Degradable Agreement in the Presence of Byzantine Faults

Nitin H. Vaidya

Technical Report # 92-020

Abstract

Consider a system consisting of a sender that wants to send a value to certain receivers. Byzantine agreement protocols [7, 6, 2] have been proposed to achieve this in the presence of arbitrary failures. The imposed requirement typically is that the fault-free receivers must all agree on the same value [7, 6]. (Dolev [2] analyzes a seemingly weaker form of agreement). It has been shown that such an agreement is impossible if a third or more of the nodes are faulty [7, 6, 2].

We propose an agreement protocol that achieves Lamport's Byzantine agreement [7] up to a certain number of faults and a degraded form of agreement with a higher number of faults. Essentially, the degraded form of agreement allows the fault-free receivers to agree on at most two different values one of which is necessarily the default value. The default value is distinguishable from all other values. The proposed approach is named "degradable agreement". Specifically, m/u-degradable agreement is defined using two parameters, m and u, and the following four conditions. (The term *node* refers to the sender and the receivers).

(1) If the sender is fault-free and at most m nodes are faulty, then all the fault-free nodes must agree on the sender's value.

(2) If the sender is faulty, and the number of faulty nodes is at most m, then all the fault-free nodes must agree on an identical value.

(3) If the sender is fault-free, and the number of faulty nodes is more than m but at most u, then the fault-free nodes may be partitioned into at most two classes. The fault-free nodes in one of the classes must agree on the sender's value, and the fault-free nodes in the other class must all agree on the default value.

(4) If the sender is faulty, and the number of faulty nodes is more than m but at most u, then the fault-free nodes may be partitioned into at most two classes. The fault-free nodes in one of the classes must agree on the default value, and the fault-free nodes in the other class must all agree on an identical value.

It is shown that at least $2m+u+1$ nodes and network connectivity of at least $m+u+1$ are necessary to achieve m/u-degradable agreement. An m/u-degradable agreement algorithm is presented for more than $2m+u$ nodes. Conditions (3) and (4) imply that, up to u faults, at least $m+1$ fault-free nodes are guaranteed to agree on the same value.

1 Introduction

Consider a system consisting of a sender that wants to send a value to certain receivers. Byzantine agreement (weak [6] or otherwise [7]) and Crusader agreement [2] protocols have been proposed to achieve this in the presence of arbitrary (possibly malicious) failures. The requirement is typically that the fault-free receivers must all agree on the same value [7, 6]. (Dolev [2] analyzes a seemingly weaker form of agreement). Prior work has shown that such agreements are impossible if a third of the nodes (or more) are faulty. This report also assumes the arbitrary failure model which is also known as the Byzantine failure model.

We propose an agreement protocol that achieves Lamport's Byzantine agreement [7] up to a certain number of failures and a degraded form of agreement with a higher number of faults. Essentially, the degraded form of agreement allows the fault-free receivers to agree on at most two different values one of which is necessarily the default value.[1] This is a degraded form as compared to Byzantine agreement [7] which requires all the fault-free receivers to agree on a single value. The proposed approach is named "degradable agreement". The next section presents a definition of degradable agreement.

This report shows that degradable agreement is of interest in practice. Also, it is shown that degradable agreement provides an ability to achieve forward recovery as well as backward recovery when the number of failures is large (more than a third of the nodes may be faulty).

For the sake of simplicity, this report draws on and extends the concepts presented in two well-known report by Lamport et al. [7] and Dolev [2].

Section 2 defines the proposed degradable agreement approach. Section 3 motivates the proposed approach and discusses an application. An algorithm for achieving the proposed agreement is presented in Section 4. Lower bounds on the number of nodes and connectivity for the proposed form of agreement are presented in Section 5. The problem addressed in this report suggests a degradable clock synchronization approach. Section 6 discusses the issue of clock synchronization. Section 7 summarizes the results.

[1] Default value, denoted V_d, is distinguishable from all other values.

2 Degradable Agreement Protocol

The system model can be described as follows. The system consists of a sender and some receivers. The sender wants to send its value to the receivers. In the following, the term *node* may refer to the sender or a receiver. A faulty node (sender or receiver) may demonstrate arbitrary behavior. V_d denotes the default value. The default value V_d is assumed to be distinguishable from all other relevant values.

Degradable agreement is defined using two parameters, m and u. Degradable agreement defined by parameters m and u is hereafter called m/u-degradable agreement. An m/u-degradable agreement protocol satisfies the following conditions, where f is the number of faulty nodes.

m/u-Degradable Agreement:

- if $f \leq m$, then conditions D.1 and D.2 below must be satisfied.
- if $m < f \leq u$, then conditions D.3 and D.4 below must be satisfied.

(D.1) If the sender is fault-free, then all the fault-free receivers must agree on the sender's value.

(D.2) If the sender is faulty, then the fault-free receivers must agree on an identical value.

(D.3) If the sender is fault-free, then the fault-free receivers may be partitioned into at most two classes. The fault-free receivers in one class must agree on the sender's value, and the fault-free receivers in the other class must all agree on the default value.

(D.4) If the sender is faulty, then the fault-free receivers may be partitioned into at most two classes. The fault-free receivers in one class must agree on the default value, and the fault-free receivers in the other class must all agree on an identical value.

Conditions D.1 and D.2 are identical to those satisfied by Lamport's Byzantine agreement [7]. Conditions D.3 and D.4 define degraded agreement and are applied in fault situa-

tions with more than m but at most u faults. Thus, when $m = u$, degradable agreement is equivalent to Lamport's Byzantine agreement.

Let N be the number of nodes in the system. Observe that, if $N > 2m + u$ then m/u-degradable agreement ensures (by conditions D.3 and D.4) that at least $m+1$ fault-free nodes (including the sender) agree on an identical value, even when the number of faults is more than m (but at most u). Thus graceful degradation can be achieved. Note that graceful degradation is possible up to u faults even when $u \geq N/3$ only if we insist on achieving Byzantine agreement only up to m faults for some $m < \left\lfloor \frac{N-1}{3} \right\rfloor$. In other words, the capability to achieve Byzantine agreement can be traded with the capability to achieve degraded agreement up to a larger number of faults.

It is later proved that to achieve m/u-degradable agreement the system must consist of at least $2m + u + 1$ nodes (including the sender), and also that $2m + u + 1$ nodes are sufficient. Therefore, given a system consisting of 7 nodes, one may achieve one of the following:

- 2/2-degradable agreement, or
- 1/4-degradable agreement, or
- 0/6-degradable agreement.

This illustrates the trade-off between Byzantine agreement and degraded agreement. The following table lists the minimum number of nodes necessary for different values of m and u.

m \ u	1	2	3	4	5
1	4	5	6	7	8
2	–	7	8	9	10
3	–	–	10	11	12

Bhandari [1] looks at algorithms which correctly achieve interactive consistency [9] up to $\frac{N-1}{3}$ faults in an N node system. Bhandari proves that graceful degradations is

impossible with such algorithms if the number of faults is more than $N/3$. His result is not applicable to m/u-degradable agreement when $m < \frac{N-1}{3}$. Although Bhandari's result seems to contradict our results when $m < u \leq m+2$ and $N = 2m+u+1$, it should be noted that his result applies to interactive consistency and not Byzantine agreement (even though the two problems are closely related). Interactive consistency requires each node to agree on a vector of N values containing one value sent by each node in the system [9].

Proof of correctness for the m/u-degradable agreement algorithm presented in this report assumes that the clocks on all the fault-free nodes are synchronized. It is known that if a third (or more) of the clocks are faulty, it is not possible to achieve clock synchronization [3, 5]. Section 6 discusses this issue.

3 Motivation: Forward and Backward Recovery

Consider a fault tolerant system consisting of multiple computation channels. Figure 1(a) illustrates a system with three channels. Byzantine agreement is useful in such systems to distribute information from a single sender (for example, a sensor) to all the channels [11]. The three channels in Figure 1(a) obtain their input from the sensor and then perform

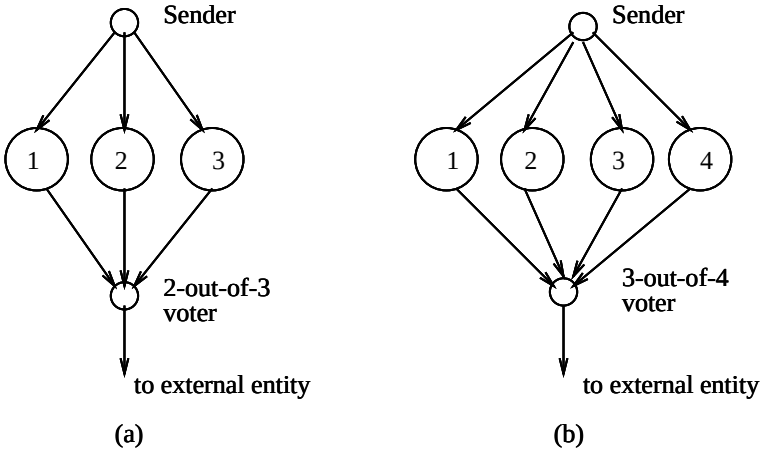

Figure 1: Multiple channel systems

computations on that input. Eventually, the output of the three channels must be sent to

an external entity (for example, to a controller). The external entity takes a majority vote on the output of the three channels and determines the correct value. It is clear that if the sender is itself faulty, the external entity may not be able to obtain the correct value. Thus, in such a system, Byzantine agreement [7] ensures the following conditions:

(B.1) Given is a system with $3m$ channels and 1 sender. If the sender is fault-free and at most m channels are faulty, then the external entity obtains the correct value using majority vote.

(B.2) All the fault-free channels are in an identical state, up to m faults.

Although the proposed approach is useful when multiple senders measure the same quantity and send its value to the channels, the discussion in this report is limited to a single sender. The three-channel system in the above example may fail, if two of the channels obtained the same incorrect value from the sender. This could happen if two nodes out of four (three channels and one sender) are faulty, as Byzantine agreement with four nodes only tolerates one fault. In general, if more than m faults occur, Byzantine agreement may result in the external entity using an incorrect output, even if the sender is fault-free. However, Byzantine agreement tolerates up to m faults, meaning that forward recovery [10] can be performed in the presence of up to m faults.

The concept of a default value is pertinent to the discussion below. If the external entity obtains a default value from the multiple channel system, it can take a "default" action which usually results in a safe operation. Another possibility is to re-do the computation (backward recovery [10]).

Degradable agreement improves the ability to survive more than m faults. Obviously, achieving this requires more resources, but we show that the increase in resource requirements is minimal. Consider a four channel system shown in Figure 1(b). For this system, $m = 1$ and $u = 2$. Using the proposed degradable agreement approach the following conditions can be ensured as compared to those listed above for Byzantine agreement.

(C.1) Given is a system with $(2m + u)$ channels and 1 sender ($m \leq u$). If the sender is fault-free and at most m channels are faulty, then the external entity obtains the correct

value using $(m+u)$-out-of-$(2m+u)$ vote[2] on the outputs of the $2m+u$ channels.

(C.2) If the sender is fault-free and more than m but at most u channels are faulty, then the external entity obtains either the correct value or the default value.

(C.3) The fault-free channels are all in an identical state if number of faults is at most m. Also, up to u faults, the fault-free channels are divided into at most two classes; the channels in one class are in a "default" state (i.e. a safe state).

It is clear that in many situations, it is safer to use the default value as compared to an incorrect value. For instance, if a controller in a fly-by-wire system receives a default value from the computer, as a safety precaution it can inform the pilot of the problem.

Condition C.2 results in a correct or default output even when more than a third of the channels may be faulty. Condition C.1 is essentially the same as B.1. Thus, by condition C.2, the degradable agreement approach improves the ability to survive a larger number of faults. Also, conditions C.2 and C.3 ensure that the state of the fault-free channels diverges gracefully.

The above discussion motivates the proposed degradable agreement approach. The proposed approach, in general, improves the safety of the system and also improves the ability of the system to perform backward recovery in the presence of more than m faults. Similar to Byzantine agreement, degradable agreement can perform forward recovery up to m faults.

4 An Algorithm for m/u-Degradable Agreement

This section presents an algorithm to prove that m/u-degradable agreement can be achieved with more than $(2m+u)$ nodes. No attempt is made here to present an efficient algorithm.

It is assumed that V_d is a default value that is distinguishable from other values. Define VOTE(μ, ν) of ν values $w_1, w_2, \cdots w_\nu$ as α if at least μ of the ν values are equal

[2] $(m+u)$-out-of-$(2m+u)$ vote of $2m+u$ values is α if $\geq (m+u)$ values are α, default value otherwise.

to α, else VOTE(μ, ν) is defined to be the default value V_d. Also, in case of a tie, define VOTE$(\mu, \nu) = V_d$. For example, VOTE(2,4) of values 1, 2, 2, 3 is 2 and VOTE(2,4) of values 1, 2, 0, 3 is V_d. VOTE(2,4) of values 1, 2, 2, 1 is V_d because of the tie.

Algorithm BYZ presented below may be viewed as an extension of an algorithm in [7]. BYZ assumes that the nodes are fully connected. Following assumptions are made regarding messages when proving correctness of algorithm BYZ: (a) All messages are delivered correctly, (b) absence of a message can be detected, and (c) source of a received message can be identified. As discussed in Section 6, assumption (b) can be relaxed when more than m faults exist.

Algorithm BYZ is recursive. The algorithm for $m = 0$ is omitted here. Algorithm BYZ(m, m) achieves m/u-degradable agreement given at least $2m + u + 1$ nodes. Now we present BYZ$(1, m)$ and BYZ(t, m). In these algorithms, the following notation is used. In BYZ$(1, m)$, n is the number of nodes to which algorithm BYZ$(1, m)$ is being applied. Similarly, in BYZ(t, m), n is the number of nodes to which algorithm BYZ(t, m) is being applied. N is the total number of nodes in the system. m and u are the two parameters that define the m/u-degradable agreement that we want to achieve in this system of N nodes. For BYZ(m, m), $n = N$. It is assumed that $N > 2m + u$ and $u \geq m > 0$. It can be seen that for BYZ(t, m), $n = N - m + t$.

Algorithm BYZ$(1, m)$

1. The sender sends its value to all the $(n - 1)$ receivers.

2. Each receiver broadcasts the value it received from the sender to $(n-2)$ other receivers. As there are $(n - 1)$ receivers, each receiver now has $(n - 1)$ values.

3. Each receiver uses VOTE$(n - 1 - m, n - 1)$ of these $(n - 1)$ values.

BYZ$(1, m)$ is not recursive. Lemma 2 in Section 4.1 proves some properties of algorithm BYZ$(1, m)$.

Algorithm BYZ(t, m), $1 < t \leq m$

1. The sender sends its value to all the $(n-1)$ receivers.

2. For each i, let v_i be the value receiver i received from the sender in step 1. Receiver i acts as the sender in algorithm BYZ$(t-1, m)$ to send the value v_i to each of the $(n-2)$ other receivers.

3. For receiver i, let $w_i = v_i$ and for each $j \neq i$, let w_j be the value receiver i received from receiver j in step 2 (using algorithm BYZ$(t-1, m)$). Thus, receiver i now has $n-1$ values $w_1, w_2, \cdots, w_{n-1}$. Receiver i uses VOTE$(n-1-m, n-1)$ of these $(n-1)$ values.

Algorithm BYZ(m, m) achieves m/u-degradable agreement if $N > 2m+u$, as proved below. Note that as the recursion unfolds in BYZ(m, m), the values of n and t change at each level of the recursion, however, the value of m remains fixed.

4.1 Proof of Correctness: Algorithm BYZ

The correctness of algorithm BYZ is being proved assuming that the absence of a message can be correctly detected. Therefore, the following assumes that each node always sends a message when it is supposed to. However, a faulty node may send an incorrect message. (Also see Section 6 for a related discussion). Assume that $N > 2m+u$.

Lemma 1 *When BYZ(t, m) is called with $t \geq 1, m \geq 1$, the following conditions hold: (a) $n > t+m+u$ and (b) $u < n-1-m$.*

Proof: The proof is by induction on t. Initially when BYZ(m, m) is executed, $t = m$ and $n = N > 2m+u = t+m+u$. Therefore, $n > t+m+u$. Therefore, condition (a) holds for $t = m$.

Now we assume that (a) holds for some $t \leq m$ and show that it holds for $t-1$. As (a) holds for t, we have $n > t+m+u$. BYZ$(t-1, m)$ is called in step 2 of BYZ(t, m) with

$n-1$ nodes. As $n > t+m+u$, we have $(n-1) > (t-1)+m+u$. Thus, condition (a) holds for $t-1$.

Thus, condition (a) is proved. Condition (a) implies that $u < n-t-m$. As $t \geq 1$, this implies that $u < n-1-m$. □

Lemma 2 *Let f be the number of faulty nodes in the system. If $n > 1+u+m$, then*

1. *BYZ$(1,m)$ satisfies condition D.1 if $f \leq m$.*
2. *BYZ$(1,m)$ satisfies condition D.2 if $f = 1$.*
3. *BYZ$(1,m)$ satisfies condition D.3 if $m < f \leq u$.*
4. *BYZ$(1,m)$ satisfies condition D.4 if $m = 1$ and $1 < f \leq u$.*

Proof: Let $n > 1+u+m$. f is the number of faulty nodes in the system.

Case 1: $f \leq m$ and the sender is fault-free.

Assume that the fault-free sender sends value α to the receivers in step 1 of BYZ$(1,m)$. In step 2, each fault-free receiver broadcasts α (the value received from the sender) to the other $(n-2)$ receivers. When these broadcasts are complete, each receiver will have $(n-1)$ values, of which at least $(n-1-m)$ must be α (as at least $n-1-m$ of the receivers are fault-free). Also, as $n > 1+u+m$ and $u \geq m$, we have $n-1-m > m$. Therefore, no value other than α is received from any $n-1-m$ nodes. Therefore, each fault-free node obtains α in step 3 of BYZ$(1,m)$. Thus, item 1 in the lemma is proved.

Case 2: $f = 1$ and the sender is faulty.

In this case all the receivers are fault-free. Therefore, in step 2 of BYZ$(1,m)$, each receiver must obtain the same set of $n-1$ values. This implies that in step 3, each receiver will obtain the same value using VOTE$(n-1-m, n-1)$. Thus, item 2 in the lemma is proved.

Case 3: $m < f \leq u$ and the sender is fault-free.

Assume that the sender sends value α to the receivers in step 1 of BYZ$(1, m)$. Each fault-free receiver broadcasts the value received from the sender to the other $n-2$ receivers. When these broadcasts are complete, each receiver will have $n-1$ values of which at least $n-1-u$ must be α as at least $n-1-u$ receivers are fault-free. By Lemma 1, $n-1-m > u$. Therefore, a fault-free receiver must obtain VOTE$(n-1-m, n-1)$ equal to either α or V_d. Thus, item 3 in the lemma is proved.

Case 4: $m = 1,\ 1 < f \leq u$ and the sender is faulty.

As $m = 1$, we have $n > 2 + u$. At the end of step 2, each fault-free receiver obtains a set of $n-1$ values. As the sender is faulty, at least $n-u$ receivers are fault-free. Therefore, at least $n-u$ of the $n-1$ values obtained by a fault-free receiver (in step 2) must be identical to those obtained by all the fault-free receivers.

Now, any two subsets containing $n-1-m$ $(= n-2)$ receivers out of the $n-1$ receivers must contain at least one fault-free receiver in common. (To see this note that, if there were two subsets containing at least $n-2$ receivers each with no fault-free receivers in common, then the total number of receivers must be at least $(n-2)+(n-2)-(u-1) \geq n$. But there are only $n-1$ receivers.). Therefore, if a fault-free receiver obtains VOTE$(n-2, n-1) = \beta$, $\beta \neq V_d$, then any other fault-free receiver must obtain VOTE$(n-2, n-1)$ equal to either β or V_d. Thus, item 4 in the lemma is proved. □

Condition (a) of Lemma 1 implies that the condition $n > 1 + u + m$ required for Lemma 2 to be true is satisfied by BYZ$(1, m)$.

Lemmas 3 through 6 below prove that BYZ(m, m) achieves m/u-degradable agreement. The proofs of these lemmas assume that: (i) $m > 1$, (ii) $N > 2m + u$ and (iii) $u \geq m > 0$.

Lemma 3 *For $1 \leq t \leq m$, algorithm BYZ(t, m) satisfies condition D.1 if $n > t + u + m$ and at most m nodes are faulty.*

Proof: By Lemma 1, $n > t + u + m$. Also, it is given that $u \geq m$.

The proof is by induction on t. The number of faulty nodes is at most m. Condition D.1 assumes that the sender is fault-free. Therefore, the lemma is true for $t = 1$ by Lemma 2.

We now assume that the lemma is true for BYZ$(t-1, m)$ where $2 \leq t \leq m$, and prove it for BYZ(t, m). In step 1 of BYZ(t, m), the fault-free sender sends a value, say α, to all the $(n-1)$ receivers. In step 2, each fault-free receiver acts as a sender in BYZ$(t-1, m)$ to send value α (which it received from the sender) to the other $(n-2)$ receivers. As $n > t + u + m$, $(n-1) > (t-1) + u + m$. Therefore, the induction hypothesis holds for BYZ$(t-1, m)$. Thus, at the end of step 2, every fault-free receiver gets $w_j = \alpha$ for each fault-free receiver j. Since there are at most m faulty receivers, at least $n - 1 - m$ are fault-free. Therefore, each fault-free receiver i has $w_j = \alpha$ for at least $(n - 1 - m)$ of the $n - 1$ receivers. As $m \leq u < n - 1 - m$ (by Lemma 1), this implies that each fault-free receiver obtains VOTE$(n - 1 - m, n - 1) = \alpha$. Thus, the lemma is proved for BYZ(t, m). □

Lemma 4 *For $1 \leq t \leq m$, algorithm BYZ(t, m) satisfies condition D.2 if $n > t + u + m$ and at most t nodes are faulty.*

Proof: The proof is by induction on t. By Lemma 2, BYZ$(1, m)$ satisfies condition D.2 if at most 1 node is faulty. We therefore assume that the lemma is true for BYZ$(t-1, m)$ where $2 \leq t \leq m$, and prove it for BYZ(t, m).

The number of faulty nodes is at most t. Condition D.2 assumes that the sender is faulty. Therefore, at most $(t-1)$ of the receivers are faulty. As $n > t + u + m$, $(n-1) > (t-1) + u + m$. In step 2, a receiver uses BYZ$(t-1, m)$ to send the value it received from the sender to the other $n-2$ receivers. As at most $t-1$ of the $n-1$ receivers are faulty, we can apply the induction hypothesis to conclude that BYZ$(t-1, m)$ satisfies condition D.2. As $t \leq m$, by Lemma 3, it follows that BYZ$(t-1, m)$ satisfies condition D.1 as well. Hence, any two fault-free receivers must obtain the same vector $w_1, w_2, \cdots, w_{n-1}$ and therefore must obtain the same value VOTE$(n - 1 - m, n - 1)$. Thus, the lemma is proved for BYZ(t, m). □

Lemma 5 *For $1 \leq t \leq m$, algorithm BYZ(t, m) satisfies condition D.3 if $n > t + u + m$ and more than m but at most u nodes are faulty.*

Proof: The proof is by induction on t. Condition D.3 assumes that the sender is fault-free. By Lemma 2, BYZ$(1, m)$ satisfies condition D.3 if at most u nodes are faulty. We therefore assume that the lemma is true for BYZ$(t-1, m)$ where $2 \leq t \leq m$, and prove it for BYZ(t, m).

In step 1 of BYZ(t, m), the fault-free sender sends a value, say α, to all the $(n-1)$ receivers. In step 2, each fault-free receiver applies BYZ$(t-1, m)$ with $(n-1)$ nodes. As $n > t + u + m$, $(n-1) > (t-1) + u + m$. Therefore, we can apply the induction hypothesis to conclude that every fault-free receiver gets $w_j = \alpha$ or V_d for each fault-free receiver j. Since at most u receivers are faulty, at least $n - 1 - u$ of the $n-1$ values received by any fault-free receiver must be α or V_d. Now, $u < n - 1 - m$ by Lemma 1. Therefore, each fault-free receiver must obtain VOTE$(n-1-m, n-1)$ equal to either α or V_d. Thus, the lemma is proved for BYZ(t, m). □

Lemma 6 *Algorithm BYZ(m, m) satisfies condition D.4 if more than m but at most u nodes are faulty and $n > 2m + u$.*

Proof: Condition D.4 assumes that the number of faulty nodes is at most u and the sender is faulty. Therefore, at most $(u-1)$ of the receivers are faulty. For BYZ(m, m), $n = N > 2m + u$.

In step 2 of BYZ(m, m), each fault-free receiver sends the value it received (from the sender) to the other $n-2$ receivers using BYZ$(m-1, m)$. As $n > 2m+u$, $(n-1) > (m-1) + u + m$. Therefore, by Lemma 5, we know that BYZ$(m-1, m)$ satisfies condition D.3. As at most $u-1$ receivers are faulty, at least $n-u$ are fault-free. Without loss of generality, assume that receivers 1 through $n-u$ are fault-free. v_j is the value fault-free receiver j received from the sender in step 1 of BYZ(m, m). Therefore, at the end of step 2, each fault-free receiver must obtain a vector of $n-1$ values of the form $(v_1/V_d, v_2/V_d, \cdots, v_{n-u}/V_d, X, \cdots, X)$, where

v_j/V_d indicates that the corresponding value is either v_j or V_d, and X denotes a value that is not relevant in our discussion here.

Now, suppose that in step 3 of BYZ(m, m), a fault-free receiver i obtains VOTE$(n-1-m, n-1) = \beta$ where $\beta \neq V_d$. This implies that, for fault-free receiver i, at least $(n-1-m)-(u-1) = n-m-u$ of the first $n-u$ values in the vector of $n-1$ values must be β. As $n = N$, $n-m-u \geq m+1$. This implies that at least $m+1$ fault-free receivers received β from the sender in step 1. Therefore, each fault-free receiver must have (at the end of step 2 of BYZ(m, m)) at least $m+1$ values that are β or V_d. Thus, VOTE$(n-1-m, n-1)$ cannot be any value other than β and V_d. Thus, the lemma is proved. □

Theorem 1 *BYZ(m, m) achieves m/u-degradable agreement if $N > 2m + u$.*

Proof: For $m = 1$, the proof follows from Lemma 2. For $m > 1$, the proof follows from Lemma 1 and from Lemmas 3 through 6 by choosing $t = m$. □

5 Lower Bounds for Degradable Agreement

This section presents the lower bounds on the number of nodes and the network connectivity necessary to achieve m/u-degradable agreement. For future reference note that, by definition, a system that achieves m/u-degradable agreement also achieves Byzantine agreement [7] up to m faults.

Theorem 2 *Given N nodes, m/u-degradable agreement can be achieved only if $N > 2m+u$.*

Proof: The proof is in two parts. Part I proves that 1/2-degradable agreement is impossible with less than 5 nodes. Part II extends this result to prove the theorem in general.

Part I: It is clear that the number of nodes must be at least 4, else Byzantine agreement with 1 fault cannot be achieved. Therefore, assume that 1/2-degradable agreement can be achieved in a system of 4 nodes. Let the nodes be named S, A, B and C, where S is the sender

node. Figures 2(a) through (c) illustrate three fault scenarios; the shaded nodes are faulty. In the following, let α and β be two different values distinct from V_d, i.e., $V_d \neq \alpha \neq \beta \neq V_d$.

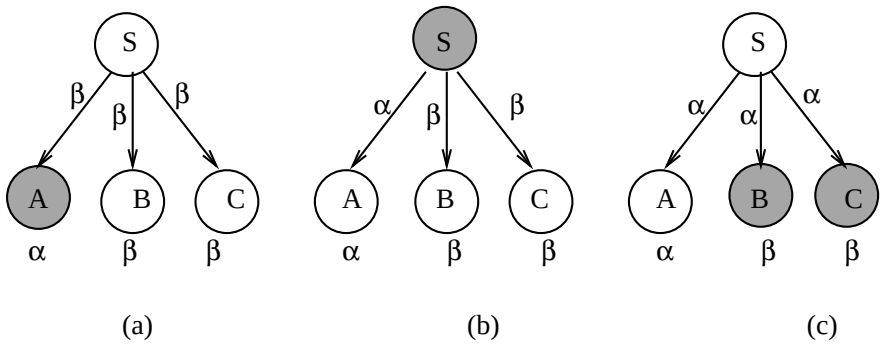

Figure 2: Proving lower bound on the number of nodes

In scenario (a), only node A is faulty. Also, in scenario (a), let the sender's value be β. Therefore, by condition D.1, fault-free nodes B and C must agree on β. The arcs in Figure 2(a) indicate the values sent by S to the other nodes. Value β written below node B (C) in the figure indicates that node B (C) may tell the other receivers that the sender sent him β. Value α written below node A indicates that node A pretends to have received α from sender S. As node A is faulty, such a behavior is possible.

In scenario (b), only sender node S is faulty. Also, the faulty sender S sends α to node A and β to nodes B and C, as indicated in Figure 2(b). As node S is faulty, such a behavior is possible. Now, fault-free node B cannot distinguish between scenarios (a) and (b). As seen above, in scenario (a), node B agrees on β. Hence, in scenario (b) also node B must agree on β. Therefore, by condition D.2, nodes A and C too must agree on β.

In scenario (c), nodes B and C are faulty. Also, let the sender's value be α. Additionally, in scenario (c) assume that faulty nodes B and C pretend that they received value β from the sender, as illustrated in Figure 2(c). Now, fault-free node A cannot distinguish between scenarios (b) and (c). We have already concluded that node A agrees on β in scenario (b). Therefore, A must agree on β in scenario (c) as well. However, condition D.3 requires that fault-free node A should agree on either α or V_d in scenario (c). As $\alpha \neq \beta \neq V_d$, this is a contradiction.

Thus, 1/2-degradable agreement cannot be achieved with less than 5 nodes.

Part II: The proof in Part II is similar to a proof in [7]. It is clear that to achieve m/u-degradable agreement at least $3m + 1$ nodes are necessary (otherwise Byzantine agreement cannot be achieved for m faults). Therefore, consider a system consisting of N nodes such that $N = 3m + \mu$ where $1 \leq \mu \leq (u - m)$. Thus, $N \leq 2m + u$. Also, $m + \mu \leq u$. Assume that m/u-degradable agreement can be achieved in this system.

Divide the $3m+\mu$ nodes into four groups such that three groups contain m nodes each and the fourth group contains μ nodes. Name the three groups containing m nodes each as S_m, A_m and B_m, where S_m is the group that contains the sender node. The group containing μ nodes is named C_μ. Now, assume that the four node system in Figure 2 simulates the $3m + \mu$ node system; nodes S, A, B and C in Figure 2 simulate the nodes in S_m, A_m, B_m and C_μ, respectively. Node A agrees on value α only if all the nodes in A_m agree on value α (similarly for S, B and C). Also, if node A is faulty, then all the nodes in A_m are faulty (similarly for S, B and C). Now, as the $3m + \mu$ node system can achieve m/u-agreement, it should be able to reach correct agreement in all the three fault scenarios corresponding to those illustrated in Figure 2. But that is an impossibility as shown in Part I. □

As proved by the correctness of algorithm BYZ, $2m + u + 1$ nodes are not only necessary but also sufficient to achieve m/u-degradable agreement.

Theorem 3 *Given N nodes, m/u-degradable agreement can be achieved only if the network connectivity is at least $m + u + 1$.*

Proof: Instead of a formal proof, a sketch is presented. This sketch is similar to the proof of the lower bound for Crusader agreement in [2].

Let G be the network of nodes under consideration. Let the connectivity of G be $\kappa \leq m + u$. It is clear that κ must be at least $2m + 1$, otherwise even Byzantine agreement is not possible with m faults. Thus, $2m < \kappa \leq m + u$. Assume that m/u-degradable agreement can be achieved in this system.

Consider the following scenario. Let $F = \{a_1, a_2, \cdots, a_\kappa\}$ be a set of nodes which can disconnect the network into two non-empty parts G_1 and G_2. Let $F_1 = \{a_1, \cdots, a_m\}$ and $F_2 = \{a_{m+1}, \cdots, a_\kappa\}$. Thus, $F_1 \cup F_2 = F$ and $F_1 \cap F_2 = \emptyset$. Also $|F_1| = m$ and $m < |F_2| \leq u$.

Assume that the sender is in G_1. Let the sender be fault-free and its value be α. Let the nodes in F_1 be all faulty and let all other nodes be fault-free. The faulty nodes in F_1 change each message from G_1 to G_2 to carry value β (where $\alpha \neq \beta \neq V_d$) and change each other message to carry value α. Now, by condition D.1, m/u-degradable agreement requires that in this fault scenario, the nodes in G_2 must agree on α. Now, the nodes in G_2 cannot differentiate between the above scenario from the following scenario: All the nodes in G_1 including the sender are fault-free, the sender's value is β, and the nodes in F_2 are faulty. In this second fault scenario, as $m < |F_2| \leq u$, by condition D.3, the nodes in G_2 must agree on either β or V_d. Thus, the two scenarios require the nodes in G_2 to agree on two different values, causing a contradiction. □

It turns out that connectivity of $m + u + 1$ is not only necessary but also sufficient to achieve m/u-degradable agreement in a system of more than $2m + u$ nodes.

6 Message Passing and Clock Synchronization

The most critical implementation issue is that of clock synchronization. In order to ensure that the absence of a message can be detected, it is necessary to synchronize the clocks of all the fault-free nodes. However, it has been shown that clock synchronization cannot be achieved if a third (or more) clocks are faulty [3, 5]. When using m/u-degradable agreement, u may be larger than a third of the number of nodes. Thus, clock synchronization cannot be guaranteed *if* a node being faulty necessarily implies that its clock is faulty as well. There are two approaches to deal with this problem, as discussed in the following two subsections.

6.1 Degradable Clock Synchronization

For brevity, the proof of correctness for algorithm BYZ assumed that the absence (and presence) of messages can be correctly detected up to u faults. However, it is possible to

prove the correctness of algorithm BYZ under the following relaxed conditions as well.

1. When at most m nodes are faulty, absence and presence of messages is detected correctly. (This requires clock synchronization).

2. When more than m but at most u nodes are faulty, a fault-free node may incorrectly declare a message from another fault-free node to be absent. (This may happen due to time-outs).

The above conditions can be satisfied provided the clocks of fault-free nodes are synchronized up to m faults. This is achievable, as $\frac{N}{3} > \frac{2m+u}{3} \geq m$. The above implies correctness of the algorithm in the sense that m/u-degradable agreement is achieved whenever the algorithm terminates. However, in the absence of clock synchronization with more than m faults, the algorithm is not (yet) guaranteed to complete within bounded time.

Termination within bounded time can be guaranteed with more than m faults if one of the following is true: (i) at least $m+1$ fault free nodes have their clocks synchronized, or (ii) at least $m+1$ fault-free nodes detect the presence of more than m faulty nodes. Keeping these conditions in mind, we introduce the following problem.

m/u-Degradable Clock Synchronization:

1. if at most m clocks are faulty, all the fault-free clocks must be synchronized and should "approximate" the real time.

2. if more than m but at most u clocks are faulty then, either

   - at least $m+1$ fault-free clocks must be synchronized and should approximate the real time, or
   - at least $m+1$ fault-free clocks should detect the existence of more than m faulty clocks.

We conjecture that m/u-degradable clock synchronization can be achieved with more than $2m+u$ clocks. Motivation for this conjecture is the observation in Section 2 that, given

$2m + u + 1$ nodes, if at most u nodes are faulty, at least $m + 1$ fault-free nodes agree on the same value using m/u-degradable agreement. This conjecture is being investigated currently. The reader can verify that the impossibility result in [5] does not as such apply to the above problem.

6.2 Traditional Clock Synchronization

One solution, applicable to systems such as FTMP, NETS [11] and FTP [4], is the use of hardware clock synchronization (as opposed to software algorithms). In typical systems, the complexity and cost of clock hardware is orders of magnitude lower as compared to the processor (or node) complexity. Therefore, the failure rates for clock hardware are likely to be significantly lower. Therefore, one may assume that at most a third of the clocks may fail (although more than a third of the processors may fail). In such a case the term "node failure" in our discussion refers to the failure of the processor. Essentially, a processor being faulty does not necessarily imply that the associated clock hardware is faulty as well. For example, in Figure 1(b), we may assume that at most one clock may be faulty, while using 1/2-degradable agreement for the processors. Such an assumption is not likely to affect the overall system reliability or safety adversely as the clock failure rates are much lower than processor failure rates. Also, as clock complexity is lower, it is possible to use conservative designs to further lower the failure rates.

Another approach is to use more clocks than the number of processors. For example, for the system in Figure 1(b), one may use two more clocks in addition to those associateed with each of the nodes, making the system capable of tolerating two clock failures. Addition of clocks can be used to make the system capable of tolerating up to μ clock failures, for some μ, $m < \mu \leq u$. This idea is analogous to the concept of witnesses proposed for maintaining consistency in replicated file systems [8].

7 Summary

m/u-degradable agreement protocol that achieves Lamport's Byzantine agreement [7] up to m faults, and a degraded form of agreement with more than m but at most u faults is proposed. Up to m faults, all the fault-free nodes agree on an identical value. For more than m faults (up to u faults), the degraded form of agreement allows the fault-free nodes to agree on at most two different values one of which is necessarily the default value; the other value is the sender's value if the sender is fault-free. It is shown that at least $2m + u + 1$ nodes and network connectivity of at least $m+u+1$ are necessary to achieve m/u-degradable agreement. Also, an m/u-degradable agreement algorithm is presented for more than $2m+u$ nodes.

The results presented in this report suggest that degradable agreement is a cost-effective approach for tolerating a small number of Byzantine failures using forward recovery and a large number of failures using backward recovery. Further research is required to explore the applications of degradable agreement.

The report also formulates the problem of degradable clock synchronization. This problem is under investigation presently.

References

[1] I. S. Bhandari, "On the graceful degradation of byzantine agreement," Master's thesis, University of Massachusetts-Amherst, September 1985.

[2] D. Dolev, "The Byzantine generals strike again," *J. Algo.*, pp. 14–30, 1982.

[3] D. Dolev, J. Y. Halpern, and H. R. Strong, "On the possibility and impossibility of achieving clock synchronization," *J. Computer and System Sciences*, vol. 32, pp. 230–250, 1986.

[4] R. E. Harper, J. H. Lala, and J. J. Deyst, "Fault tolerant parallel processor architecture overview," in *Digest of papers: The* 18^{th} *Int. Symp. Fault-Tolerant Comp.*, pp. 252–257, 1988.

[5] C. M. Krishna and I. S. Bhandari, "On graceful degradation of phase locked clocks," in *IEEE Real-Time Systems Symposium*, pp. 202–211, 1988.

[6] L. Lamport, "The weak Byzantine generals problem," *J. ACM*, vol. 30, pp. 668–676, July 1983.

[7] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Prog. Lang. Syst.*, vol. 4, pp. 382–401, July 1982.

[8] J.-F. Paris, "Voting with witnesses: A consistency scheme for replicated files," in *International Conf. Distributed Computing Systems*, pp. 606–612, 1986.

[9] M. Pease, R. Shostak, and L. Lamport, "Reaching agreement in the presence of faults," *J. ACM*, pp. 228–234, April 1980.

[10] D. P. Siewiorek and R. S. Swarz, *The Theory and Practice of Reliable System Design.* Digital Press, Bedford, MA, 1982.

[11] T. B. Smith III et al., *The Fault-Tolerant Multiprocessor Computer.* Park Ridge, NJ: Noyes Publications, 1986.