Reliability Modeling of Large Fault-Tolerant Systems*

N. Suri†, M.M. Hugue, C.J. Walter
Allied–Signal Aerospace Company
9140 Old Annapolis Road, Columbia, MD-21045

Abstract

A cluster-based ultra-reliable architecture is presented, offering synchronization and system functionality comparable to that of fully connected systems, with reduced system overheads. Existing combinatorial and Markov models do not sufficiently model concurrently occurring faults in such large systems. A reliability model considering the distribution of concurrent faults across the system clusters is shown to increase the accuracy of reliability and system fault-tolerance estimates. The hybrid fault model, which classifies faults based on their behavior, further improves reliability estimates and enhances the fault handling capability of each cluster. Linear growth in cluster reliability with respect to cluster size is possible, as are refinements in the convergence and consistency algorithms for synchronization.

1 Introduction

Existing ultra-reliable system designs, such as SIFT [18] and FTMP [6], rely on fully connected inter-processor structures to maintain synchronization based operations across the system through all possible fault scenarios, including Byzantine faults. As the performance requirements of complex control systems require increased processing power, these systems need to be extended. The exponential growth of link complexity with system size makes the fully connected structure un-realizable. While large, sparsely connected, ultra-reliable systems have been designed [14, 13, 5] to improve performance, they have been unable to simultaneously reproduce the fault resiliency of fully connected architectures. Instead, they rely on trade-offs among the system characteristics of synchronization, fault tolerance, reconfiguration, and performability.

*Research supported in part by ONR Contract # N00014-91-C-0014

†e-mail : suri@batc.allied.com

The cluster-based design presented here uses a divide-and-conquer strategy similar to [14] to meet the objective of supporting fault tolerant real time control tasks at a cluster and system level, without the algorithmic overhead of a fully connected system. This approach alleviates the compromises made in other large ultra-reliable designs, without the node degree and link costs associated with full connectivity. Furthermore, the algorithms for all system operations are formally validatable.

Also of concern in the design of large, ultra-reliable systems is accurately assessing the system reliability against the design objectives. A large body of research exists for modeling cluster level reliability. However, we show that the combinatorial and Markov methods used to model large system models do not consider concurrent fault occurrence cases and result in imprecise reliability estimates. A further improvement in these estimates is made by classifying various fault types at the cluster level.

In Section 2, we discuss the design basis for the architecture. Section 3 describes the proposed cluster model and its characteristics of synchronization and fault tolerance. Section 4 details the reliability model for such systems, and introduces the hybrid fault model. This reliability modeling approach, based on realistic distinctions among fault effects can be applied to any ultra-reliable system architecture. We conclude with a summary of our contributions, and future research endeavors.

2 Background and Motivation

The complete interconnection structure of SIFT[18] and FTMP[6] provided direct support for consistency based distributed and fault tolerant operations. However with graph and algorithmic overheads associated with these designs, extending these to large system designs impacts the performance and operational aspects. The hexagonal mesh structure of HARTS [13] and cluster-based approach of FTPP [5] increased the

number of processors within the system, but either relaxed the overall system synchronization assumptions or added expensive, special-purpose hardware to support the required synchronization primitives. Before addressing cluster-based architectures, we examine the challenges in designing such systems.

The operational framework of large hard real time fault-tolerant control systems places stringent requirements on system functions. To meet the fault-tolerance and performance requirements, tasks must be performed redundantly on multiple processors, with groups of tasks associated with groups of processors. Provable, fault-tolerant synchronization must be provided across the entire system. Synchronization requires both a local time frame within each cluster and a global time base across the entire system. Individual task groups need locally synchronous frames for agreement functions. System functions, such as deadlines, require globally synchronous frames for decisions throughout task interactions.

Synchronization primitives and low inter-node message transit times, that are naturally provided by a fully connected structure, must be maintained with fewer links in a cluster model. Since performance is also an important design issue, closely related tasks needing physical processor proximity to minimize communication overheads can be accommodated by fully connecting the nodes within a cluster. The inter-cluster linkages support synchronization operations, with each node in a cluster capable of obtaining information about other system nodes and clusters. The ability to change task-to-processor or task-to-cluster associations and to reconfigure in the presence of faults requires dependable synchronization among both nodes and clusters. The graceful degradation of the system functions across all clusters must also be supported. The system reliability and its resiliency to Byzantine faults should be maximized, while minimizing the number of interconnection links in the system.

The system objectives demand that relatively tight synchronization at both the intra-cluster and inter-cluster levels be maintained in the presence of reduced interconnections. So, the reliability of a system then becomes a function of the reliability of the inter-cluster linkage. A symmetric interconnect design aids the computation of system reliability and facilitates algorithm design and support. Earlier designs restricted reliability considerations specifically to intra-cluster operations. Since the proposed design requires each node to participate in both cluster and system operations, the loss of a cluster, but not necessarily all of its nodes, permits the remaining operational nodes to continue participating in the system's metabolic operations. A direct consequence is higher reliability and performance than predicted for comparable designs.

3 The Cluster Model

The cluster architecture is a software-synchronized, frame-based synchronous system with low graph metric complexity and algorithmic overhead. As the relatively sparse connections among nodes and clusters distinguishes the cluster model from its predecessors, we first describe the architecture and its salient characteristics based on these interconnections. Then, we present the synchronization properties which give the cluster model the functionality comparable to completely connected systems.

3.1 The Cluster Architecture

Let $G = (V, E)$ represent the directed system graph. The $|V| = N$ nodes and the E edges (synchronization and communication links) are partitioned into n clusters, with the i^{th} cluster expressed as $C_i = (V_i, E_i)$, $|V_i| \equiv \nu_i$. The links within the cluster (E_i) are specified by the intra-cluster and inter-cluster communication structures. For ease of reference, the nodes in cluster C_i are numbered sequentially and denoted by V_i^α, $\forall$ integers $\alpha \in \{1, \ldots \nu_i\}$. Node V_i^α is connected to all other nodes within C_i via *intra-cluster links*. *Inter-cluster links* are assigned according to intra-cluster node numbers, with the inputs to a node in cluster C_i received from the node in cluster C_k, having the same node number (modulus the cardinality of V_k if $\nu_k \neq \nu_i$). That is, for each node $V_i^\alpha \in V_i$, V_i^α receives inputs from node $V_k^{(\alpha)_{mod_{\nu_k}}}$ of cluster C_k, $\forall$ $k \in \{1, \ldots, n\}$, $k \neq i$. In cluster C_i, the node degree is given by $\nu_i + n - 1$. Using this connectivity structure, a node has direct access to all other nodes in the same cluster and a two hop access to any other system node.

While other graphs exist with similar link and degree reductions [14], the cluster model interconnect structure offers three significant benefits detailed in [?] and summarized below. First, a unique variation of convergence-based clock synchronization mechanisms is supported. Second, startup synchronization at both cluster and system levels is feasible and formally validatable in the presence of Byzantine failures. Such synchronization is usually assumed by other structures, with the techniques for achieving it left unspecified. And third, consistency is possible at both the

N	n	ν	E_{FC}	E_C	%E	D_{FC}	D_C	%D
24	6	4	552	216	60	24	9	62
40	8	5	1560	480	69	40	12	70
64	8	8	4032	960	76	64	15	76
100	10	10	9900	1900	80	100	19	81

Table 1: Link and Deg. Reduction for Diff. Node Sets

n_a	$\|V_a\|$	n_b	$\|V_b\|$	n_c	$\|V_c\|$	D/f.t	E	%E
1	100	0	0	0	0	99/33	9900	00.00
10	10	0	0	0	0	18/6	1900	80.80
5	10	10	5	0	0	18/6	2000	79.80
6	5	10	7	0	0	19/6	2140	78.30
6	5	8	6	3	7	21/7	2169	78.00
8	5	10	6	0	0	21/7	2260	77.17
17	6	0	0	0	0	22/7	2244	77.33
20	5	0	0	0	0	23/7	2400	75.75
25	4	0	0	0	0	27/9	2800	71.70

Table 2: E and D for Diff. Cluster Sizes, N=100

discrete cluster level and at a combined system level. Table 1 quantifies the link(E) and degree(D) reductions for the system, comparing the cluster model(C) to fully connected(FC) structures of similar sizes. In Table 2, the fault tolerance, node degree and link complexity are compared for three clusters with node sets V_a, V_b, and V_c of different sizes for a fixed N.

3.2 Synchronization Primitives

Convergence and consistency operations are performed using two layered protocols. Each node participates in both the cluster level and inter-cluster level convergence and consistency functions. The architecture supports a variety of synchronization primitives including those described in [3], [4], [12], [15], [11]. We assume algorithm variants of those used in the MAFT[9].

Virtually all existing convergence algorithms are based on fully connected structures, in which each non-faulty participant derives the same set of clock values for all other nodes. The achievable synchronization skew for an N node system is δ. The communication cost is $O(N^2)$ and each node receives N values.

For the cluster model described above, different clusters may contain different numbers of nodes. Nodes within the same cluster receive different sets of input signals as they are connected to different inter-cluster nodes. However, interactive convergence can still be achieved. This is a unique variant of the interactive convergence problem, based on defining the minimum overlapping information required in a non-fully connected system to achieve a convergence property. The resulting synchronization skew is 3δ and the cost is $O(N(\nu_i+n-1))$. We point out that in [14], the convergence-related cluster model relies on a guarantee that all nodes in a cluster receive identical sets of input signals and is altogether a different convergence scenario. Additionally, the design of [14] is a convergence model and can be shown to be inadequate to support inter-cluster consistency, which is a design objective for the proposed cluster model.

Consider all clusters C_i, for $i \in \{1, \ldots, w\}$, $w \leq n$, that exceed the minimum Byzantine resilient size of $\nu_i = 3f_i+1$, where f_i is the number of node faults that cluster C_i can tolerate. During startup, the system tolerates $\sum_{i=1}^{w} f_i$ node failures. A two-round inter-cluster startup algorithm can directly sustain $w+1$ faults, assuming a minimal one fault per cluster coverage. Furthermore, the system can sustain $\lfloor \frac{w-1}{3} \rfloor$ cluster faults in the presence of w Byzantine resilient clusters. Then, synchronization skew to within 2δ is achievable between any two nodes of the system for up to 2 faults and to 3δ for up to $\lfloor \frac{\nu_i+w-2}{3} \rfloor$ Byzantine faults.

However, since an overall objective was high performance, the cost of achieving this skew and supporting the system consistency operations must also be assessed. The divide-and-conquer strategy of the architecture provides considerable algorithmic support, as illustrated by comparing the time complexities of executing synchronization algorithms on different structures, shown in Table 3.

3.3 Fault-Tolerance Limits

Since each node, V_i^{α}, in the system has ν_i intra-cluster links and $(n-1)$ inter-cluster links, $\lfloor \frac{\nu_i+n-2}{3} \rfloor$ faults can be tolerated solely on the basis of the system convergence algorithms. For simplicity, we assume that all clusters are of the same size, with $\nu_i \equiv \nu$, and each cluster can tolerate up to $\lfloor \frac{\nu-1}{3} \rfloor$ Byzantine faults. Next, suppose that an optimistic fault distribution can be assumed, such that of m concurrently occurring faults across n cluster, at most $\lfloor \frac{\nu-1}{3} \rfloor$ faults occur within each cluster. If a cluster is able to detect and subsequently isolate a faulty node through its *local consistency* operations, the node fault can be masked, and a spare module can be integrated into the system. The specified system fault tolerance is thus maintained without requiring a global decision to reconfigure the faulty node out of the system.

Thus, the cluster model supports minimally $\lfloor \frac{\nu+n-2}{3} \rfloor$ faults and maximally $2\lfloor \frac{\nu+n-2}{3} \rfloor$ faults, assuming a uniform fault distribution, where each cluster remains below its individual fault-limit and has a

System	N	CNV Ohd. (Msgs/round)	IAC Ohd.(msg/node)	δ(m=0)	δ(m=1)
Full	128	$O(N^2)$	$O(N^{m+1})$	25 μs	25 μs
HARTS	128	$O(N(N-1)(2m+1)\log\ N)$	$O((N(2m+1)\log\ N)^{m+1})$	630 μs	1170 μs
Cluster	128	$O(N(\nu+n-1))$	$O(\nu^{\lfloor\frac{\nu-1}{3}\rfloor+1}+n^{\lfloor\frac{n-1}{3}\rfloor+1})$	75 μs	75 μs

Table 3: Comparing Full, Hypercube and Cluster System Complexity

single spare node per cluster.

Consider a discrete cluster fault with more than its local limit of $\lfloor\frac{\nu-1}{3}\rfloor$ node faults. Even though the cluster is faulty, the system wide consistency operations can still detect and isolate the faulty cluster when the number of faulty clusters is less than $\lfloor\frac{n-1}{3}\rfloor$. Note that we are still considering the faults as existing concurrently. Although the system fault-tolerance limit is dynamic, based on the fault distributions across the entire system, we can bound this limit.

Suppose that a single cluster is faulty and has been detected and isolated. The fault-tolerance of the system is then reduced to $\lfloor\frac{\nu+n-2-1}{3}\rfloor$. Both a node fault and a full cluster fault have identical impact upon the fault tolerance limits, even though the system's sensitivity to future faults differs.

If we now assume at that at most $m = \lfloor\frac{\nu+n-2}{3}\rfloor$ faults can be tolerated by the convergence limits, the fault distribution that causes the most cluster failures affects a maximum of $\frac{m}{\lfloor\frac{\nu-1}{3}\rfloor+1}$ clusters. For inter-cluster consistency operations, it is necessary to satisfy the condition :

$$\lfloor\frac{n-1}{3}\rfloor \geq \frac{m}{\lfloor\frac{\nu-1}{3}\rfloor+1}.$$

Similarly, the minimal fault-tolerance limit is

$$\left\lfloor \frac{n+\nu-2-\frac{m}{\lfloor\frac{\nu-1}{3}\rfloor+1}}{3} \right\rfloor.$$

This limit neglects spares and assumes a fixed fault distribution and the execution of cluster level consistency operations for fault detection and isolation.

The discrete handling of faults within clusters also reduces the software overhead for this function. The message complexity for a fully connected N-node system to mask m faults is $O(N^{m+1})$.

In a system with n clusters of size ν, each cluster runs its fault detection procedures concurrently, resulting in the overall complexity of $O(\nu^{\lfloor\frac{\nu-1}{3}\rfloor+1})$, where ν and $\lfloor\frac{\nu-1}{3}\rfloor$ are several orders of magnitude smaller than $N = n \times \nu$ and $m = \lfloor\frac{N-1}{3}\rfloor$.

It is unrealistic to assume that an N node system achieves its maximum fault-tolerance limit. For example, a 100 node system with 25 clusters and 4 nodes per cluster can tolerate up to 9 Byzantine faults. A fully connected system could cover 33 Byzantine faults, but this is a highly improbable and unrealistic fault coverage scenario. There is also the issue of latency of fault coverage. The $m+1$ rounds of message exchange required to handle m faults in such a system, with $m = 33$ in this case, also implies a $m+1$ round time latency in completing the Byzantine agreement algorithm. In a cluster model with an appropriate fault distribution, m faults could exist concurrently, but still appear as sequential faults to most clusters. Not only are these faults handled concurrently and with low algorithmic latency, the corresponding cost(complexity) of performing the coverage is also low. This establishes the architecture basis for the reliability modelling.

4 Cluster Model Reliability

We now show how the system reliability can be obtained as a composite function of cluster reliability and propose refinements to such a model.

4.1 Conventional Reliability Modeling

A conventional model obtains the system reliability in two phases. First, the cluster reliability, R_c, is obtained as a function of the number of node faults in a cluster, under the assumption of identical reliability among the clusters. Subsequently, the system reliability, R_{sys}, for sustaining up to x faulty clusters is derived as a combinatorial expression based on R_c values, i.e.,

$$R_{sys} = \sum_{i=0}^{x} \binom{n}{i} R_c^{n-i}(1-R_c)^i \qquad (1)$$

Alternatively, a Markov(or semi-Markov) model can be developed for the same case, showing each system state in detail. Since most of the system states are taken to be equivalent, they can be collapsed and the same combinatorial expression results. Using this expression, system reliability is simply a function of the number of cluster faults in the system and is insensitive to the distribution of node faults among the clusters.

The cases where i clusters are faulty through x node faults or where i cluster faults occur through y, $y \neq x$ node faults are indistinguishable for Eq. 1. The model ignores the variation of system states within different clusters. None of these distinct fault states is segregated in the combinatorial expression. Hence, the system reliability expression is insensitive to fault distributions. Further, concurrency of faults in different clusters is also neglected.

By definition, a Markov chain represents a sequence of events. Thus, transitions are sequential, and the usual approximation to modeling concurrent events is as near-coincident bifurcations. In the Byzantine fault model, the system is designed to tolerate m *concurrent faults*. There are no precise methods of handling this aspect in a Markov model. Usually a cascaded fault model[1] considers changes in system states through the occurrence of a single fault at a time, resulting in inherent inaccuracies in the predicted reliability.

4.2 Cluster Reliability

As discussed in Section 3.3, analysis of the cluster model reliability requires a different approach. For simplicity, consider an $N = n$ x ν node system tolerating up to m concurrent Byzantine faults. Each cluster is at least minimally Byzantine resilient, and can maintain individual consistency-based fault-detection, masking and recovery procedures. The occurrence of m concurrent faults in the full system, where the concurrency is with respect to a global time frame, distributes the faults over the various clusters. If f_i nodes fail in cluster V_i, for $\nu > 3f_i$, then the cluster can detect, isolate and recover from the faults. If, instead $\nu \leq 3f_i$, then cluster V_i fails. This simple cluster failure can be modeled using coincident faults without losing any accuracy for a single cluster. However, the m faults occurring concurrently in time are reduced to "sequential" faults on a spatial basis across the clusters due to localized fault handling in each cluster.

A new issue of fault distribution in the reliability model is thus spawned. In Eq. 1, the binomial coefficient, $\binom{n}{i}$, is derived by considering a single cluster to be faulty, followed by the recovery or failure of the cluster, with each cluster considered one at a time. This "sequential" spatial fault distribution, requires an enumeration of all possible cases of spreading the m faults across the clusters, with different clusters receiving differing number of faults. The concurrent fault handling model and the sequential model discussed below both address this scenario. We assume the use of discrete time Markov models though continuous time models are equally applicable.

4.2.1 Concurrent Fault Model

Most existing models consider concurrently occurring faults at both the cluster and inter-cluster levels as either nearly simultaneous faults or nearly coincident faults, invariably modeling them as a transition path to a failed system state. At most these model concurrency within a cluster and never at the system level.

Realistically, in a large system there is a non-zero probability of faults occurring concurrently, that increases with the number of nodes in the system. Hence, we must compute the probability of a system being operational given that exactly m concurrent faulty nodes exist in the system. System failure is defined in terms of exceeding either the consistency limit on faulty clusters, $f_{CONS} = \lfloor \frac{n-1}{3} \rfloor$, or the convergence limit on faulty nodes $f_{CONV} = \lfloor \frac{\nu+n-2}{3} \rfloor$.

We consider such a concurrent fault case in the cluster model. The number of ways to spread m faults among n clusters, allowing multiple faults in a cluster, is $\binom{n+m-1}{m}$[10]. However, as we demonstrate in the example below, this is an upper bound, because the number of faults which can occur in a single cluster is limited by the number of nodes in the cluster. Then, we must determine the number of clusters failing for each fault distribution.

Example 1 *Consider a system with $N = 80$, $n = 20$, $\nu = 4$ with fault tolerance(f.t) = 7. Table 4 enumerates the various fault distribution cases when at most 5 faults occur. The tuple $(x, y, \ldots)$ describes a fault distribution with x faults in one cluster, y in another and so on. Each cluster is capable of sustaining a single Byzantine fault. The column* Perm *refers to the number of ways a fault pattern can be spread across the clusters. P refers to the probability of occurrence of each distribution. Each fault distribution can occur across clusters in a variety of ways,* e.g. *E6 can occur in $\frac{n!}{1!2!(n-3)!}$ ways. Scenario E5 shows that a number of fault distributions are not possible for the defined structures. The distributions such as $(5, 0, 0, 0, \ldots)$ cannot exist for clusters of size 4, and these are marked with an "X."* □

Thus, we see that the binomial coefficient, $\binom{n}{i}$, no longer applies. Instead, we must determine the total number of ways that a cluster can be faulty over the *full* set of faults that the system can handle, for all $m \in \{1, \ldots, m_{max}\}$. Each of these fault distributions then needs to be evaluated for all of these fault ranges.

The discrete mathematics problem presented by the cluster model is non-trivial. The first step is to generate all possible fault distribution vectors (FDV's).

Ref	# N Flts	Fault Distn	# C Flts	P	Perm n=20	Perm n=10
A1	1	(1,0,0,0,0,...)	0	1	20	10
B1	2	(1,1,0,0,0,...)	0	1/2	190	45
B2	2	(2,0,0,0,0,...)	1	1/2	20	10
C1	3	(1,1,1,0,0,...)	0	1/3	1140	120
C2	3	(1,2,0,0,0,...)	1	1/3	380	90
C3	3	(3,0,0,0,0,...)	1	1/3	20	10
D1	4	(1,1,1,1,0,...)	0	1/5	4845	210
D2	4	(1,1,2,0,0,...)	1	1/5	3420	360
D3	4	(1,3,0,0,0,...)	1	1/5	380	90
D4	4	(4,0,0,0,0,...)	1	1/5	20	10
D5	4	(2,2,0,0,0,...)	2	1/5	190	45
E1	5	(1,1,1,1,1,...)	0	1/6	15504	X
E2	5	(1,1,1,2,0,...)	1	1/6	19380	X
E3	5	(1,1,3,0,0,...)	1	1/6	3420	X
E4	5	(1,4,0,0,0,...)	1	1/6	380	X
E5	5	(5,0,0,0,0,...)	X	X	X	X
E6	5	(1,2,2,0,0,...)	2	1/6	3420	X
E7	5	(2,3,0,0,0,...)	2	1/6	380	X

Table 4: Fault Distribution across Clusters

For a specific m, this is equivalent to enumerating all ways of representing the number m as a sum of non-negative integers : FDV's. No closed form expression exists for this quantity.

Once the FDV's have been computed, the faults in that vector must be distributed among the different clusters and their nodes. Table 4($col.\ P$) shows that, for a given m, each case has an equal probability of occurrence. However, since the total number of distributions is not known *a priori*, this probability is obtainable only after enumerating for a given m. Also, when clusters are chosen to be of different sizes, with different fault tolerance limits, enumerating the distributions become even more difficult. This computation quickly becomes intractable.

Figure 1 illustrates the Markov model for Table 4($n = 10$) with 4 concurrent faults. For each cluster, the faults within a cluster may still need to be treated as near-coincident faults, admitting the possibility of inadequate coverage. Examples of such behavior appear in Table 4 for cases B2, C2, C3 etc.

An alternate way of modeling system reliability, R_{sys}, is to treat it as function of the number of faults instead of as a function of time. A conventional fully connected system capable of sustaining f sequential faults has a decreasing reliability value till a f_{max} limit is reached, and falling to zero the moment the fault limit is exceeded. In the concurrent fault case, the reliability drops to zero for $f > 1$.

In contrast, when using a concurrent fault model, no change in the reliability occurs until f_1 faults have

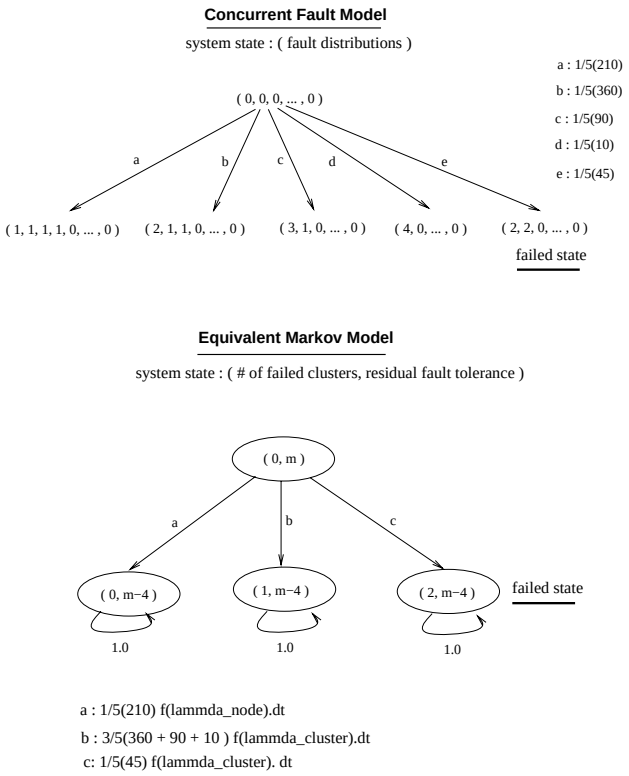

Figure 1: Concurrent Fault Markov Model

occurred, because all clusters remain fully functional. After f_1 faults have occurred, some of the fault distributions can cause multiple cluster failures leading to a system failure. However, there is also a finite probability of occurrence of distributions for which more than f_1 faults do not impair the system, resulting in a staggered step function. It is interesting to note that for certain values of m concurrent faults, no FDV's are obtainable which cause the system fault tolerance limits to be exceeded and causing system failure.

4.2.2 Sequential Fault Model

This fault model is a refinement of existing Markov models, assuming sequential fault occurrences and transitions, with fault detection possible in various states. Figure 2 illustrates a model for a 40 node system of 10 clusters. We assume that two cluster faults cause system failure, and that at most four node faults can occur. The permutations and fault distributions are as described in Table 4 for $n = 10$, although they are now obtained recursively from the preceding states. Each row of the model illustrates the distributions for a given m. The states in each row are disjoint, precluding the use of the standard technique of

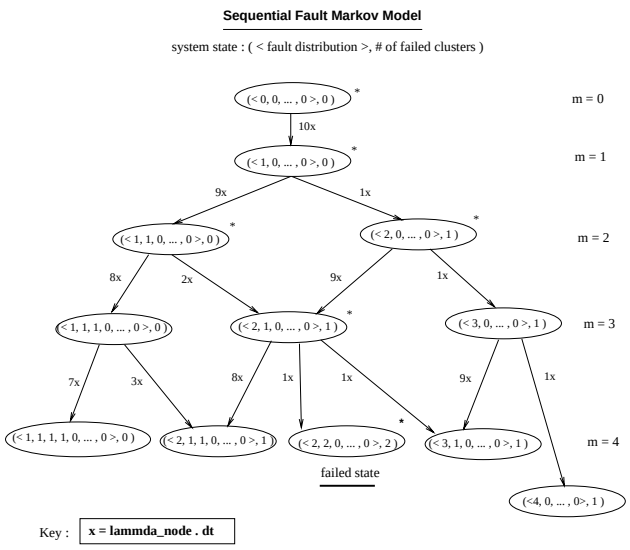

Figure 2: Sequential Fault Markov Model

collapsing states into equivalence classes. The starred ($\star$) states in the path from $(0,\ldots,0)$ to the failure state (2,2,0,...,0) indicates one set of states considered in conventional modeling.

All other operational state distributions are aggregated in the conventional model, even though they are discrete states and have non-zero occurrence probabilities. The R_c term of Eq. 1 lumps all operational states into one and is only indicative of the probability of being in one or more operational states. There is no differentiation among the various combinations of operational states possible with the fault distributions. The imprecision in conventional reliability calculations are caused by disregarding these states and their associated transitions. Table 5 illustrates differences in the reliability values obtained for the cluster model using the conventional(C) and the proposed schemes(C[dist]). The values for FTPP and for standard clusters assume reconfiguration at the cluster and system level. The cluster model(C[dist]) values are computed assuming no reconfiguration.

Based on this analysis, we make the following statement, to be formally proven in future work.

Statement : The system reliability of a distributed system with local (intra-cluster) and global (inter-cluster) fault containment regions is a function of both the number of node and cluster faults, and the spatial distribution of those faults across the system.

In a future model, we will combine the two models to allow the occurrence of concurrent faults from any state in the sequential model to the other states.

System[a] *(N)*	P (10hrs)	P (100hrs)	P (200hrs)
FTPP(80)	1.9×10^{-8}	1.957×10^{-7}	3.918×10^{-7}
C(40)	1.49×10^{-11}	1.09×10^{-8}	8.639×10^{-8}
C(40)[dist]	1.94×10^{-13}	6.76×10^{-11}	4.97×10^{-10}

[a] $\lambda_{proc} = 1\text{x}10^{-4}$/hr, coverages = unity

Table 5: Reliability variations for Fully Connected, Basic Cluster and with fault distributions

4.3 The Hybrid Fault Model

We can also improve the reliability estimates by relaxing the assumption that all faults are Byzantine. The *hybrid fault taxonomy*, based on our work in [16], [17], and on the following definitions, classifies faults according to the errors they cause and the techniques needed to tolerate those errors.

The *scope* of a fault is the portion affected by that fault. A *symmetric fault* generates errors that are seen identically throughout the fault scope. Otherwise, the fault is *asymmetric*. *Active or dynamic redundancy* techniques achieve fault-tolerance by fault-detection alone, or in conjunction with location and recovery. *Passive or static redundancy* techniques mask fault effects, thus avoiding *errors*.[1] *Non-iterative* passive redundancy techniques require a single round of message exchange while *iterative* techniques, such as interactive convergence and interactive consistency [3],[2], require multiple iterations of message exchange among participants. Fault-tolerant voting techniques, such as majority and median, are non-iterative passive redundancy techniques on which iterative passive redundancy techniques are often based. *Non-malicious* faults can and will be detected[2] in a non-faulty node by the active redundancy techniques implemented in that node, while *malicious* faults must be masked using passive redundancy techniques. Hybrid techniques combine active and passive redundancy techniques to enhance fault fault tolerance.

Combining the attributes of malice and symmetry produces the three mutually exclusive and collectively exhaustive fault sets of the *hybrid taxonomy*: *non-malicious faults* ($\mathcal{B}$), *malicious symmetric faults* ($\mathcal{S}$), and *malicious asymmetric faults* ($\mathcal{A}$). The *worst-case*, or most severe, faults in are those in $\mathcal{A}$, corresponding to the classic Byzantine fault where a faulty node supplies two different, yet feasible, values to different nodes. Faults in $\mathcal{S}$ are less severe than the faults in $\mathcal{A}$, but are more severe than faults in $\mathcal{B}$. Faults in

[1] For further details, see [8].

[2] By definition, a fault that is undetected by the active redundancy techniques implemented in a non-faulty node is malicious.

$\mathcal{B}$ include benign faults, crash faults, and the subset of Byzantine faults that can be detected using active redundancy techniques, such as sanity checks.

The overly-pessimistic reliability models assume all faults to be of type $\mathcal{A}$. An overly-optimistic model takes all faults to be in $\mathcal{B}$. The *hybrid fault model*[7] combines commonly used single fault-type reliability models with the hybrid taxonomy to cover mixed fault types. The standard fault tolerance algorithms are replaced by hybrid algorithms that identify non-malicious faults and ignore them during masking or voting. If no hybrid algorithms are used, then the hybrid model reverts to the standard worst case single-fault model with no improvement.

For synchronization in the cluster model, the fault set is $\mathcal{F} \equiv \mathcal{S} \bigcup \mathcal{B} \bigcup \mathcal{A}$ for both the node and cluster faults; so, all possible faults are covered. A minimum of $n = 2f_{\mathcal{A}} + 2f_{\mathcal{S}} + f_{\mathcal{B}} + \tau + 1$ nodes is sufficient to tolerate $f_{\mathcal{A}} + f_{\mathcal{B}} + f_{\mathcal{S}}$ faults, with $f_{\mathcal{A}} \leq \tau$ within each fully connected cluster, assuming that a hybrid interactive consistency algorithm is used and that $\tau = r$, the number of rebroadcast rounds implemented in the algorithm. At least $\tau + 1$ good nodes are assumed to be necessary for the system to remain operational.

Table 6 shows the improvement in estimated reliability over the Byzantine Generals'(BG) model assuming exponential node failure rates of 1×10^{-4} per hour. The additional fault scenarios covered by the hybrid model(HM) are also listed, with the assumed probabilities of fault types $(f_{\mathcal{A}}, f_{\mathcal{B}}, f_{\mathcal{S}})$ taken as $(.01, .98, .01)$. Table 7 shows the effects of improved cluster-level reliability estimates upon the system reliability in a 40 node cluster system, with two failed clusters leading to a failed system state. In both tables, the combinatoric expression of Equation 1 and the reliability expression given in [7] are used to estimate the reliabilities under the assumption that no repair or reconfiguration occurs.

Model	N	$1 - R_{sys}$ (1 hr)	Fault Handling Capability
BG	4	6.0×10^{-8}	$f_{\mathcal{A}} = 1$
BG	5	1.0×10^{-7}	$f_{\mathcal{A}} = 1$
BG	6	1.5×10^{-7}	$f_{\mathcal{A}} = 1$
HM	4	2.4×10^{-9}	$f_{\mathcal{A}} = 1, f_{\mathcal{B}} = 0, f_{\mathcal{S}} = 0$ $f_{\mathcal{A}} = 0, f_{\mathcal{B}} \leq 2, f_{\mathcal{S}} = 0$ $f_{\mathcal{A}} = 0, f_{\mathcal{B}} = 0, f_{\mathcal{S}} = 1$
HM	5	4.1×10^{-11}	$f_{\mathcal{A}} = 1, f_{\mathcal{B}} \leq 1, f_{\mathcal{S}} = 0$ $f_{\mathcal{A}} = 0, f_{\mathcal{B}} \leq 3, f_{\mathcal{S}} = 0$ $f_{\mathcal{A}} = 0, f_{\mathcal{B}} \leq 1, f_{\mathcal{S}} = 1$
HM	6	1.5×10^{-11}	$f_{\mathcal{A}} = 1, f_{\mathcal{B}} \leq 2, f_{\mathcal{S}} = 0$ $f_{\mathcal{A}} = 1, f_{\mathcal{B}} = 0, f_{\mathcal{S}} \leq 1$ $f_{\mathcal{A}} = 0, f_{\mathcal{B}} \leq 4, f_{\mathcal{S}} = 0$ $f_{\mathcal{A}} = 0, f_{\mathcal{B}} \leq 2, f_{\mathcal{S}} = 1$ $f_{\mathcal{A}} = 0, f_{\mathcal{B}} = 0, f_{\mathcal{S}} = 2$

Table 6: Hybrid Model System Reliability

Clus.	BG Model	Hybrid Model
(n, v)	(1-R)$_C$[a] : (1-R)$_{Sys}$ @20 hrs : @50 hrs	(1-R)$_C$: (1-R)$_{Sys}$ @20 hrs : @50 hrs
(10,4)	2.39E-5 : 3.91E-10	9.77E-7 : 3.07E-14
(8,5)	3.98E-5 : 8.36E-10	2.06E-8 : 3.53E-17
(7,6)	5.96E-5 : 1.75E-9	6.15E-9 : 1.39E-17

[a] (1-R)$_C \equiv$ (1-R) for C

Table 7: System Reliability for Different Cluster Sizes

5 Conclusions

We have presented a cluster based architecture that provides tight cluster level and system level synchronization without the graph and algorithm complexity limitations of a fully connected structure. We also demonstrated the imprecision of conventional combinatorial techniques for modeling cluster based systems. The reliability of a distributed system was shown to be a function of both the number of faults and their spatial distribution, an aspect disregarded in earlier schemes. Discrete Markov modeling techniques have been developed to cover such reliability modeling scenarios.

At the cluster level, we have described methods to distinguish faults and classify them depending upon the type and severity of errors they cause. This is shown to provide a more realistic estimation of the reliability of a system than the existing overly-pessimistic and overly-optimistic models. This mechanism also provides a way of choosing linearly increasing cluster sizes with associated linearly increasing reliability values, contrary to the predictions of the usual Byzantine reliability models.

We need to extend the system level reliability models to consider (a) varying cluster sizes, (b) allowing for reconfigurations at the inter-cluster level and (c) modelling correlations among clusters. The fault distribution model and the reliability model using fault classifications have currently been obtained on an individual basis. For a full and comprehensive system reliability model, these need to be integrated along with consideration of link faults.

References

[1] S. Bavuso *et al.*, "Analysis of typical fault-tolerant architecture using HARP," *IEEE Trans.*

on Reliability, vol. r-36, pp. 176–185, June 1987.

[2] D. Dolev *et al.*, "An efficient algorithm for Byzantine agreement without authentication," *Journal of Info. and Control*, 52, pp. 257–274, 1982.

[3] D. Dolev *et al.*, "Reaching approximate agreement in the presence of faults," *IEEE Sym. on Reliability in Real Time Sys.*, pp. 145–154, Oct 1983.

[4] J. Halpern *et al.*, "Fault tolerant clock synchronization," *Principles of Distr. Computing*, pp. 89–102, 1984.

[5] R. Harper, *Critical Issues in Ultra-Reliable Parallel Processing*. PhD thesis, MIT, Cambridge, Boston, May 1987.

[6] A. L. Hopkins *et al.*, "FTMP – a highly reliable fault-tolerant multiprocessor for aircraft," *Proceedings of IEEE*, vol. 66, Oct 1978.

[7] M. M. Hugue, "Static reliability modelling with the hybrid fault model," *Proceedings of ONR Ultra Dependable Multi-computers and Electronic Systems*, pp. 23 – 28, Nov 1991.

[8] B. Johnson, *Design and Analysis of Fault Tolerant Digital Systems*. Addison-Wesley, 1989.

[9] R. Kieckhafer, C. Walter, A. Finn, and P. Thambidurai, "The MAFT architecture for distributed fault tolerance," *IEEE Trans. on Computers*, vol. 37, pp. 398–405, Apr 1988.

[10] C. L. Liu, *Elements of Discrete Mathematics*. McGraw-Hill, 1977, 1985.

[11] J. Lundelius-Welch and N. Lynch, "A new fault tolerant algorithm for clock synchronization," *Information and Computation*, vol. 77, no. 1, pp. 1–36, 1988.

[12] N. Lynch *et al.*, "A simple and efficient Byzantine generals algorithm," *Proc. 2nd Symp. on Reliability in Distributed Software and Database Systems*, pp. 46–52, July 1982.

[13] P. Ramanathan, D. Kandlur, and K. Shin, "Hardware-assisted software clock synchronization for homogeneous distributed systems," *IEEE Trans. on Computers*, vol. C-39, pp. 514–524, April 1990.

[14] K. Shin and P. Ramanathan, "Clock synchronization of a large multiprocessor system in the presence of malicious faults," *IEEE Trans. on Computers*, vol. c-36, pp. 2 – 12, Jan 1987.

[15] T. K. Srikanth and S. Toueg, "Optimal clock synchronization," *Journal of ACM*, vol. 34, July 1987.

[16] P. Thambidurai and Y.-K. Park, "Interactive consistency with multiple failure modes," *Proc. Symp. on Reliable Dist. Systems*, pp. 93–100, Oct 1988.

[17] P. Thambidurai, Y.-K. Park, and K. Trivedi, "On reliability modelling of fault-tolerant distributed systems," *Intl. Conf. on Distr. Computing*, pp. 136–142, June 1989.

[18] J. W. Wensley *et al.*, "SIFT : Design and analysis of a fault-tolerant computer for aircraft control," *Proceedings of IEEE*, vol. 66, Oct 1978.