Achieving Agreement In Three Rounds With Bounded-Byzantine Faults

Mahyar R. Malekpour[*]
Langley Research Center, Hampton, VA

A three-round algorithm is presented that guarantees agreement in a system of $K \geq 3F+1$ nodes provided each faulty node induces no more than F faults and each good node experiences no more than F faults, where, F is the maximum number of simultaneous faults in the network. The algorithm is based on the Oral Message algorithm of Lamport, Shostak, and Pease and is scalable with respect to the number of nodes in the system and applies equally to traditional node-fault model as well as the link-fault model. We also present a mechanical verification of the algorithm focusing on verifying the correctness of a bounded model of the algorithm as well as confirming claims of determinism.

Keywords: Oral Message, Agreement, Byzantine, fault tolerant, synchronization, distributed, model checking

Nomenclature

D	=	minimum communication event-response delay
d	=	network communication imprecision
γ	=	$D + d$
Sync	=	communication message
F	=	maximum number of faults in the network
K	=	number of nodes in the network
i	=	1..K
N_i	=	ith Node
f_i	=	number of faults associated with N_i

I. Introduction

Distributed systems have become an integral part of safety-critical computing applications, necessitating system designs that incorporate complex fault-tolerant resource management functions to provide globally coordinated operations with ultra-reliability. As a result, robust clock synchronization has become a required fundamental component of fault-tolerant safety-critical distributed systems. Synchronization has practical significance as a fundamental service for higher-level algorithms that solve other problems. For example, in safety-critical TDMA (Time Division Multiple Access) architectures[1,2,3], synchronization is the most crucial element of these systems. Typically, the assumed topology is a regular graph such as a fully connected graph or a ring since they provide a base case to solve the distributed synchronization problem.

A fundamental property of a robust distributed system is the capability of tolerating and potentially recovering from failures (loss of service due to a fault) that are not predictable in advance. A *fault* is a defect or flaw in a system component resulting in an incorrect state[2,4]. The requirement to handle faults adds a new dimension to the complexity of the synchronizing distributed systems. In the context of fault-tolerant distributed systems, a fault presenting different symptoms to different observers is known as a Byzantine (arbitrary) fault. We assume that there are a maximum of F simultaneous faults in the network.

We call an approach to solving the clock synchronization problem *direct* if it relies solely on local (node level) detection and filtering of faults. This approach is primarily limited to detecting timing and/or value faults of a node's incoming messages. In contrast, we call an approach *indirect* if it relies on the network level detection and

[*] Aerospace Technologist, NASA LaRC, 1 S. Wright St. MS130, Hampton, VA, 23681-2199, senior AIAA.

filtering of faults independent of, and in addition to, the local detection and filtering of the faults. This approach however requires coordination at the network level.

Thus far, there is no verifiable solution for the general case of the clock synchronization problem, where the topology is arbitrary and any number of various types of faults are tolerated. Furthermore, most attempts have been in trying to solve this problem directly, although there are some approaches to solve this problem indirectly using authenticated (signed) messages[5]. Driscoll[6], Hall, Sivencrona, and Zumsteg however argue that "while the arguments of unforgeable signed messages make sense in the context of communicating generals, the validity of necessary assumptions in a digital processing environment is not supportable. In fact, the philosophical approach of utilizing cryptography to address the problem within the real world of digital electronics makes little sense. The assumptions required to support the validity of unbreakable signatures are equally applicable to simpler approaches (such as appending a simple source ID or a CRC to the end of a message). It is not possible to prove such assumptions analytically for systems with failure probability requirements near 10^{-9}/hr." Furthermore, addressing network element imperfections, such as oscillator drift with respect to real time and differences in the lengths of the physical communication media, is necessary to make a solution applicable to realizable systems.

The main issue in solving the clock synchronization problem is a lack of symmetric view (agreement) in the system among the participating good nodes in the sense that two good nodes may disagree on the message sent. However, there are a number of ways of achieving message symmetry across the system. In Ref. 5 and 7 various ideas for overcoming failures in a robust distributed system are addressed that include tolerating Byzantine faults. In solving the consensus problem, which is the ability of a set of nodes to agree on a single value despite failures, Schmid, Weiss, and Keidar argue in Ref. 8 that: "A fully-fledged n-process consensus algorithm is obtained by using a separate instance of a Byzantine agreement algorithm (with n-1 receivers) for disseminating any process's local value, and using a suitable choice function (majority) for the consensus result[†]." The consensus problem, and hence, the proposed idea by Schmid, is based on an inherent assumption of synchrony among the good nodes, and so is not applicable to solving the clock synchronization problem.

Other methods include using variety of engineering practices, e.g., using a self-checking pair at the node level[9,10] or central guardians at the system level[11,12]. However, as Driscoll, Hall, Sivencrona, and Zumsteg reported in Ref. 6, correctness of claims of these approaches may not be verifiable. Furthermore, we believe that to be generally useful, algorithms that guarantee agreement must be able to handle non-authenticated messages. Thus, the crux of our idea, as proposed in Ref. 13, is to solve this problem indirectly by first converting any message to a symmetric message, and then use a verified protocol based on the symmetry assumption to solve the synchronization problem.

The Oral Message (*OM*) algorithm of Lamport[5], Shostak, and Pease that solves the Byzantine Agreement (BA) problem[14] is also an indirect approach, and is meant to reliably transform a message from a single source to a symmetric message (an agreement) for a fully connected network. The *OM* algorithm has been proven to reach agreement at the network level for a given source[5,14,15] and does not require initial synchrony among the good nodes. The *OM* requires $F+1$ rounds of exchanges and, with a message complexity of $O(K^F)$, the number of exchanged messages grows exponentially as F grows linearly. Therefore, the use of the *OM* algorithm for $F > 2$ is very costly and impractical.

In this paper, we present an alternative for achieving agreement, hereafter referred to as *3ROM* (3 Rounds using *OM*) algorithm that is based on the *OM* algorithm. The *3ROM* guarantees agreement in a system of $K \geq 3F+1$ nodes, where F is the maximum number of simultaneous faults in the network. The *3ROM* assumes each node N_i, $i = 1..K$, either induces up to F faults if it is a faulty node, or experiences no more than F faults if it is a good node. The *3ROM* algorithm is independent of the fault model (node-fault or link-fault model), and as the name implies, achieves agreement in three rounds. Thus, it is independent of the number of faults (in terms of number of required rounds, not the amount of messages). The algorithm has a message complexity of $O(K^3)$, and is also scalable with respect to K. We also present the model checking results of a bounded model of the algorithm to verify its correctness.

This paper is organized as follows. We describe the fault models in Section II. In Section III we provide a system overview. We present the *3ROM* algorithm and its formal proof in Section IIIV. In Section V, we present the model checking efforts toward verification of correctness of a bounded model of the algorithm and the results of that effort. Finally, we present concluding remarks in Section VI.

[†] Since we use N_i to address a node, we use K here instead of n as is traditionally used in the literature.

II. Fault Models

In synchronous message-based distributed systems, a fault is typically defined as a message that was not transmitted when it was expected or a message that was transmitted but not received or received but not accepted, i.e., deemed invalid by a receiver. Thus, the fault is either associated with the source node of the message, the corresponding link between the source node and the destination node, or the destination node. Consequently, there are two viewpoints, *node-centric* and *link-centric*, and thus, there are two ways of modeling faults. In the node-centric model, we refer to as the *node-fault model*, the faults are associated with the source node of the message and all fault manifestations between the source and the destination nodes for the messages from that source count as a single fault, which is specially the case when the faults are associated with a Byzantine faulty node[5,6,16,17]. In this model all links are assumed to be good. Miner[16], Geser, Pike, and Maddalon, for instance, model the absence of a link as a link fault and even though both nodes and links failures are considered, they abstractly model link failures as failures of the source node.

In the link-centric model, we refer to as the *link-fault model*, a fault is associated with the communication means connecting the source node to the destination node. In this model, all nodes are assumed to be good and an invalid message at the receiving node is counted as a single fault for the corresponding input link. Thus, from the global perspective, a Byzantine faulty node manifests as multiple link failures.

A link-fault model introduced by Schmid[18], Weiss, and Rushby is called *perception-based hybrid fault model*, where faults are viewed from the perspective of the receiving nodes. Faults are associated with their input links, and all nodes are assumed to be good. They argued that since F faulty nodes can produce at most F faulty perceptions in any node, the link-fault model is compatible with traditional node-fault model and so, all existing lower bound and impossibility results remain valid.

“In the perception-based model, the system-wide number of faults is replaced by the number of faults that are observable in the nodes’ local “perceptions” of the system. Formally, node r’s *perception vector* $V_r = (V_r^1, V_r^2, \ldots, V_r^K,)$ is considered, where every *perception* $V_r^s \in V_r$ represents the message node r received from node s in some specific round; type and value(s) depend upon the particular algorithm considered[18]”. In that paper, Schmid[18], Weiss, and Rushby present a solution for synchronous deterministic consensus problem, where all nodes are expected to achieve agreement on a single value, in synchronous distributed systems with link faults.

III. System Overview

We consider “synchronous” message-passing distributed systems and model the system as a graph with a set of nodes (vertices) that represent the pulse-coupled entities and a set of communication links (edges) that represent their interconnectivity. Same as the *OM* algorithm, the underlying topology considered is a fully connected network of K nodes that exchange messages through a set of communication links. The system consists of a set of good nodes and a set of faulty nodes. A good node is assumed to be an active participant and correctly execute the algorithms. A faulty node is either benign (detectably bad), symmetrically faulty, or arbitrarily (Byzantine) faulty. However, in this paper our primary focus is Byzantine faults.

The communication links are point-to-point and unidirectional, each connecting a source to a destination node. Thus, the fully connected graph consists of $K(K\text{-}1)$ unidirectional links. A good link is assumed to correctly deliver a message from its source node to its destination node within a bounded communication delay time. A faulty link does not deliver the message, delivers a corrupted message, or delivers a message outside the expected communication delay time.

The nodes communicate with each other by exchanging broadcast messages. Broadcast of a message by a node is realized by transmitting the message, at the same time, to all nodes that are directly connected to it. The communication network does not guarantee any relative order of arrival of a broadcast message at the receiving nodes, that is, a consistent delivery order of a set of messages does not necessarily reflect the temporal or causal order of the message transmissions[1]. A maximum of F faults are assumed to be present in the system, where $F \geq 0$. We assume $K \geq 3F+1$ and define the minimum number of good nodes in the system, G, by $G = K\text{-}F$ nodes. The minimum number of nodes needed[6,14,19] to maintain synchrony is well established to be $3F+1$.

A. Communication Delay

The communication delay between directly connected (adjacent) nodes is expressed in terms of the minimum event-response delay, D, and network imprecision, d. These parameter are measured at the network level. A message broadcast by node at real time t is expected to arrive at its directly connected adjacent nodes, be processed, and subsequent messages to be generated by those nodes within the time interval $[t+D, t+D+d]$. Communication between independently-clocked nodes is inherently imprecise. The network imprecision, d, is the maximum time difference among all receivers of a message from a transmitting node with respect to real time. The imprecision is due to many factors including, but not limited to, the drift of the oscillators with respect to real time, jitter, discretization error, temperature effects and differences in the lengths of the physical communication media. These parameters are assumed to be bounded, $D > 0$, $d \geq 0$, and both have units of real-time clock ticks and their values known in the network. The communication delay, denoted γ, is expressed in terms of D and d, is defined as $\gamma = D+d$, and has units of real-time clock ticks. In other words, we assume synchronous communication and bound the communication delay between any two directly connected adjacent nodes by $[D, \gamma]$. However, for simplicity of notation, in the remainder of this paper we assume that the messages arrive *logically* at the same time at the destination nodes.

B. The *Sync* Message And Its Validity

In order to achieve and maintain desired synchrony, the nodes communicate by exchanging *Sync* messages, where *synchrony* is defined as a measure of the relative imprecision of the good nodes. Assuming physical-layer error detection is dealt with separately, the reception of a *Sync* message is indicative of its validity in the value domain. A *Sync* message from a given source is *valid* if it arrives at or after one D of an immediately preceding *Sync* message from that source, that is, the message validity in the value domain, i.e., valid *Sync* messages are rate-constrained. Again, assuming physical-layer error detection is dealt with separately, the reception of a *Sync* message is indicative of its validity in the value and time domains.

IV. 3ROM

In a synchronous distributed system, the Oral Message algorithm of Lamport[5], Shostak, and Pease solves the Byzantine Agreement (BA) problem[14] by reliably transforming a message, in the presence of faults, to a symmetric message at the network level, whereby the good nodes reach an agreement and collectively either accept or reject the message. The *OM* algorithm is recursive and every iteration of the execution of the algorithm constitutes a step (round) of exchange of messages by the nodes. An instance of the *OM* algorithm starts with the source node broadcasting a message, the first round, followed by other nodes (except the source of the message) recursively rebroadcasting (relaying) the messages they receive to others in subsequent rounds. For a fully connected graph of K nodes, the *OM* algorithm requires $F+1$ communication rounds. At the end of the $F+1$ rounds, the nodes vote and reach agreement.

In this section we present a three-round algorithm, similar to the *OM* algorithm, that achieves agreement among the good nodes, within a time bound, independent of F, provided that $K \geq 3F+1$ nodes, where K is the number of nodes, and $f_i \leq F$ for $i = 1..K$, where f_i is the number of faults associated with N_i. For the node-fault model, the faulty nodes do whatever they want as long as their behavior is bounded by the assumption, i.e., at any round a faulty node broadcasts a valid message to at least K-F other nodes. Similarly, for the link-fault model, at any round no more than F link faults are perceived by a receiving good node. We will also show that this three-round algorithm applies equally to the node-fault and link-fault models. To simplify presentation of this algorithm, we assume broadcast messages arrive logically at the same time at their destination good nodes. We later remove this simplifying assumption and show that agreement is reached at the good nodes within a time bound. We use two types of messages: *Sync* and *Relay*, and extend the message validity argument of Section III.B to both messages.

The *3ROM* Algorithm

The algorithm depends on two positive parameters α and β, which are used in determining the final acceptance or rejection of a *Sync* message from a source node. Hence, the algorithm described is actually *3ROM(α, β)*. The algorithm consists of three rounds and a vote

Round 1 – The source node broadcasts a *Sync* message to all other nodes, effectively saying "*I'm here*." A node does not send a message to itself even though it uses its own message. The nodes that receive the message record that the message was received.

Round 2 – All nodes (including the source node) that received a *Sync* message in *Round 1* broadcast a *Relay* message to all other nodes, essentially saying "*I've got a message*." The good nodes that do not receive the *Sync* message do nothing.

Round 3 – All nodes broadcast a vector of K messages containing what they have received from all other nodes in *Rounds 1* and *2*; effectively saying "*This is what I've received from others*."

Vote – At the end of *Round 3*, each node locally constructs a *KxK* network-level matrix M of received messages, where entries $M(i, j) = \{s, r, 0\}$, $i, j = 1..K$, where '*s*' indicates having received a *Sync* message, '*r*' indicates having received a *Relay* message and '*0*' for not receiving any messages, i.e., a fault. A column c_j of the matrix M corresponds to the messages transmitted by N_j and a row r_i of the matrix M corresponds to the messages received by N_i. Let, for $i, j = 1..K$, $X_i = 1$ if $\sum c_j > \alpha$ and $X_i = 0$, otherwise, where $\sum c_j$ is the sum of the non-zero entries in column c_j, i.e., treating '*s*' and '*r*' entries equally. Finally, the node votes "accept" if $\sum X_i > \beta$, i.e., the node accepts the message from the source node if more than β columns of M have more than α non-zero entries each.

In addition to the network-level matrix M that each node constructs at the end of *Round 3*, the proofs to follow rely on another related matrix M_{global} that can be constructed at the end of *Round 2*. Following *Round 2*, each node has a vector of received messages that describe the messages it received from each node. The M_{global} with entries related to the matrix M, i.e., $M_{global}(i, j) = \{s, r, 0\}$, reflects a global view of the network at the end of *Round 2*. While it is inaccessible to any nodes, it is related to the matrices M built by each node.

We need to point out that in the *3ROM* algorithm, in rounds 2 and 3, if a node does not receive a message, it does not broadcast anything. This is in contrast to the *OM* algorithm where, after the first round, a node that did not receive a message is required to transmit a default message to others.

A. Proof Of The *3ROM* Algorithm For Link-Fault Model

The proof of correctness of the *OM* algorithm, and consequently, the proof of correctness of the *3ROM* algorithm, is centered on the following two properties.

AP (Agreement Property): If receiver nodes p and q are nonfaulty, then they agree on the value ascribed to the transmitter.

VP (Validity Property): If the transmitter is nonfaulty, then every nonfaulty receiver computes the correct value.

Theorem 1. *For a fully connected graph with $K > 3F$ nodes and the link-fault model, i.e., $f_l \leq F$, 3ROM(K/3, 2K/3) guarantees agreement at the good nodes.*

Proof. The source node is a good node (all nodes are good in the link-fault model).

Round 1 – N_S, the source node, broadcasts a *Sync* message and it is received correctly (valid) by at least *K-F* nodes. Let H be this set of nodes.

Round 2 – Each node in H broadcasts a *Relay* message to all other nodes and at least *K-F* other nodes will receive the *Relay* message correctly. Hence, for each node in H, its corresponding column in M_{global} has at least *K-F* non-zero entries, i.e., '*s*' and '*r*'.

Round 3 – Each node broadcasts its vector of received messages to all other nodes and at least *K-F* nodes will receive it correctly. At the end of this round, a node constructs its matrix M, which is similar to, but likely different from, M_{global}. Since the rows of M are the messages in *Round 3*, at most F rows can be different from M_{global}. Thus, the sum of non-zero entries in any column in M_{global} is reduced by at most F. Thus, the columns in M corresponding to nodes in H have at least *K-F-F* non-zero entries. Since $K > 3F$, (equivalently, $K/3 > F$), there are at least $K\text{-}F > 2K/3$ columns in M (the nodes in H), with at least $K\text{-}2F > K/3$ non-zero entries. Thus, each node votes "accept" for *3ROM(K/3, 2K/3)*. □

Table 1 is an example of the network-level matrix at the end of *Round 3* for $F = 2$ and $K = 7$. The grayed cells along the diagonal in this matrix are the messages a node sends to itself, thus, cannot be faulty. An "~~s~~ *r*" entry indicates that a *Sync* message was received in *Round 2* and was replaced by a *Relay* message from the same node in *Round 3*.

Table 2 shows the matrices at N_1 and N_2, as examples, at the end of *Round 3*. The grayed rows indicate the effects of link faults at the two nodes in *Round 3*.

Table 1. An example of matrix of received messages at the end of *Round 3*.

N_i	1	2	3	4	5	6	7
1	~~s~~ r	0	0	r	r	0	0
2	~~s~~ r	r	0	0	r	0	0
3	~~s~~ r	r	r	0	0	0	0
4	~~s~~ r	0	r	r	0	0	0
5	s	r	r	r	r	0	0
6	~~0~~ r	r	r	r	r	0	0
7	0	r	r	r	r	0	0
X_i	1	1	1	1	1	0	0

Table 2. An example of matrix of received messages at N_1 and N_2 at the end of *Round 3*.

N_1	1	2	3	4	5	6	7	
1	~~s~~ r	0	0	r	r	0	0	
2	~~s~~ r	r	0	0	r	0	0	
3	~~s~~ r	r	r	0	0	0	0	
4	~~s~~ r	0	r	r	0	0	0	
5	s	r	r	r	r	0	0	
6	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	
X_i	1	1	1	1	1	0	0	V=1

N_2	1	2	3	4	5	6	7	
1	0	0	0	0	0	0	0	
2	~~s~~ r	r	0	0	r	0	0	
3	~~s~~ r	r	r	0	0	0	0	
4	~~s~~ r	0	r	r	0	0	0	
5	s	r	r	r	r	0	0	
6	~~0~~ r	r	r	r	r	0	0	
7	0	0	0	0	0	0	0	
X_i	1	1	1	1	1	0	0	V=1

B. Proof Of The *3ROM* Algorithm For Node-Fault Model

In the classic node-fault model, a node's message may be perceived as faulty by many other nodes. However, we assume there are up to F simultaneous Byzantine faulty nodes present in the network and they behave arbitrarily but are limited to inducing no more than F faults at each round, i.e., $f_i \leq F$.

Theorem 2. *For a fully connected graph with $K \geq 3F+1$ nodes and the node-fault model, i.e., $f_i \leq F$, 3ROM(K/3, 2K/3) guarantees agreement at the good nodes.*

Proof. Note that a bounded-Byzantine faulty node can be modeled by a link-fault model, where the faulty links are exactly those where the received messages are invalid. Thus, whether the source node is good or Byzantine faulty, since the maximum number of bounded-Byzantine faulty nodes, F, is less than a third of K and since the link-fault model allows each node up to F faults per round, the proof of Theorem 1 applies in this case. □

Table 3 is an example of the matrix after *Round 3* at N_i. We would like to point out that, unlike the Table 2 for the link-fault model, this matrix is the same at all good nodes except for the rows and columns corresponding to the faulty nodes; c_6, c_7, and r_6, r_7, respectively.

Table 3. An example of matrix of received messages at the good nodes N_i, i = 1..5, at the end of *Round 3*. N_6 and N_7 are the Byzantine faulty nodes and N_6 is the source node.

N_i	1	2	3	4	5	6	7	
1	r	r	r	0	0	~~s~~ r	0	
2	r	r	r	0	0	~~s~~ r	r	
3	r	r	r	0	0	s	r	
4	r	r	r	0	0	~~0~~ r	0	
5	r	r	r	0	0	0	r	
6	x	x	x	x	x	~~s~~ r	r	
7	x	x	x	x	x	~~s~~ r	r	
X_i	1	1	1	0	0	1	1	V = 1

Theorem 3 (Agreement). *For any F and K, for a fully connected graph with $K \geq 3F+1$ nodes and $f_i \leq F$, the 3ROM algorithm satisfies AP at the good nodes.*

Proof. It follows from Theorems 1 and 2 that, if the assumptions are met, the *3ROM* algorithm always guarantees agreement at the good nodes. □

Theorem 4 (Validity). *For any F and K, for a fully connected graph with $K \geq 3F+1$ nodes and $f_i \leq F$, algorithm 3ROM satisfies AP and VP.*

Proof. It follows from Theorems 1 and 2 that, if the assumptions are met, the *3ROM* algorithm satisfies the agreement and validity properties at the good nodes. □

Corollary 5. *The number of rounds required by 3ROM algorithm is independent of F.*

Proof. It follows from Theorem 3 that the *3ROM* algorithm always guarantees agreement at the good nodes in three rounds and regardless of a particular value of F. □

Theorem 6. *For a fully connected graph with $K \geq 3F+1$ nodes and $f_i \leq F$, the link-fault model supersedes the node-fault model.*

Proof. Given the assumptions that $f_i \leq F$, i.e., a node either is faulty and induces up to F faults per round or it is good and experiences no more than F faults per round. Given the link-fault model, the *3ROM* algorithm converts any message from a node N_i to a symmetric message in three rounds. With the link-fault model, the nodes are considered to be good even though the faults are manifested on their links. Thus, given up to F faults per node (i.e., the maximum number of outgoing faulty links per node), a maximum of KF faults per round are tolerated. With the node-fault model, a maximum of F faulty nodes are assumed to be present with up to F faults per outgoing links of a faulty node, thus, a total of F^2 faults per round are tolerated. Since for $F > 0$, $F^2 < KF$, the link-fault model supersedes the node-fault model. □

Thus far, we assumed that the Byzantine faulty nodes behave arbitrarily but are limited to inducing no more than F faults at any round. We now weaken the assumption of $f_i \leq F$ so that a faulty node behaves arbitrarily in *Round 2* and/or *Round 3*. The *3ROM* algorithm still achieves agreement, but, the voting criteria needs to be adjusted to accommodate this weakened assumption, i.e., *3ROM(K/3, K/3+1)*. One manifestation of a faulty behavior is for the node to not broadcast anything during *Round 2* and/or *Round 3*. Note that when a node fails crash-silent, $f_i = K$ and it can readily be detected from the network-level matrix at the end of *Round 3* since its corresponding column will have at least *K-F* zeroes. This diagnosis information can potentially be used at the network level. We now show that the *3ROM* algorithm still achieves agreement when the source is a Byzantine faulty node.

Theorem 7. *For a fully connected graph with K > 3F nodes and the node-fault model, i.e., $f_i \leq F$, when the source node is a Byzantine faulty node, 3ROM(K/3, K/3+1) guarantees agreement at the good nodes.*

Proof. The source node is a Byzantine faulty node.

Round 1 – The source node broadcasts a *Sync* message to at least *K-F* nodes where at least $K\text{-}F^{‡}\text{-}F^{§}$ are good nodes and receive the message correctly (valid). Let *H* be this set of good nodes.

Round 2 – Each node in *H* broadcasts a *Relay* message to all other nodes and at least *K-F* other nodes will receive the *Relay* message correctly. Hence, for each node in *H*, its corresponding column in M_{global} has at least *K-F* non-zero entries.

Round 3 – Each node broadcasts its vector of received messages to all other nodes and at least *K-F* nodes will receive it correctly. At the end of this round, a node constructs its matrix *M*, which is similar to, but likely different from, M_{global}. The matrix *M* at a good node N_i is different from M_{global} in at most *F* rows and *F* columns corresponding to the Byzantine faulty nodes. Also, the matrix *M* at a good node N_i is identical to the matrix *M* at other good nodes N_j, j=1..G and $j \neq i$, except in the same rows and columns corresponding to the Byzantine faulty nodes. Therefore, the columns in *M* corresponding to nodes in *H* have at least *K-F* non-zero entries. Furthermore, the column corresponding to the faulty source node, has at least *K-2F* non-zero (the nodes in *H*) entries in *M*. Since $K > 3F$, (equivalently, $K/3 > F$), there are at least $K\text{-}2F\text{+}1 > K/3\text{+}1$ columns in *M* (the nodes in *H* plus the faulty source), with at least $K\text{-}2F > K/3$ non-zero entries. Thus, each node votes "accept" for *3ROM(K/3, K/3+1)*. □

Note that when the source is a good node (node-fault model), since the set *H* consists of at least *2K/3* good nodes, this weaker assumption holds and Theorems 2 and 7 apply. Also, although this weaker assumption does not apply to the link-fault model (all nodes are good) and Theorems 1 still holds, nevertheless, since with this weaker assumption, $\beta = K/3+1$ and $K/3+1 < 2K/3$, Theorem 7 applies to both models.

C. Message Observation Window, Agreement Within A Time Bound

Earlier in this paper we stated that to simplify the explanation of the problem and our proposed solution, a transmitted message from a single source arrived at the receiving nodes *logically* at the same time. In this section we visit this assumption and justify this rationality. In an implementation, as we have explained in Section III.A, a given message from a single source arrives at the receiving nodes within *d* units of each other. Figure 1 is a depiction of a message through three rounds of the *3ROM* algorithm. N_S is the source node, N_i and N_j represent the nodes that receive the message at the two extremes of the communication latencies, i.e., *D* and $D+d = \gamma$, respectively, $i \neq j \neq S$. Thus, unless proper measures are taken, consequent relaying of messages at subsequent rounds widens

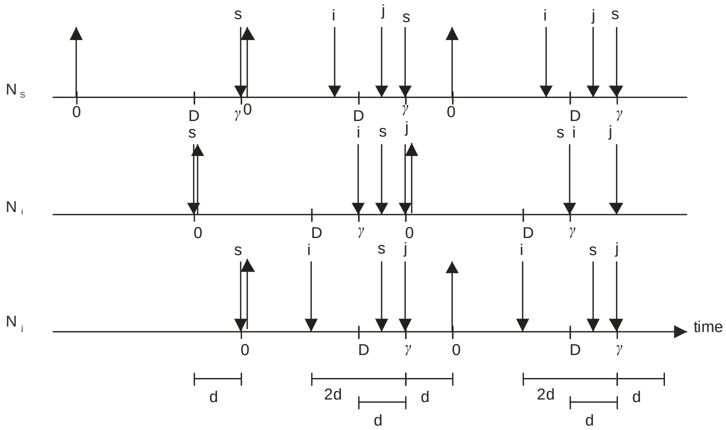

Figure 1. Message observation window = [-2*d*, *d*] from *γ*.

[‡] Up to *F* good nodes do not receive the message.

[§] Up to *F* simultaneous faulty nodes.

message arrivals at the nodes for every round by an additional *d*. In this figure, '↑' indicates broadcasting a message, '↓' indicates receiving a message, and the labels on these arrows, *s*, *i*, and *j*, correspond to N_S, N_i, and N_j, respectively, as the initiators of the messages.

For the following lemmas, N_S is the source node initiating *Round 1*, N_i is the node that receives the message at the earliest time, i.e., *D*, and N_j is the node that receives the message at the latest time, i.e., $D+d = \gamma$.

Lemma 8. *Message observation window for Round 2 is [*γ*-2d,* γ*+d].*

Proof. *Round 2* begins by N_i and N_j relaying the messages they received from N_S. For this round, the nodes relay the messages as soon as they receive it, i.e., within at most *d* of each other. Thus, at the end of *Round 2*, the messages arrive at the nodes within 2*d* of each other. Since a node does not physically send a message to itself, but uses its own message, to account for the worst case message delivery time, its message is assumed to arrive at itself at γ. At the end of *Round 2*, the earliest a message can arrive is at N_j and from the first node that started its *Round 2*, i.e., N_i. As shown on the timeline of activities, this message arrives at the longest delay minus the accumulated drift for two rounds, i.e., γ-2*d*. Similarly, the latest a message can arrive is at N_i and from the node that started its *Round 2* last, i.e., N_j, and at the longest delay plus the initial drift between the nodes from the previous round, i.e., γ+*d*. Thus, the window of observation for message arrival for *Round 2* is [γ-2*d*, γ+*d*]. □

Lemma 9. *All good nodes participating in Round 2 finish Round 2 and start Round 3 within d of each other*.

Proof. From Figure 1, from the start of *Round 1* to the end of *Round 2*, the earliest message (EM) and latest message (LM) arrival time at N_S at $EM_S = \gamma+D-d$ and $LM_S = \gamma+\gamma$, respectively. Similarly, for N_i, $EM_i = D+d+D$, $LM_i = D+d+\gamma$, and for N_j, $EM_j = \gamma+D-d$, $LM_j = \gamma+\gamma$. Simple algebraic operations results in $\Delta_{EM} = d$ and $\Delta_{LM} = 0$ for any two nodes, i.e., the nodes finish *Round 2* and start *Round 3* within *d* of each other. □

Lemma 10. *Message observation window for Round 3 is [*γ*-2d,* γ*+d].*

Proof. It follows from Lemma 9 that the nodes start *Round 3* within *d* of each other. In a similar argument as Lemma 8 the window of observation for message arrival for *Round 3* is [γ-2*d*, γ+*d*]. □

It follows from Lemmas 8 through 10 that at the end of each round the nodes are within *d* of each other, thus, justifying rationality of our assumption of logical timing of arrival of messages.

D. Complexity Of The *3ROM* Algorithm

In the *3ROM* algorithm, since a node does not send a message to itself, the number of transmitted messages per node and for each round is (*K*-1). For the worst case analysis, all nodes participate in *Rounds 2* and *3*. Thus, the total number of messages transmitted per round is (*K*-1), *K*(*K*-1), and K^2(*K*-1), for *Rounds 1, 2* and *3*, respectively. Therefore, the total number of exchanged messages is (*K*-1)+*K*(*K*-1)+K^2(*K*-1) and the message complexity for the *3ROM* algorithm is $O(K^3)$. However, if a message is indeed physically broadcast to all, e.g., when the communication means is wireless, then the number of broadcast messages per node for each round is 1. Thus, the total number of messages broadcast per round is 1, (*K*-1), and *K*(*K*-1), for *Rounds 1, 2* and *3*, respectively. Therefore, the total number of exchanged messages is 1+(*K*-1)+*K*(*K*-1) and the message complexity for the *3ROM* algorithm is $O(K^2)$. However, the message complexity for the *OM* algorithm for the above two scenarios is $O(K^F)$ and $O(K^{F-1})$, respectively. We would like to emphasize that the number of rounds of exchanged messages for the *3ROM* algorithm is independent of *F*.

V. Model Checking

In this section we present a mechanical verification of the *3ROM* algorithm using the model checking approach for its ease, feasibility, and quick examination of the problem space, to verify correctness of our formal proof of the algorithm. The Symbolic Model Verifier (SMV) was used in the modeling of this algorithm on a PC with 4GB of memory running Linux[20]. SMV's language description and modeling capability provide relatively easy translation from the pseudo-code. SMV semantics are synchronous composition, where all assignments are executed in parallel and synchronously. Thus, a single step of the resulting model corresponds to a step in each of the components.
A number of cases for each fault model were model checked. In particular, for the node-fault model, scenarios with *F* = 0..3 and *K* = 4..10, respectively, were model checked with the weaker assumptions, that is, $\sum c_j \geq F+1$ and $\sum X_i \geq F+2$. Model checking of the link-fault model requires specific number of link faults being considered. Two cases with *F* = 2, *K* = 7, and *F* = 3, *K* = 10, were model checked. Model checking of larger graphs and with more number of node and link faults can readily be accommodated. Due to space limitations we do not discuss the SMV models in detail. The models can be found online at http://shemesh.larc.nasa.gov/people/mrm/publications.htm.

A. Propositions

Computational tree logic (CTL), a temporal logic, is used to express properties of a system. In CTL formulas are composed of **path quantifiers**, *E* and *A*, and **temporal operators**, *X*, *F*, *G*, and *U*[21]. In this section the claims of agreement at the good nodes and at the end of the third round is examined. The node-fault and link-fault models are model checked separately for $F = 1, 2$, and 3, while the same CTL proposition is used to verify agreement has been reached at all good nodes for both models.

For model checking of each scenario, a particular node is instructed to be the source and scheduled to initiate broadcast of a *Sync* message at a particular time. Since the 3*ROM* is deterministic, the final vote time, *VotingResultTime*, is set to the end of the 3rd round after the broadcast of the initial *Sync* message. Validation of the CTL proposition requires examination of an underlying proposition. In particular, the variable *VoteTime* is used in these properties and is defined here.

VoteTime = (GlobalClock ≥ VotingResultTime) ;

The *GlobalClock* is a measure of elapsed time from the beginning of the operation with respect to the real time, i.e., external view. The *VoteTime* is indicative of the *GlobalClock* reaching its target value of *VotingResultTime* and the *GlobalAgreement* is defined as the conjunction of voting results at all good nodes.

Proposition *SystemLiveness*: *AF (VoteTime)*

This property addresses the liveness property of the system and whether time advances and the amount of time elapsed, *VoteTime*, has advanced beyond the broadcast of the message and the three rounds to reach agreement on that message.

Proposition *GlobalAgreement*: *AF (VoteTime* & *GlobalAgreement)*

This proposition encompasses the criteria for the agreement property as well as the claim of determinism. The proposition specifies whether or not the system will reach agreement in the three rounds after the message was initially broadcast. This property is expected to hold.

The model checking results of the bounded model of the algorithm have verified the correctness of the algorithm for fully connected networks with $K \geq 3F + 1$ nodes, for both node-fault and link-fault models, and for the following scenarios; $F = 0, 1, 2, 3$ simultaneous faults. In addition, the results have confirmed the claims of determinism and independence of the algorithm from *F*.

VI. Conclusion

Distributed systems have become an integral part of safety-critical computing applications, necessitating system designs that incorporate complex fault-tolerant resource management functions to provide globally coordinated operations with ultra-reliability. As a result, robust clock synchronization has become a required fundamental component of fault-tolerant safety-critical distributed systems. The main issue in solving the clock synchronization problem for the general case is a lack of symmetric view in the system at the participating good nodes. We first enumerated several ways of achieving message symmetry across the system, and then presented an alternative, referred to as *3ROM* algorithm, that guarantees agreement in a system in three rounds provided that $K \geq 3F+1$ nodes, each faulty node induces no more than *F* faults and each good node experience no more than *F* faults. The algorithm is based on the Oral Message algorithm of Lamport[5], Shostak, and Pease, is scalable with respect to the number of nodes in the system, and applies equally to traditional node-fault model as well as the link-fault models. The *3ROM* is independent of the fault model (node-fault or link-fault model), and is independent of the number of faults (in terms of number of required rounds, not the amount of messages), and has a message complexity of $O(K^3)$. We also presented a mechanical verification of the algorithm for up to three simultaneous Byzantine faults. The model-checking effort was focused on verifying the correctness of a bounded model of the algorithm as well as confirming claims of determinism. The underlying topology in this paper was a fully connected graph and we leave the generalization of our solution to other topologies, including an arbitrary graph that meets the minimum requirements of number of nodes and connectivity, to future works.

References

[1]Kopetz, H: *Real-Time Systems*, Design Principles for Distributed Embedded Applications, Kluwar Academic Publishers, ISBN 0-7923-9894-7, 1997.

[2] Torres-Pomales, W; Malekpour, M.R.; Miner, P.S.: “ROBUS-2: A fault-tolerant broadcast communication system,” NASA/TM-2005-213540, pp. 201, March 2005.

[3] Torres-Pomales, W.; Malekpour, M.R.; Miner, P.S.: “Design of the Protocol Processor for the ROBUS-2 Communication System,” NASA/TM-2005-213934, pp. 252, November 2005.

[4] Butler, R.: “A primer on architectural level fault tolerance,” NASA/TM-2008-215108, February 2008.

[5] Lamport, L.; Shostak, R.; Pease, M.: “The Byzantine General Problem,” *ACM Transactions on Programming Languages and Systems*, 4(3), pp. 382-401, July 1982.

[6] Driscoll, K.; Hall, B.; Sivencrona, H.; Zumsteg, P.:, “Byzantine Fault Tolerance, from Theory to Reality,” *LNCS, 22nd International Conference on Computer Safety, Reliability and Security*, pp. 235-248, September 2003.

[7] Lamport, L.; Melliar-Smith, P.M.: “Synchronizing clocks in the presence of faults,” *J. ACM*, vol. 32, no. 1, pp. 52-78, 1985.

[8] Schmid, U.; Weiss, B.; Keidar, I.: “Impossibility Results And Lower Bounds For Consensus Under Link Failures,” *in SIAM Journal on Computing 38(5)*, p. 1912-1951, January 2009.

[9] Hoyme, K.; Driscoll, K.: “SAFEbusTM,” *11th AIAA/IEEE Digital Avionics Systems Conference*, pages 68–73, October 1992.

[10] Aeronautical Radio, Inc., Annapolis, MD. “ARINC Specification 659: Backplane Data Bus,” December 1993. Prepared by the Airlines Electronic Engineering Committee.

[11] Kopetz, H.; Grünsteidl, G.: “TTP – a time-triggered protocol for fault-tolerant real-time systems,” *Fault Tolerant Computing Symposium 23*, pages 524–533, June 1993.

[12] Bauer, G.; Kopetz, H.; and Steiner, W.: “The central guardian approach to enforce fault isolation in a time-triggered system,” *Proc. of 6th International Symposium on Autonomous Decentralized Systems (ISADS 2003)*, pp. 37–44, April 2003.

[13] Malekpour, M.R.: “A Self-Stabilizing Hybrid-Fault Tolerant Synchronization Protocol,” NASA/TM-2014-218285, July 2014.

[14] Pease, M.; Shostak, R.; and Lamport, l.: “Reaching agreement in the presence of faults,” *Journal of the ACM*, 27(2): 228-234, April 1980.

[15] Lincoln, P.; Rushby J.: “A Formally Verified Algorithm for Interactive Consistency Under a Hybrid Fault Model,” *Proceedings of the Fault-Tolerant Computing Symposium*, FTCS 23, June 1993.

[16] Miner, P.S.; Geser, A.; Pike, L.; Maddalon, J: *A Unified Fault-tolerance Protocol*, In Yassine Lakhnech and Sergio Yovine, editors, Formal Techniques, Modeling and Analysis of Timed and Fault-Tolerant Systems, volume 3253, pp. 167-182, Springer, 2004.

[17] Malekpour, M.R.: *A Self-Stabilizing Byzantine-Fault-Tolerant Clock Synchronization Protocol*, NASA/TM-2009-215758, June 2009.

[18] Schmid, U.; Weiss, B.; Rushby, J.: *Formally Verified Byzantine Agreement in Presence of Link Faults*, in 22nd International Conference on Distributed Computing Systems (ICDCS’02), pp. 608–616, July 2002.

[19] Dolev, D.; Halpern, J.Y.; Strong, R.: *On the Possibility and Impossibility of Achieving Clock Synchronization*, proceedings of the 16th Annual ACM STOC, pp. 504-511, 1984.

[20] http://www-2.cs.cmu.edu/~modelcheck/smv.html

[21] Clarke, E.M.; Emerson, E.A.: *Design and synthesis of synchronization skeletons using branching time temporal logic*, In Logic of Programs: Workshop, LNCS 131, Springer, May 1981.