3	3	10,11,12

Table 3 Fault Resiliency Under Arbitrary Fault Model ($C_{\mathcal{A}}$)

can now tolerate either two benign faults or a single symmetric malicious fault. Four nodes can tolerate three benign faults or one symmetric malicious fault and one benign fault. The resiliency $N_{\mathcal{S}}$ is given in Table 2, with the entry corresponding to row $f_{\mathcal{B}}$ and column $f_{\mathcal{S}}$ giving the number of nodes ($N_{\mathcal{S}}$) needed to tolerate $f_{\mathcal{S}}$ symmetric malicious and $f_{\mathcal{B}}$ benign faults.

Classic Arbitrary ($C_{\mathcal{A}}$) and CFEM Arbitrary ($CFEM_{\mathcal{A}}$) Fault Scenarios

Under the assumption of all arbitrary faults in $\mathrm{C_A}$, we have $N_{\mathcal{A}} = 3f_{\mathcal{A}} + 1$ as the resiliency for $f_{\mathcal{A}}$ faults in $\mathcal{A}$. A minimum of four nodes is required to tolerate a single fault with a single rebroadcast round ($r = 1$). Seven nodes *and* an additional round of rebroadcast ($r = 2$) are required to tolerate two faults, and so on. Thus, there is no benefit under this model to adding an additional node above the minimum, because no more faults can be tolerated by adding only one node. In fact, the overall system reliability will decrease, because there are more nodes which can fail. Increasing r increases the number of messages which need to be exchanged exponentially. Table 3 depicts the number of faults tolerated by a given number of nodes for given values of r.

For the CFEM, the fault set is $\mathcal{B} \bigcup \mathcal{S} \bigcup \mathcal{A}$; so, all possible CFEM faults are covered. A minimum of $N_{\mathcal{A}} = (2f_{\mathcal{A}} + 2f_{\mathcal{S}} + f_{\mathcal{B}} + \tau_{\mathcal{A}} + 1)$ nodes is sufficient to tolerate $(f_{\mathcal{A}} + f_{\mathcal{B}} + f_{\mathcal{S}})$ faults. The maximum number of faults in $\mathcal{A}$ that can be tolerated is $\mathcal{A}_{\max} = \lfloor \frac{N_{\mathcal{A}}-1}{3} \rfloor$ with $f_{\mathcal{A}} \leq \mathcal{A}_{\max}$, $\tau_{\mathcal{A}} \geq \mathcal{A}_{\max}$, and at least $(\tau_{\mathcal{A}} + 1)$ good nodes assumed to be necessary for the system to remain operational. If a consensus algorithm with r rounds of rebroadcast is used, then the further restriction of $f_{\mathcal{A}} \leq r$ is also necessary. For interactive convergence, $\tau_{\mathcal{A}} = f_{\mathcal{A}}$, i.e., $\tau_{\mathcal{A}}$ relates to r correspondingly.

Under the CFEM for $CFEM_{\mathcal{A}}$, using a CFEM based interactive consistency algorithm such as HOM(r), we have $N_{\mathcal{A}} = 2f_{\mathcal{A}} + 2f_{\mathcal{S}} + f_{\mathcal{B}} + r + 1$, as demonstrated in Table 4, with $N_{\mathcal{A}}$ for different values of $f_{\mathcal{B}}$, $f_{\mathcal{S}}$ and $f_{\mathcal{A}}$ given by the corresponding table entry. Increasing the number of processors without increasing r thus permits more benign and symmetric malicious faults to be tolerated.

For interactive convergence algorithms, the resiliency under the usual arbitrary fault model is identical to that shown in Table 3. Under $CFEM_{\mathcal{A}}$ of the CFEM model, the resiliency for interactive convergence algorithms is that shown in Table 4 for $r = 1$. Tables 5 and 6 summarize the *composite* fault-set resilience

$r = 1$								
	$f_{\mathcal{A}} = 0$				$f_{\mathcal{A}} = 1$			
	$f_{\mathcal{S}} = 0$	$f_{\mathcal{S}} = 1$	$f_{\mathcal{S}} = 2$	$f_{\mathcal{S}} = 3$	$f_{\mathcal{S}} = 0$	$f_{\mathcal{S}} = 1$	$f_{\mathcal{S}} = 2$	$f_{\mathcal{S}} = 3$
$f_{\mathcal{B}} = 0$		4	6	8	4	6	8	10
$f_{\mathcal{B}} = 1$	3	5	7	9	5	7	9	11
$f_{\mathcal{B}} = 2$	4	6	8	10	6	8	10	12
$f_{\mathcal{B}} = 3$	5	7	9	11	7	9	11	13
$f_{\mathcal{B}} = 4$	6	8	10	12	8	10	12	14
$f_{\mathcal{B}} = 5$	7	9	11	13	9	11	13	15
$f_{\mathcal{B}} = 6$	8	10	12	14	10	12	14	16

Table 4 Fault Resiliency of a CFEM System Under $CFEM_{\mathcal{A}}$
For a given node size N (table entries), the X and Y co-ordinates combine to depict the combination of $f_{\mathcal{B}} + f_{\mathcal{S}} + f_{\mathcal{A}}$ tolerated under CFEM

Nodes: N	**2**	**3**	**4**	**5**	**6**	**7**	**8**
$C_{\mathcal{B}}$	1	2	3	4	5	6	7
$C_{\mathcal{S}}$		1	1	2	2	3	3
C_{A}			1	1	1	2	2
$CFEM_{\mathcal{S}}$		$(\leq 2, 0, 0)$ $(0, 1, 0)$	$(\leq 3, 0, 0)$ $(1, 1, 0)$ $(0, 1, 0)$	$(\leq 4, 0, 0)$ $(\leq 2, 1, 0)$ $(0, \leq 2, 0)$	$(\leq 5, 0, 0)$ $(\leq 3, 1, 0)$ $(1, 2, 0)$	$(\leq 6, 0, 0)$ $(\leq 4, 1, 0)$ $(2, 2, 0)$ $(1, 2, 0)$	$(\leq 7, 0, 0)$ $(\leq 5, 1, 0)$ $(\leq 3, 2, 0)$ $(1, 3, 0)$ $(0, \leq 3, 0)$
$CFEM_{\mathcal{A}}$			$(\leq 2, 0, 0)$ $(0, 0, 1)$ $(0, 1, 0)$	$(\leq 3, 0, 0)$ $(1, 1, 0)$ $(1, 0, 1)$ $(0, 1, 0)$ $(0, 0, 1)$	$(\leq 4, 0, 0)$ $(\leq 2, 1, 0)$ $(\leq 2, 0, 1)$ $(0, \leq 2, 0)$ $(0, 1, 1)$	$(\leq 4, 0, 0)$ $(\leq 2, 1, 0)$ $(\leq 2, 0, 1)$ $(0, \leq 2, 0)$ $(0, 1, 1)$	$(\leq 5, 0, 0)$ $(\leq 3, 1, 0)$ $(\leq 3, 0, 1)$ $(1, 2, 0)$ $(1, 1, 1)$ $(0, \leq 2, 0)$ $(0, 1, 1)$ $(0, 0, \leq 2)$

Table 5 Classical and CFEM Covered Faults (The fault tuples are represented as $(f_{\mathcal{B}}, f_{\mathcal{S}}, f_{\mathcal{A}})$)

offered by the CFEM for the various fault scenarios.

Unlike many existing fault models, the fault model scenarios presented in this chapter explicitly define the type and number of faults that can be tolerated by a system satisfying a specific scenario, as well as the class of algorithms needed to tolerate those faults. The main difference between the CFEM and other fault models is the inclusion of implementation information in designing a system's fault handling resources. A fault may be manifested benignly in one system implementation, while another system would view the fault as an asymmetric value fault. As we shall see subsequently, the increased precision of the mixed fault-type view over the single fault-type view helps provide a more realistic estimate of systems reliability.

4 Utilizing CFEM: (a) Fault Tolerant Voting Functions

So far, we have shown the flexibility of the CFEM in being able to handle sets of fault-effects of varying fault severity as compared to existing fixed fault-severity models. However, before we can actually utilize the CFEM facets in the system operations, we need to systematically develop voting functions

Scenario	Assumed Coverage	Redundancy	Caveat
$C_{\mathcal{B}}$	$\mathcal{B}$	$N \geq f_{\mathcal{B}} + 1$	$f \in (\mathcal{A} \cup \mathcal{S})$ not covered.
$C_{\mathcal{S}}$	$\mathcal{S} \cup \mathcal{B}_{\mathcal{S}}$	$N \geq 2f_{\mathcal{S}} + 1$	$f \in (\mathcal{B}_{\mathcal{A}} \cup \mathcal{A})$ not covered.
$C_{\mathcal{A}}$	$\mathcal{A} \cup \mathcal{S} \cup \mathcal{B}$	$N \geq 3f_{\mathcal{A}} + 1$	Multiple rounds of messages needed.
$CFEM_{\mathcal{B}}$	$\mathcal{B}$	$N \geq f_{\mathcal{B}} + 1$	$f \in (\mathcal{S} \cup \mathcal{A})$ not covered.
$CFEM_{\mathcal{S}}$	$\mathcal{B} \cup \mathcal{S}$	$N \geq f_{\mathcal{B}} + 2f_{\mathcal{S}} + 1$	$f \in \mathcal{A}$ not covered.
$CFEM_{\mathcal{A}}$	$\mathcal{A} \cup \mathcal{S} \cup \mathcal{B}$	$N \geq f_{\mathcal{B}} + 2f_{\mathcal{S}} + 3f_{\mathcal{A}} + 1$	Multiple rounds of messages needed.

Table 6 Attributes of N-Node Systems Under Classic and CFEM Assumptions

and algorithms which can support the CFEM. Fortunately, most of the existing techniques developed to mask or to detect faults in redundant resources or components can be directly modified to take advantage of CFEM. We focus on forward-recovery methods which are designed to ensure continual service in the presence of a limited number of faulty nodes. When combined with the appropriate system assumptions, each of the techniques described below can be the basis for some consistency or convergence operation.

Under the CFEM, each incoming message received by a node is examined for potential faults by some set of detection mechanisms. Such mechanisms include sanity checks, formatting checks, and error detection and/or correction codes. If no discrepancies are detected in a message, the message contents may be correct or a malicious fault-effect may be present, or an intolerable (uncovered) fault-effect may have occurred. At the local node level, no further fault type discrimination is possible.

If an error is detected, such as a framing, parity, or encoding fault, a missing message, or a range violation, then we adopt a default error or status value, $\mathcal{E}$, as the sending node's value. Under no circumstances can $\mathcal{E}$ be an acceptable value, and it may differ based on the data types of correct values or on the type of algorithm in which the information is to be used. Without loss of generality, we assume that the value $\mathcal{E}$ is greater than any permitted numerical data value.

Since each node performs local detection only, the adoption of $\mathcal{E}$ as a value means that a faulty node with *locally benign* effects was detected. The detected fault could be a benign fault, with all good nodes adopting $\mathcal{E}$ for that sender's message. Or, it could be an asymmetric malicious fault that sent detectably erroneous messages to some, but not all, good nodes. Since the number of $\mathcal{E}$ values adopted by two good nodes can legitimately differ, standard fault masking algorithms must be extended. If no nodes adopt the default error value, $\mathcal{E}$, then the redundancy algorithms revert to the classic single fault models.

4.1 Fault Handling Under the CFEM: Voting Functions

Voting functions which support CFEM have the generic structure of filtering specified numbers of error-status values, $\mathcal{E}$, to yield a consistent voted value. Thus, standard fault tolerant voting functions, such as the majority or median, are extended to accommodate Customizable Fault/Error by applying an exclusion function to the data value set prior to voting. We do mention that our intent here is to detail the CFEM variations of the voting functions; the application of these voting functions is discussed in the section of hybrid algorithms.

We first define the filtering function $exclude(V)$, which takes a set V of N elements, $\{v_1, v_2, \ldots, v_N\}$, removes any error values, $\mathcal{E}$, from V, and returns the set $(V - \mathcal{E})$, containing $(N - N_{\mathcal{E}})$ elements. $N_{\mathcal{E}}$ represents the number of discerned $\mathcal{E}$ values. In the absence of benign faults ($f_{\mathcal{B}} = 0$), no elements are

excluded from the vote. The subsequent CFEM voting functions are based on the *exclude()* function. Note that the functions in these sections are not sufficient to mask faults in $\mathcal{A}$ which require iterative algorithms such as the approximate agreement functions described later in Section 5. If an (uncovered) asymmetric malicious fault occurs, all good nodes might not compute consistent values, and system failure could result.

CFEM Majority Vote

A majority vote is typically used by each good node to compute a common final value for bimodal values received from other nodes or input sources. For $N_\diamond$ defined as $(N - N_\mathcal{E})$, the *CFEM_majority(V)* is given by:

$$majority(exclude(V)) = \begin{cases} v, & \text{if more than } \lfloor \frac{N_\diamond - 1}{2} \rfloor \text{ of the } v_i = v. \\ \mathcal{E}, & \text{otherwise.} \end{cases}$$

The default value, $\mathcal{E}$, returned when no majority exists, must be defined *a priori* and must be a potentially correct value, to avoid introducing a fault into a fault-free scenario. Since the *majority* function ignores $\lfloor \frac{N_\diamond - 1}{2} \rfloor$ elements, the composite function will be able to tolerate up to f faults, where $f = f_\mathcal{B} + \lfloor \frac{(N - N_\mathcal{E}) - 1}{2} \rfloor$.

CFEM Mean and Midpoint

The functions *mean* and *midpoint* are commonly used to average numerical data. The *mean* of n values $v_i \in V$, for $i \in \{1, \ldots, n\}$, is $mean(V) = \frac{1}{p} \sum_{i=1}^{n} v_i$.
The *midpoint* of n values $v_i \in V$, for $i \in \{1, \ldots, n\}$, is the mean of extrema, with
$midpoint(V) = \frac{1}{2}(\min_{i=1,n}(v_i) + \max_{i=1,n}(v_i))$, often called the *mean of medial extremes* or MME. Since these functions are sensitive to extremal values, fault-tolerant versions are defined using the *reduce* function, where, if V is a set of values to be voted on, t extremal values need to be removed [5], i.e.,

$$reduce(V, t) = \{V\} - \ \{\text{the } t \text{ largest and } t \text{ smallest } v_i\}.$$

The *CFEM fault_tolerant_mean* and *CFEM fault_tolerant_MME* functions apply the *mean* and *midpoint* functions to restricted subsets of values, where the restriction first removes the $\mathcal{E}$ values from detected benign faults, then eliminates the extrema from the remaining elements using the *reduce* function, as defined in [5]. The number of extrema eliminated now depends on $N_\diamond = N - N_\mathcal{E}$, the number of elements remaining after removing the $f_\mathcal{B}$ benign fault values $\mathcal{E}$.

$$CFEM\_fault\_tolerant\_mean(V) = mean(reduce(exclude(V), f(N_\diamond)))$$
$$CFEM\_fault\_tolerant\_MME(V) = midpoint(reduce(exclude(V), f(N_\diamond))),$$

with $f(N_\diamond) = \lfloor \frac{(N - N_\mathcal{E}) - 1}{2} \rfloor$. Each function tolerates a total of $f = f_\mathcal{B} + f(N_\diamond)$ faulty elements. However, since $n_\mathcal{E}$ varies with the particular fault-set instantiation[6], the value of the t assumed by the *reduce* functions is not fixed. Thus, the number of items to be reduced by the *reduce* function is a based on a run-time calculation of $N_\diamond$.

[6].i.e., # of faults $f_\mathcal{B}$ is not fixed

CFEM Median

The *CFEM_median* consists of the *median()* applied after the *exclude()* function. For a set V of m *ordered* values $\{v_1, v_2, \ldots, v_n\}$, where $v_q \leq v_{q+1}$,

$$CFEM\_median(V) = \quad median(exclude(V)) = \quad \frac{(v_i + v_j)}{2},$$

where $i = 1 + k$, $j = m_{\mathcal{E}} - k$, and $k = \lfloor \frac{N - N_{\mathcal{E}} - 1}{2} \rfloor$. Since $v_q \leq \mathcal{E}$ by definition, the excluded values $\mathcal{E}$ will be the $f_{\mathcal{B}}$ largest values. So, the elements remaining in $V_{\mathcal{E}}$ after application of the *exclude* function will be $\{v_1, v_2, \ldots, v_{N-N_{\mathcal{E}}}\}$.

5 Utilizing CFEM: Convergence and Consensus Functions

At this stage we have developed the CFEM fault scenarios and the associated primitives of fault tolerant voting functions that are supported under the CFEM. We now switch to developing the functions of distributed consensus and convergence that are essentially utilized in providing for dependable services in generic synchronous distributed systems. Our intent here is to demonstrate that the CFEM can directly provide for consensus and convergence operations which are flexible in terms of (a) covering combinations of fault types, and (b) are not restricted by the classical assumptions of each node possessing identical number of data elements. The latter property allows for these functions to provide additional flexibility of fault handling in real, multiple fault scenarios.

When the system is required to tolerate at least one arbitrary node fault, interactive versions of the previous fault masking algorithms are required. In this section CFEM versions of iterative algorithms needed to assure consensus or convergence under the assumption of arbitrary fault effects are derived. In these algorithms, each node has an initial value which it transmits to all other nodes. Each node adopts a final value based on the values of all other nodes.

To ensure dependability, fault-free nodes are expected to make decisions and compute values consistent with those of other good nodes, based on information received in the messages from other nodes. Intuitively, the only condition necessary for the system to operate correctly is for good nodes to make consistent decisions or to compute the same value (or values guaranteed to be arbitrarily close). The system fails when consistent decisions or computations across the system are no longer possible. That is, some type of consensus conditions, similar to those given in Table 7, must be satisfied to guarantee that good nodes will make consistent decisions or compute consistent values. While there are many variations possible in stating the conditions needed to achieve exact agreement or approximate agreement among distributed nodes [2, 5, 8, 10, 11, 13, 14, 16, 17, 19], the requirements for consensus (**CS**) and convergence (**CV**) given in Table 7 are sufficiently general for our purposes.

Critical system functions must employ algorithms that achieve specified agreement conditions in all good nodes in the presence of faulty resources. Fault detection and masking techniques developed for other fault assumptions are integrated with the CFEM in this section to enhance their resilience to faults. System reliability estimates are then based on the relative occurrence probabilities of different fault effects and their impact on the consensus operation. Note that the behavior of a faulty node is not constrained by this definition. Furthermore, if the sending node is faulty, it does not matter what decision is reached by the good nodes regarding the faulty node's value, as long as they all agree. Third, good nodes are not required to recognize which nodes are faulty.

CS: Consensus Conditions

EA (Exact Agreement): All good nodes will agree on the value received from the sending node.

EV (Validity): If the sending node is non-faulty, then the value used by the receiving node corresponds to the sending node's value.

CV: Convergence Conditions

AA (Approximate Agreement): All good nodes' final values will be within a predefined range of each other.

AV (Validity): The final value of any non-faulty node is in the range of the initial values of all other good nodes. value used by the receiving node corresponds to the sending nodes value.

Table 7 Consensus and Convergence Conditions

(Hybrid Oral Messages) HOM(r):

S1: The Transmitter sends its personal value, v, to all receivers.

S2: For each i, let v_i denote the value that Receiver i gets from the Transmitter.

If $r = 0$, and either no value or an obviously incorrect value (out of range, failed check sum, etc.) is received, Receiver i adopts $\mathcal{E}$.

Otherwise, Receiver i adopts v_i. The algorithm then terminates.

If $r > 0$, each Receiver adopts $R(\mathcal{E})$, if an obviously incorrect or no value is received, and $R(v_i)$ otherwise. Each receiver then acts as the Transmitter in Algorithm HOM($r-1$) sending its personal value to the other $N-2$ nodes.

S3: For each i and j, with $i \neq j$, let v_j denote the value Receiver i gets from sender j in **S2** of HOM($r-1$). If no message is received or v_j is obviously incorrect (If the value $R^k(\mathcal{E})$ is received, where $k \geq r-l$ in **S2** HOM($r-l$), then that too is recognized as an error, and $\mathcal{E}$ should be adopted). Receiver i adopts $\mathcal{E}$ for v_j; otherwise, v_j is used.

Since all Receivers act as senders in HOM($r-1$), each Receiver will have a vector containing (N-1) values at the end of HOM($r-1$). Receiver i adopts $v = \text{HOM\_maj}(v_1, v_2, \ldots v_{N-1})$ as the Transmitter's value.

Table 8 Algorithm HOM

Consensus under CFEM

The Hybrid Oral Messages (HOM) algorithm[7], presented in Table 8 is an extension the oral messages (OM) algorithm of Lamport, et.al. [8], which can be proven to satisfy the consensus conditions **EA** and **EV** from Table 2.2.7 when certain conditions regarding the number and types of faults are met. Algorithm HOM differs from OM algorithm, as it must deal with values corresponding to detected node faults.

Algorithm HOM(r) assumes a fixed number of rebroadcast rounds, r, with $f_{\mathcal{A}} \leq r$. While detected node faults will yield $\mathcal{E}$ values, a malicious fault can take on any value in W, where W is the set of potentially correct values, without violating a range check. The function HOM_maj(V), employed by HOM(r), computes a consistent value from set of V The values in V can be a combination of correct values from W, incorrect values from W, and values from the default error value set.

Within the algorithm, the value $\mathcal{E}$ is adopted when an obviously incorrect value or no value is received from some node, say i, by another node, k. A node participating in HOM(r) may then need

[7] Variants of this algorithm have appeared earlier in [10, 19]

to indicate to node j that it recognized a fault in the original sender. However, since $\mathcal{E}$ is defined such that no good node can send it as a correct value, the second receiving node will assume that node k is faulty if it receives $\mathcal{E}$ from it, even though the $\mathcal{E}$ value is due to node i begin faulty. To remove this ambiguity, the value $R(\mathcal{E})$ is sent when a node recognizes an error in a message it receives. When $r > 1$ and $a > 0$, $R^2(\mathcal{E}) \equiv R(R(\mathcal{E}))$ is adopted and sent when a node receives $R(E)$ from the transmitter, $R^3(\mathcal{E}) \equiv R(R(R(\mathcal{E})))$ is adopted and sent upon receipt of $R^2(\mathcal{E})$ from the transmitter, and so on. If the power k of $R^k(\mathcal{E})$ received by an node exceeds r, then the value is erroneous, and $\mathcal{E}$ should be substituted. The use of the $R()$ operator to enclose $\mathcal{E}$ also prevents information from good nodes from being ignored when the $\mathcal{E}$ values are excluded in HOM_maj below. Thus, the default error set is extended to include (r) distinct error values, $\{\mathcal{E}, R(\mathcal{E}), R^2(\mathcal{E}), \ldots R^r(\mathcal{E})\}$, where $R^0(\mathcal{E}) \equiv \mathcal{E}$, $R^2(\mathcal{E}) \equiv R(R(\mathcal{E}))$, etc. The operator $R()$ applied to a value, such as $\mathcal{E}$, indicates that the sender of $R(\mathcal{E})$ detected a locally benign fault in a message it received, and adopted $R(\mathcal{E})$ For any value $x \in W$, $R(x) = x$. The inverse of the R operator, R^{-1} is also defined, with $R^{-1}(R(\mathcal{E})) = \mathcal{E}$, $R^{-1}(R^k(\mathcal{E})) = R^{k-1}(\mathcal{E})$, and $R^{-1}(x) = x$ for $x \in W$. These additional values are needed to prevent a good node from being viewed as faulty for passing on a message from a faulty transmitting node.

The function HOM_*maj* is similar to the *CFEM_majority* function defined earlier, except that it recognizes elements from the extended error value set set $\{R(\mathcal{E}), \ldots R^r(\mathcal{E})\}$ as acceptable values. Given a set V of k values, $v_i, \ldots, v_k$, HOM_*maj*(V) is given by

$$\text{HOM\_}maj(V) = \begin{cases} \mathcal{E}, & \text{if all of the } v_i \text{ satisfy } v_i = \mathcal{E}. \\ R^{-1}(v_{\mathcal{E}}), & \text{if } v_{\mathcal{E}} = majority(exclude(V)) \text{ exists, otherwise} \\ v_0, & \text{where } v_0 \text{ is a functionally determined value.} \end{cases}$$

The provision which assumes $\mathcal{E}$ if all the v_i are $\mathcal{E}$ can't occur on a good node. It is included to provide a fail safe default value should that case occur on a partially faulty node. In [18], we prove that for any $r \geq 0$, any $f_{\mathcal{A}} \leq r$, any $f_{\mathcal{S}} \geq 0$, and any $f_{\mathcal{B}} \geq 0$, Algorithm HOM(r) satisfies **EA** and **EV** for $\geq 2f_{\mathcal{A}} + 2f_{\mathcal{S}} + f_{\mathcal{B}} + r + 1$.[8]

Convergence under CFEM

As discussed previously, interactive convergence has been achieved if conditions **AA** and **AV** in Table 7 are satisfied. In Table 9 we now describe the (non-terminating) synchronous[9] convergence algorithm applicable under CFEM.[10].

Before addressing the convergence properties of the CFEM convergence algorithm, we point out major differences between this algorithm and the corresponding algorithms in [5, 20]. Nodes p and q may receive different values from asymmetrically faulty nodes. Thus, they may identify different numbers of faults as being in $\mathcal{B}$, and the sizes of the sets U_p and U_q need not be identical. The values of f, $f_{\mathcal{B}}$, $f_{\mathcal{S}}$, and $f_{\mathcal{A}}$ are fixed globally for a given execution of the algorithm. However, the non-faulty nodes p and q are only required to compute consistent values in the presence of t faults, at most $\mathcal{A}_{\max}$ of which are in $\mathcal{A}$. They are not required to agree on the global diagnosis of different numbers and types of faults.

Also, while t is fixed in other algorithms, the number of faults tolerated by Algorithm CV varies with the numbers of faults of different types. Once the values of $n_{\max}$ and τ_A have been chosen for the scenario, all combinations of $f_{\mathcal{A}}$, $f_{\mathcal{S}}$, and $f_{\mathcal{B}}$ faults that satisfy the scenario $CFEM_{\mathcal{A}}$ assumptions of $n_{\max} \geq 2f_{\mathcal{A}} + 2f_{\mathcal{S}} + f_{\mathcal{B}} + \tau_A + 1$, with $f_{\mathcal{A}} \leq \mathcal{A}_{\max}$, must be accommodated.

[8] A similar, mechanically verified, proof of this algorithm appears in [10].

[9] An asynchronous algorithm appears in [2]

[10] The discussion of termination in [5] can then be applied under this framework, using the HOM algorithm.

Let $f = f_{\mathcal{A}} + f_{\mathcal{S}} + f_{\mathcal{B}}$ be the number of faulty nodes present during a round of algorithm execution, with $N \geq 2f_{\mathcal{A}} + 2f_{\mathcal{S}} + f_{\mathcal{B}} + \tau_A 1$, where $f_{\mathcal{A}} \leq \mathcal{A}_{\max}$, $f_{\mathcal{A}} \in \mathcal{A}$, $f_{\mathcal{S}} \in \mathcal{S}$, $f_{\mathcal{B}} \in \mathcal{B}$, and $\mathcal{A}_{\max} = \lfloor \frac{N-1}{3} \rfloor$. Let the function g be either $g_A = CFEM\_fault\_tolerant\_mean$ or $g_M = CFEM\_fault\_tolerant\_MME$. At each round, each non-faulty node p performs the following steps.

S1: Node p broadcasts its current value v_p to all nodes, including itself.

S2: Node p collects all values sent to it during that round into the extended multiset V_p. If p does not receive a feasible value v_q from each node q, or receives no value from node q, it adopts the value $\mathcal{E}$.

S3: Node p excludes all error values $\mathcal{E}$ from V_p, giving $U_p = exclude(V_p)$.

S4: Node p computes its new value, $v' = g(U_p)$.

Table 9 A CFEM Algorithm CV for Achieving Convergence

The following theorem states the convergence properties of Algorithm CV.

Theorem 1 *Let $n_{\max} \geq 2f_{\mathcal{A}} + 2f_{\mathcal{S}} + f_{\mathcal{B}} + \tau_{\mathcal{A}} + 1$, where $\mathcal{A}_{\max} = \lfloor \frac{n_{\max}-1}{3} \rfloor$, $f_{\mathcal{A}} \leq \mathcal{A}_{\max}$, and $\tau_A \geq \mathcal{A}_{\max}$. Let P be a synchronous approximation protocol in which each node executes Algorithm CV. Suppose that $T \subseteq S$ is a set of nodes, with $|T| \geq n_{\max} - t$, and $t = f_{\mathcal{A}} + f_{\mathcal{S}} + f_{\mathcal{B}}$ is the number of faulty nodes present during the execution of round k of Algorithm CV.*

Let C be a sequence of iterations or rounds of P, and let k be a round number. Let U be the multiset of values held by nodes in T immediately before round k in C, and let U' be the multiset of values held by nodes in T immediately after round k in C. Then,

1. *If $g = g_M$, then $\delta(U')$[11] $\leq \delta(U)/2$.*
2. *If $g = g_A$, then $\delta(U') \leq t\delta(U)/(n_{\max} - 2t)$.*
3. *If $g = g_A$ or $g = g_M$, $\rho(U')$[12] $\subseteq \rho(U)$.*

The complete proof of this theorem is provided in the Appendix section. Basically the theorem demonstrates that the range of values of non-faulty nodes decreases in each round by a factor dependent upon the function g employed in the CFEM convergence algorithm. Thus, the algorithm will eventually converge, proving **AA**, with **AV** applying at the end of each round. The termination properties of this algorithm, discussed in [5], remain unchanged, except that a CFEM interactive consistency algorithm needs to be employed to achieve consistent agreement on termination. Existing interactive convergence algorithms, such as those in Welch and Lynch [20] and MAFT [7], can also incorporate the CFEM fault taxonomy. A similar result has been derived for the asynchronous interactive convergence algorithm[2].

These interactive methods represent but a subset of the algorithms that need to be reexamined under the assumption of CFEM faults. The exclusion of error ($\mathcal{E}$) values prior to application of a value selection function is relatively straightforward for non-iterative passive redundancy techniques. However, the impact of exclusion upon interactive consistency and convergence algorithms is more subtle, as evidenced by the difficulties experienced in devising a correct CFEM algorithm that can achieve the consensus conditions. The ability of asymmetric malicious faults to appear as locally benign faults makes many of the interactive single-fault algorithms invalid because the exclusion of $\mathcal{E}$ values may result in different nodes having different numbers of values to vote on to achieve the final value.

[11] $\delta(U') = max(U') - min(U')$
[12] $\rho(U') = [min(U'), max(U')]$

6 Summary

In this chapter, we have presented the CFEM, in which faults are classified based on their effects upon the system and upon the fault handling techniques implemented in the system. Extending beyond the fixed fault severity models (time-domain and data-domain, s-a-X, Byzantine faults), the CFEM framework permits handling a continuum of fault types as groups of faults of varying fault manifestations under a single algorithmic paradigm.

The difficulties in classifying faults by attributes, independent of the system implementation, application and environment, demonstrated the need for a fault taxonomy that captured system-dependent effects. The fault effects taxonomy (CFEM), which partitions all faults into tolerable and intolerable faults, addresses the need for a different type of fault classification. Having provided the CFEM and algorithms, hybrid fault model scenarios were defined, to combine the covered hybrid fault classes with the algorithms needed to tolerate them. The potential for fault transformation by adding more detection mechanisms, modifying the specified fault scenario, or increasing the node redundancy was also explored.

An important consideration in achieving the full benefit of our hybrid CFEM fault theory to real systems is the lack of existing experience in identifying and tolerating mixed fault types and fault combinations. Many researchers have stated that benign faults are more probable than symmetric value faults, which are more probable than asymmetric or Byzantine value faults. While anecdotal evidence of the presence of arbitrarily malicious failures exists, there is still disagreement regarding how best to protect against them, if at all. As the statistics of probability of occurrence of various fault types are better documented, the utility of the CFEM increases correspondingly.

The CFEM approach discussed here is currently being applied to a new architecture solving a real world problem. The architecture for a ship control system is being developed using these concepts. The results have been very encouraging and have demonstrated effectiveness to date during the development stage. The full utility of the approach will continue to be explored as the project progresses. The ability to clearly formulate and test relationships, both of dependence and independence, has been very useful in verifying and validating aspects of the architecture. A current research goal is to continue to compile existing digital system experience and to develop new error extraction guidelines and fault analysis techniques. This includes assessment of the effectiveness of existing fault detection and error logging methods.

Overall we have shown that a more precise dependability model can be constructed, supported by on-line diagnosis algorithms under a generalized hybrid fault model. We believe the integration of hybrid CFEM fault theory into digital system design and validation will provide a greater understanding of fault effects and the risks associated with uncovered faults.

Acknowledgment: We extend our acknowledgment to M. M. Hugue for the valuable inputs and extensive discussions over this work.

References

[1] A. Avizienis and J.-C. Laprie, "Dependable computing: From concepts to design diversity," *Proceedings of the IEEE*, vol. 74, pp. 629–638, May 1986.

[2] M. H. Azadmanesh and R. Kieckhafer, "New hybrid fault models for asynchronous approximate agreement," *Trans. on Computers*, vol. 45, #4, pp. 439–449, April 1996.

[3] M. Barborak, M. Malek and A. Dahburra, "The Consensus problem in FT computing," *ACM Computing Surveys*, vol. 25, pp. 171–220, July 1993.

[4] F. Cristian, "Understanding fault-tolerant distributed systems," *Commn. of the ACM*, vol. 34, pp. 57–78, Feb. 1991.

[5] D. Dolev *et al.*, "Reaching approximate agreement in the presence of faults," in *Proc. Reliable Distributed Systems*, pp. 145–154, Oct. 1983.

[6] L. Gong and P. Lincoln, "Byzantine agreement and authentication: Observations and applications in tolerating hybrid and link faults," Proc. of DCCA-5, 1995.

[7] R. Kieckhafer *et al.*, "The MAFT architecture for distributed fault tolerance," *Trans. on Computers*, vol. C-37, pp. 398–405, April 1988.

[8] L. Lamport *et al.*, "The Byzantine generals problem," *ACM Trans. on Programming Languages and Systems*, vol. 4, pp. 382–401, July 1982.

[9] J. Laprie, *Dependability: Basic Concepts and Terminology*. Springer-Verlag, 1992.

[10] P. Lincoln and J. Rushby, "A formally verified algorithm for interactive consistency under a hybrid fault model," in *Proc. FTCS 23*, pp. 402–411, June 1993.

[11] N. Lynch *et al.*, "A simple and efficient Byzantine generals algorithm," in *Symp. on Reliability in Distributed Software and Database Systems*, pp. 46–52, July 1982.

[12] M. Malek, "A consensus-based framework for responsive computer system design," in *Proceedings, NATO Advanced Study Institute on Real-Time Systems*, Springer-Verlag, October 1992.

[13] F. Meyer and D. Pradhan, "Consensus with dual failure modes," *Trans. on Parallel and Distributed Systems*, vol. 2, pp. 214–222, April 1991.

[14] M. Pease *et al.*, "Reaching agreement in the presence of faults," *JACM*, vol. 27, pp. 228–234, April 1980.

[15] N. Suri, M. M. Hugue, and C. Walter, "Reliability modeling of large fault-tolerant systems," in *Proc. FTCS-22*, pp. 212–220, July 1992.

[16] P. Thambidurai and Y. Park, "Interactive consistency with multiple failure modes," in Proc. Reliable Distributed Systems, pp. 93–100, 1988.

[17] R. Turpin and B. Coan, "Extending binary Byzantine agreement to multivalued Byzantine agreement," *Info. Proc. Letters*, vol. 18, pp. 73–76, February 84.

[18] C. J. Walter, N. Suri and M. Hugue, "Continual on-line diagnosis of hybrid faults," *Proc. DCCA-4*, January 1993.

[19] C. J. Walter, P. Lincoln and N. Suri, "Formally verified on-line diagnosis," *Trans. on Software Engg.*, vol. 23, #11, pp. 684–721, Nov. 1997.

[20] J. Welch and N. Lynch, "A fault tolerant algorithm for fault tolerant clock synchronization," *Information and computation*, vol. 77, no. 1, pp. 1–36, 1988.

7 Appendix

The proof of Theorem 1 is detailed below. We first present the multiset terminology and subsequently develop the proof.

7.1 Terminology

Our notation and definitions are similar to those used in [5][13]. Let the finite multiset U of real numbers be a function $U : R \to N$, which is nonzero on at most finitely many $r \in R$. The function U assigns a finite multiplicity to each value $r \in R$. The *cardinality* of multiset U is given by $\sum_{r \in R} U(r)$, and denoted by $|U|$. A multiset is *empty* if its cardinality is zero. The difference of multisets, $V - U$, is a multiset W, defined by

$$W(r) = \begin{cases} V(r) - U(r), & \text{if } U(r) - V(r) \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

The intersection $U \cap V$ of multisets U and V is the multiset W defined by

$$W(r) = \min(U(r), V(r)).$$

The *minimum* of a non-empty multiset U, $\min(U)$ is defined by

$$\min(U) = \min\{r \in R | U(r) \neq 0\},$$

with the *maximum*, $\max(U)$, defined similarly. We denote the closed interval $[\min(U), \max(U)]$ by $\rho(U)$, and let $\delta(U)$ be the length of that interval, with

$$\delta(U) = \max(U) - \min(U).$$

The *mean* of the multiset U is defined by $mean(U) = \sum_{r \in R} r \cdot U(r)/|U|$. The *midpoint* of the multiset U is defined by $mid(U) = [\max(U) + \min(U)]/2$. If U is a nonempty multiset, we define the multiset $s(U)$, obtained by removing one occurrence of the smallest value in U, to be the multiset $W(r)$ defined by

$$W(r) = \begin{cases} U(r), & \text{if } r \neq \min(U) \\ U(r) - 1 & \text{otherwise.} \end{cases}$$

The multiset $l(U)$, in which one occurrence of the largest value in U is removed, is defined similarly. If we assume t is a fixed, non-negative integer, then if $|U| \geq 2t$, we can compute $reduce(U, t) = s^t(l^t(U))$. We next formally define the hybrid midpoint and hybrid mean functions in forms appropriate for this discussion. Let U be a finite multiset, and W be a finite multiset over a set $\mathcal{Q}$ disjoint from the reals ($R \cap \mathcal{Q} = \emptyset$), with $W(q) = 0$ everywhere in $\mathcal{Q}$ except at $v_{\mathcal{B}} \in \mathcal{Q}$.[14] The *extended multiset* V is then given by $V = U \bigcup W$. We define the fault-tolerant midpoint function g_M to be $g_M(U) = mid(reduce(U, t))$. Similarly, the fault mean or averaging function g_A is given by $g_A(U) = mean(reduce(U, t))$. Then, the *hybrid fault-tolerant midpoint* or *hybrid MME* is defined by

$$hf_M(V) \equiv g_M(exclude(V)) \equiv g_M(U).$$

Similarly, we define the *hybrid fault-tolerant mean* as

$$hf_A(V) \equiv g_A(exclude(V)) \equiv g_A(U).$$

[13] The notation and concepts used in [20] are also developed from [5]

[14] The default error value $v_{\mathcal{B}}$ should be treated as if it were not a real number, as it must be distinguishable from all potentially correct values.

7.2 Essential Lemmas

The proof of Theorem requires the following lemmas, adapted from [5], under the assumption that not all nodes vote on the same number of values. For completeness, all lemmas are stated, but proofs are provided only when they differ vastly from those in [5].
The first lemma shows that the number of elements common to two non-empty multisets is reduced by at most 1 when either the smallest or the largest element is removed from each.

Lemma 1 *Suppose that V and W are non-empty multisets. Then,*

1. $|V \cap W| - |s(V) \cap s(W)| \leq 1$.
2. $|V \cap W| - |l(V) \cap l(W)| \leq 1$.

The next lemma extends Lemma 1 to removing different numbers of extremal values from two multisets of potentially different sizes. We adopt the notation $t_v \equiv t(V)$, where $t_v \leq \min(\mathcal{A}_{\max}, \lfloor \frac{n_v - \tau_{\mathcal{A}} - 1}{2} \rfloor)$, the maximum number of faults not in $\mathcal{B}$ that can be tolerated by $n_v \equiv |V|$ nodes under our assumptions, as derived for the reliability results in [15].

Lemma 2 *Suppose V and W are two multisets with $|V| \geq 2t_v$ and $|W| \geq 2t_w$, where $t_v \leq t$ and $t_w \leq t$. Then,*

$$|V \cap W| - |reduce(V, t_v) \cap reduce(V, t_w)| \leq 2\max(t_v, t_w).$$

Proof: Applying Lemma 1, we have

$$|V \cap W| - |reduce(V, \min(t_v, t_w)) \cup reduce(W, \min(t_v, t_w))| \leq 2(\min(t_v, t_w).$$

Without loss of generality, assume $t_v \geq t_w$. Then, by Lemma 1, removal of the remaining $t_v - t_w$ smallest and largest values from V completes the result. □
The proofs in [5] apply to the next pair of lemmas, except that we use t_v to indicate that the lemmas do not assume the global t value of the algorithm. Instead, the lemmas are valid for the maximum number of faults in $\mathcal{S} \cup \mathcal{A}$ that the (local) multiset of size $|V|$ can tolerate under our assumptions.

Lemma 3 *Suppose that k is a nonnegative integer, and U and V are non-empty multisets with $|V - U| \leq kt_v$, and $|V| > 2kt_v$. Then,*

$$\rho(reduce^k(V, t_v)) \subseteq \rho(U).$$

Lemma 4 *Suppose that U and V are nonempty multisets such that $|U - V| \leq t_v$ and $|V| > 2t_v$. Then, $g_M(V) \in \rho(U)$.*

Lemma 5 illustrates the value of the midpoint function in approximation. While the proof is identical to that in [5], the implications of this lemma differ because the multisets M and N need not have the same cardinality.

Lemma 5 *Let U, M, and N be non-empty multisets with $|M \cap N| > 0$, $\rho(M) \subseteq \rho(U)$, and $\rho(N) \subseteq \rho(U)$. Then,*

$$|mid(M) - mid(N)| \leq \delta(U)/2.$$

The following lemma provides the main result for the midpoint function.

Lemma 6 *Let U, V, and W be non-empty multisets, with $|V - U| \leq t_v$, $|W - U| \leq t_w$, and $|V \cap W| > 2\max(t_v, t_w)$. Then,*

$$|g_M(V) - g_M(W)| \leq \delta(U)/2.$$

Proof: Let $M = reduce(V, t_v)$ and $N = reduce(W, t_w)$. By Lemma 2,

$$|M \cap N| \geq |V \cap W| - 2\max(t_v, t_w),$$

and, by hypothesis, $|M \cap N| > 0$. By Lemma 3, with $k = 1$, $\rho(M) \subseteq \rho(U)$ and $\rho(N) \subseteq \rho(U)$. Applying Lemma 5 yields the result. □

The next three lemmas provide results for the mean function comparable to those for the midpoint.

Lemma 7 *Let U and V be non-empty multisets such that $|V - U| \leq t$ and $|V| > 2t_v$. Then $g_A(V) \in \rho(U)$.*

Lemma 8 *Let U, M, and N be nonempty multisets, and m, n, and i be nonnegative integers such that $|M| = m$, $|N| = n$, $|M \cap N| \geq i$, $\rho(M) \subseteq \rho(U)$, and $\rho(N) \subseteq \rho(U)$. Then,*

$$|mean(N) - mean(M)| \leq \delta(U)\frac{\max(m, n) - i}{\max(m, n)} \tag{1}$$

Proof: The result holds when $m = n$ by Lemma 8 of [5]. It remains to prove for $m \neq n$.
Let $L = M \cap N$, $M' = M - L$, and $N' = N - L$. Then, we have

$$|mean(M) - mean(N)| = \left| \frac{\sum_{r \in R} rM}{m} - \frac{\sum_{r \in R} rN}{n} \right| \tag{2}$$

If equation (2) is zero, we're finished. Without loss of generality, assume that

$$\frac{\sum_{r \in R} rM}{m} - \frac{\sum_{r \in R} rN}{n} > 0 \tag{3}$$

and that $n > m$. Then we can rewrite the left hand side of expression (3) as

$$\frac{(n - m)\sum_{r \in R} rM}{nm} + \frac{\sum_{r \in R} rM}{n} - \frac{\sum_{r \in R} rN}{n} \tag{4}$$

Since $n - m > 0$, expression (4) is less than or equal to

$$\frac{(m - n)\max(M)}{n}\frac{\sum_{r \in R}|M|}{m} + \frac{\sum_{r \in R} rM' - \sum_{r \in R} rN'}{n}$$

which is at most

$$\frac{\max(M)(n - m)}{n} + \frac{\max(M)(m - |L|) - \min(N)(n - |L|)}{n} \tag{5}$$

¿From expression (5), we can derive

$$\frac{\max(M)(n - |L|) - \min(N)(n - |L|)}{n} \tag{6}$$

By hypothesis, we have expression (6) less than or equal to

$$\delta(U)\frac{n - i}{n} \tag{7}$$

Since we have $n > m$, expression (7) is equivalent to the desired result. The argument for $n < m$ is similar, and the proofs when

$$\frac{\sum_{r \in R} rM}{m} - \frac{\sum_{r \in R} rN}{n} < 0$$

are symmetric.□

Lemma 9 *Let U, V, and W be non-empty multisets, with $|V| = n_v$, $|W| = n_w$, $|V \cap W| \geq i + 2\max(t_v, t_w)$, $|V - U| \leq t_v$, and $|W - U| \leq t_w$. Then,*

$$|g_A(V) - g_A(W)| \leq \delta(U)\frac{\max(m,n) - i}{\max(m,n)} \tag{8}$$

Proof: Let $M = reduce(V, t_v)$, $m = n_v - 2t_v$, $N = reduce(W, t_w)$ and $n = n_w - 2t_w$. By Lemmas 2 and 3, the hypotheses of Lemma 8 are satisfied, yielding the result.□
Using these lemmas, we can now verify our main theorem.

7.3 Proof of Theorem 1

Let p and q be any two nodes in T. Let V_p be the initial multiset of values held by p after the exclusion of $n_{\max} - n_p$ default error values v_B, where $|V_p| = n_p$ and $t_p = \min(\mathcal{A}_{\max}, \lfloor \frac{n_p - \tau_{\mathcal{A}} - 1}{2} \rfloor)$. Similarly, V_q is initial multiset of values held by q after the exclusion of $n_{\max} - n_q$ values v_B, where $|V_q| = n_q$ and $t_q = \min(\mathcal{A}_{\max}, \lfloor \frac{n_q - \tau_{\mathcal{A}} - 1}{2} \rfloor)$. Since there are at most t faulty nodes, we have

$$|V_p - U| \leq t_p \leq t$$
$$|V_q - U| \leq t_q \leq t.$$

Furthermore, since V_p and V_q contain identical entries from non-faulty nodes, we have

$$|V_p \cap V_q| \geq n - t > 2t \geq 2\max(t_p, t_q).$$

1. Sets U, V_p and V_q satisfy the hypothesis of Lemma 6. Therefore,

   $$|g_M(V_p) - g_M(V_q)| \leq \delta(U)/2.$$

   Since p and q were arbitrary, we're done.

2. By definition $(\max_{q \in T} n_q) \leq n_{\max}$ and $(\max_{q \in T} t_q) \leq t$. Let $i = n_{\max} - 3t$ and $m = n_{\max} - 2t$. Then, U, V_p and V_q satisfy the hypotheses of Lemma 9, and we conclude that

   $$|g_A(V_p) - g_A(V_q)| \leq t\delta(U)/(n_{\max} - 2t). \tag{9}$$

   Since p and q were arbitrary, the result follows.

3. Multisets U and V_p satisfy the hypotheses of Lemma 4 and Lemma 7. Therefore, $g_M(V_p) \in \rho(U)$ and $g_A(V_p) \in \rho(U)$. Thus, p's value after round k is in $\rho(U)$. Since p was arbitrary, all elements of U' are in $\rho(U)$.

□

Theorem 2 *Algorithm CV achieves Approximate Agreement*

Proof: Theorem 1 shows that the range of values of non-faulty processes decreases in each round by a factor dependent upon the function g employed in Algorithm CV. Thus, the algorithm will eventually converge, proving Agreement. Part 3 of Theorem 1 applies at the end of each round, ensuring that the Validity condition is satisfied. □
As mentioned previously, the termination properties discussed in [5] remain unchanged.