Approximate Agreement with Mixed Mode Faults: Algorithm and Lower Bound

Richard Plunkett, Alan Fekete

University of Sydney, Australia

Abstract. Approximate agreement is a building block for fault-tolerant distributed systems. It is a formalisation for the basic operation of choosing a single real value (representing say speed) for use in later computation, reflecting the different approximations to this value reported from a number of possibly-faulty processors or sensors. We study the approximate agreement problem in distributed systems where processor failures are characterised depending on their severity. We develope a new algorithm that can tolerate up to b byzantine faults, s symmetric ones, and o send-omission faults. We analyse the convergence attained by this algorithm, and also give a universal bound on the convergence available to any algorithm no matter how complicated.

1 Introduction

Fault-tolerance is an important property for distributed systems. Distribution makes fault-tolerance possible, becausse if some processors fail, there are others which can continue with the computation. Distribution also makes fault-tolerance necessary, since as we increase the number of sites involved in a computation , we increase the likelihood that some will fail during the execution.

There are many different ways in which parts of a distributed system can misbehave during a computation. In general, a fault-tolerant system will be able to produce an appropriate answer in the face of some situations, but others will be so severe that the whole computation is corrupted. To study fault-tolerance more carefully, we describe a system in terms of a *fault model*, which describes the sorts of erroneous behaviour that can be overcome. It is clear that more severe forms of behaviour will require more sophisticated algorithms to tolerate them. In the theoretical literature, the most studied failure is *byzantine*, which covers a processor changing its state and sending messages arbitrarily. Clearly any algorithm that can overcome such arbitrary faults will be very useful, since almost any imaginable situation will be covered. However, some famous impossibility results [6, 9, 7] show that byzantine failures are very difficult to overcome. Other work has proposed a model with less severe faults, called *omission faults*, in which a faulty processor may neglect to send some messages, but whenever it does send, the transmitted data is valid. This is an abstraction of the common situation where transmission errors cause identifiable corruption (detected for example by a checksum). It is common in the theoretical literature to present an algorithm that tolerates a bounded number of a certain kind of fault; for example, one might work whenever at most t omission errors occur in any execution.

Since it seems reasonable that the number of very severe errors will be low but not zero, some of the more practical researchers have examined algorithms that are designed to operate in an environment with more than one class of faults. These are described as mixed-mode fault models. For example, Thambidurai and Park [10] divided Byzantine faults as symmetric or asymmetric, depending on whether the *same* incorrect value was sent to all processors in a round of broadcast transmission; they also considered *benign* faults which are detectable by all recipients (and so correspond to universal omission).

The most famous abstract formulation of a fault-tolerance problem is *agreement* [9], in which each processor has an initial value (representing its state so far), and the processors exchange messages in an attempt to reach a consistent basis for continued computation. It has been shown that solving this, even with very restricted fault types, requires some fault-free executions to run for a long time: at least $t+1$ rounds of message exchange.

In order to see what could be done with faster response times, Dolev Lynch Pinter Stark and Weihl [2] introduced the problem of approximate agreement, where each processor has an initial real value (say representing a sensor reading), and communication is used to try to give each processor a new value which more accurately reflects the true sensor value. This is formalised by requiring that the final value at any non-faulty processor must be within the range of initial values among the non-faulty processors (the "Validity" condition). The success of such an algorithm is measured by its *convergence*, symbolised by K, defined as the ratio between the spread of final values of non-faulty processors to the spread among initial values of non-faulty processors. This definition means that a small value of K is desirable. Following [2], most work has assumed a single round of message exchange in the algorithm; if extra time is available, one can simply iterate the protocol, reducing the spread of values by a factor of K in each round.

While most of the emphasis in the literature on Approximate Agreement has been on presenting algorithms and analysing their convergence, it is also crucial to understand the theoretical limitations on what any algorithm can do, no matter how cleverly written. Thus we need to find lower bounds on the values of K that are possible for different fault models.

In this paper, we use a model with three significant classes of faults. Byzantine faults are the most general, covering arbitrary changes of state and messages that are individually corrupted in arbitrary, non-detectable ways. Omission faults result in messages that are corrupted in ways that can be detected by the recipient: formally we represent this as the faulty node failing to send a message which is required by the protocol. Symmetric faults are not detectable, but act identically on all the messages sent by the faulty node in a round of communication. (The fourth possibility, of messages corrupted in identical detectable ways, is what has previously been called a benign fault. Since it is equivalent to a node that never does anything, we will simply ignore such nodes from the whole problem.) The justification for this model comes from separating the processor into two logical sections from which faults can emerge. The numerical calculations and data source being one section and the communication unit being the other. A fault in the numerical unit will provide an incorrect value to the communication unit which it may the distribute correctly. These are symmetric faults. Similarly the numerical unit may provide a correct value, but have the communications unit fail to send one or more of the required messages. These are omissive faults. A byzantine fault is usually the failure of both, or a severe failure (undetectably) by the communications unit.

We give an algorithm for approximate agreement that works among N processors, whenever there are at most b byzantine faults, at most o omission ones, and at most s symmetric faults. We analyse the convergence of this algorithm, and show that

$$K \leq \left(\left\lceil \frac{N-2b-2s-o}{b+o/2} \right\rceil \right)^{-1}$$

We also prove that every algorithm for this problem (involving a single round of message exchange) must attain convergence that satisfies

$$K \geq \left(2 \left\lceil \frac{N-2b-2s-o}{2b+o} \right\rceil \right)^{-1}$$

As can be seen our algorithm comes asymptotically close to this bound.

There have been five main papers dealing with approximate agreement. The problem was defined in [2], which gives an algorithm for systems with synchronous (time-constrained) communication and at most b byzantine failures. This algorithm is found to have performance $K \leq \left(\left\lceil \frac{N-2b}{b} \right\rceil \right)^{-1}$ and a matching lower bound is proved. Fekete [4] examines the problem for systems with only omission failures, and gives an algorithm with $K \leq \frac{o}{2N-2o}$; he also proves a lower bound $K \geq \frac{o}{2N+3o}$ Kieckhafer and Azadmanesh [8] use the model with byzantine and symmetric faults, and give an algorithm with $K \leq (\lceil \frac{N-2b-2s}{b} \rceil)^{-1}$.

The problem has also been considered for systems with asynchronous (unlimited delay) communication. Algorithms are found for byzantine faults [2], omission faults [5] and mixed byzantine and symmetric faults [1].

None of these papers includes a lower bound for any mixed mode model; our technique for dealing with this is a major contribution of our work. The rest of this paper is as follows. In section 2 we present the formal fault model, and also give the important facts about multisets and operations on them, used in the algorithm and its analysis. Section 3 gives the algorithm and the analysis of its convergence rate. Section 4 gives the lower bound. Section 5 concludes.

2 The Model and Preliminaries

2.1 Multisets and Some Associated Functions.

Approximate agreement requires the manipulation of multisets containing real values and the null value($\perp$). A multiset is a collection of values similar to a set, except multiple occurrences of the same are permitted. The number of times a value occurs in a multiset is referred to as its multiplicity.

There are several representations of these multisets. Let $\Re^+$ be the set of real number plus the null value $\perp$. A multiset may be represented as a mapping $V : \Re^+ \rightarrow N$. For each value r from $\Re^+$, $V(r)$ is defined as the multiplicity of r in V, hence $|V| = \sum_{r \in \Re^+} V(r)$.

Another useful representation of the multiset is as a monotonically increasing sequence. So $V = \langle v_0, v_1, \ldots, v_{|V|-1} \rangle$, ordered such that $v_i \leq v_{i+1}$, for convenience of notation we consider the null value to be greater than any real value. These two representations are equivalent, though each is best suited to different styles of manipulation. When we refer to the ith element of V as v_i we are using the ordered sequence representation.

There are a number of useful functions we need to define on multisets.

- $min(V)$ = the smallest non-null value in V.
- $max(V)$ = the largest non-null value in V.
- $\delta(V) = max(V) - min(V)$ is called the diametre of V.
- $mean(V) = \frac{1}{|V|} \left(\sum_{r \in \Re} V(r) \cdot r \right)$ = the arithmetic mean of the non-null values of V
- Union If $U = V \cup W$ then $\forall r \in \Re^+$ $U(r) = max(V(r), W(r))$
- Intersection If $U = V \cap W$ then $\forall r \in \Re^+$ $U(r) = min(V(r), W(r))$
- Sum If $U = V + W$ then $\forall r \in \Re^+$ $U(r) = (V(r) + W(r))$
- Difference If $U = V - W$ then $\forall r \in \Re^+$ $U(r) = max(V(r) - W(r), 0)$
- $redS(V) = V - min(V)$
- $redL(V) = V - max(V)$
- $reduce(V) = redS(redL(V)) = V$ with the largest and smallest values removed.

The function $redS$, $redL$, and $reduce$ can be applied recursively to remove more than one extremal values from V. A superscript identifies how many applications are needed. So $reduce^k(V)$ is the set that would remain if the k smallest and k largest values were removed from V.

We also use $|V|$ to represent the number of elements in V and V^* to represent the multiset of non-null values in V.

A simple approach to approximate agreement is to reduce the set of values received in the message exchange, by removing extreme values which might have come from faulty processors, and then average the remaining values. More highly tuned algorithms take elements from the reduced set at regular spacing, and average these. Formally, we define $select_d(v)$ is a subsequence of the multiset V obtained by selecting the first and every subsequent dth element of V. So for $V = \langle v_0, \ldots, v_{|V|-1} \rangle$ we have $select_d(V) = \langle v_o, v_d, v_{2d}, \ldots, v_{jd} \rangle$ where $j = \lfloor \frac{|V|-1}{d} \rfloor$.

2.2 Three Fault Model

We consider algorithms where there are N processors. In any execution, a nonfaulty processor must follow its protocol precisely. Since we are considering a single-round of communication in a synchronous communication system, this protocol involves sending the processors initial value to all others (for simplicity we include the processor itself among the destinations), and then after the period of one message delay, the processor will collect a multiset of values received in messages: if a processor does not send to it, we place the special null value $\perp$ in the multiset to represent the omission. Different protocols are described by indicating how the final value is to be calculated from the multiset of received data. This approach is called a full-information model of a protocol.

A processor with a symmetric fault will send the same possibly incorrect value to all other nodes. A processor with an omission fault may send its correct value to some destinations, and it may omit sending to others, but it can't send an incorrect value. A processor with a byzantine fault may send arbitrary messages to any nodes. Since symmetric faults represent numerical/data faults we will require they distribute a possibly incorrect real value, so any nodes which produces only $\perp$ values must be ommisively or byzantinely faulty. We will throughout assume that there are at most b byzantine faults, at most o omission faults, and at most s symmetric faults, in any execution of our algorithms.

Notice that our model subsumes those previously considered. For example, the byzantine model of [2] corresponds to $s = o = 0$, the omission model of [4] is $b = s = 0$, and the mixed-fault model of [8] is $o = 0$.

2.3 Variance

In analysing our new algorithm, we will make extensive use of a new measure of similarity between multisets, which we call *variance*. To define $variance(V, W)$ we will require V and W be two multisets containing the same number of entries and $V(\perp) = W(\perp) = 0$. A matching $P_{V,W}$ of these multisets is a pairing of each element in V with a separate element from W. This is most easily thought about using the monotonically increasing sequence representation for multisets. The function $match(P_{V,W}, v_i)$ refers to an element of the other set (w_j of W) to which v_i is matched in $P_{V,W}$. $D_=(P_{V,W})$ is the number of pairings in $P_{V,W}$ which have the element from V equal to the element from W. Similarly, $D_<(P_{V,W})$ is the number of pairings in which the element from V is smaller than the element from W and $D_>(P_{V,W})$ is the number of pairing in which the element from V is larger. We define $D(P_{V,W}) = max(D_<(P_{V,W}), D_>(P_{V,W}))$. We define the $variance(V, W)$ to be the minimum value d for which a matching $P_{V,W}$ exists that satisfies $D(P_{V,W}) \le d$. To prove $variance(V, W) \le d$, we need only provide an example matching $P_{V,W}$ for which $D(P_{V,W}) \le d$. While actual variance of two sets is difficult to find, calculating an upper limit on the possible value of the variance is easily done using the following lemmas.

These lemmas assume unless otherwise specified $|V| = |W| = |X| = m$ and $V =< v_0, ..., v_{m-1} >$, $W =< w_0, ..., w_{m-1} >$ and $X =< x_0, ..., x_{m-1} >$.

Lemma 1. *$variance(V, W) \le |V - W|$.*

Proof. Any matching $P_{V,W}$ of the elements from V to W which matches as many equal values as possible will leave only $|V - W|$ non-equal pairs, thus

$D_<(P_{V,W}) + D_>(P_{V,W}) = |V - W|$ thus $D(P_{V,W}) \leq |V - W|$ so $variance(V, W) \leq |V - W|$.

Lemma 2. *$variance(reduce^k(V), reduce^k(W)) \leq variance(V, W)$.*

Lemma 3. *$variance(V, W) \leq variance(V, X) + variance(X, W)$*

Lemma 4. *Suppose C is a multiset, and $|C| \bmod 2 = 0$. Then*

$$variance(V, reduce^{\frac{|C|}{2}}(V + C)) \leq \frac{|C|}{2}$$

Proof. Let $V' = reduce^{\frac{|C|}{2}}(V + C)$. Divide C into two sets C_1 (containing the $\frac{|C|}{2}$ smallest elements of C) and C_2 (containing the $\frac{|C|}{2}$ largest elements of C). Note V' now equals $reduce^{\frac{|C|}{2}}(V + C_1 + C_2)$. Consider the following matching $P_{V,V'}$. Match all elements of V that are retained in V' to themselves, since $reduce^{\frac{|C|}{2}}(V + C_1 + C_2)$ removes only $|C|$ elements from $V + C_1 + C_2$ at least $|V| - |C|$ elements of V are retained. Now consider the effect of the reduce function. Firstly it will remove the $\frac{|C|}{2}$ smallest elements, which must be in V and/or C_1. Let cv_1 be the number of those smallest elements found in V and cc_1 be the number found in C_1. Now we have that $cv_1 + cc_1 = \frac{|C|}{2}$ and $\frac{|C|}{2} - cc_1 = cv_1$ elements of C_1 will be found in $reduce^{\frac{|C|}{2}}(V + C_1 + C_2)$. Match those cv_1 elements from C_1 to the cv_1 elements of V which are removed as being in the $\frac{|C|}{2}$ smallest of $V + C_1 + C_2$. Each of these matching has the element in V smaller than or equal to the elements it is matched to in V' since the removed element were smaller than or equal to any of the remaining elements in V and C_1 by definition of reduce. Similarly for the top half, $\frac{|C|}{2}$ elements will be removed from either V and/or C_2, let cv_2 be the number of those elements which are from V. So cv_2 elements from C_2 will then be found in $reduce^{\frac{|C|}{2}}(V + C_1 + c2)$. Match those elements to the elements of V that were removed as being in the largest $\frac{|C|}{2}$ elements of $V + C_1 + C_2$. These matchings all have the elements from V being greater than or equal to the elements it is matched to in V' since the removed elements were greater than or equal to any of the remaining elements.

So the matching $P_{V,V'}$ has $D_<(P_{V,V'}) \leq \frac{|C|}{2}$ and $D_>(P_{V,V'}) \leq \frac{|C|}{2}$ which guarantees $variance(V, V') \leq \frac{|C|}{2}$.

Lemma 5. *If $variance(V, W) \leq d$ then $v_i \leq w_{i+d}$ and $w_i \leq v_{i+d}$.*

Proof. By contradiction, assume $v_i > w_{i+d}$. Then consider a matching $P_{V,W}$ for which $D(P_{V,W}) = variance(V, W) \leq d$. Look at which values of V the first $i + d + 1$ elements of W are matched to. At most i can be matched to elements smaller than v_i ($\langle v_0, ..., v_{i-1} \rangle$) and as such may be equal or less than the element from V they are matched to. This leaves at least $d + 1$ elements less than or equal to w_{i+d} which must be matched to elements of V greater than or equal to v_i. If $v_i > w_{i+d}$ then all of these matchings are the greater than type matchings, leaving $D_>(P_{V,W}) \geq d+1$, $D_>(P_{V,W}) > d$ for that matching. However, $D_>(P_{V,W}) \leq D(P_{V,W}) = d$ so $D_>(P_{V,W}) \not> d$ so $v_i \not> w_{i+d}$. So $v_i \leq w_{i+d}$. By a symmetric argument $w_i \leq v_{i+d}$.

Lemma 6. *If $variance(V, W) \leq d$ then*

$$mean(select_d(V)) - mean(select(W)) \leq \delta(V \cup W) \cdot \frac{1}{\lceil \frac{m}{d} \rceil}$$

Proof. The select function produces sets $X =< v_0, v_d, v_{2d}, ..., v_{jd} >$ and $Y =< w_0, w_d, w_{2d}, ..., w_{jd} >$ where $j = \lfloor \frac{m-1}{d} \rfloor$, so $|X| = |Y| = \lceil \frac{m}{d} \rceil$. Let X also be labelled $< x_0, ...x_j >$ and Y also be labelled $< y_0, ...y_j >$. From Lemma 5 we know that $v_{id} \leq w_{(i+1)d}$ and $w_{id} \leq v_{(i+1)d}$ so $x_i \leq y_{i+1}$ and $y_i \leq x_{i+1}$. Wlog assume $\sum X \geq \sum Y$. $\sum X - \sum Y = (x_0 - y_0) + (x_1 - y_1) + ... + (x_j - y_j) = -y_0 + (x_0 - y_1) + (x_1 - y_2) + ... + (x_{j-1} - y_j) + x_j$. Since all the middle terms are guaranteed to be less than or equal to zero, maximising the equation has them set to zero, leaving $\sum X - \sum Y \leq x_j - y_0 \leq \delta(V \cup W)$. Thus $mean(select(V)) - mean(select(W)) \leq \delta(V \cup W) \cdot \frac{1}{\lceil \frac{m}{d} \rceil}$

3 The New Algorithm *BSO*

Here is the algorithm, which we call BSO, to be followed by a processor m, once it has sent its initial value to all processors, and received the values they sent.

1. Collect multiset M_1 of n values, one from each processor. Processes from which values aren't received are assumed to have sent $\perp$.
2. If there are more than o elements of value $\perp$ in M_1, replace the excess elements by a default real value (eg zero). More formally, let M_2 denote the multiset defined by $M_2(\perp) = min(M_1(\perp), o)$, $M_2(0) = M_1(0) + max(0, M_1(\perp) - o)$, and $\forall x \notin \{0, \perp\} : M_2(x) = M_1(x)$.
3. Remove the highest and lowest $b+s$ values from the multiset. Formally, let $M_3 = reduce^{b+s}(M_2)$
4. Let j_m be the number of $\perp$ values in M_3 and apply the double operator to the multiset. Thus $j_m = M_3(\perp)$, and $M_4 = double(M_3)$.
5. Remove all $2j_m$ elements of value $\perp$ from the multiset. Thus $M_5(\perp) = 0$ and $\forall x \neq \perp$: $M_5(x) = M_4(x)$. For brevity, we will write $M_5 = M_4^*$ to denote this.
6. Again we remove extreme values, to make the multiset have a fixed size of $2N - 4b - 4s - 2o$. Formally, let $M_6 := reduce^{o-j_m}(M_5)$.
7. We use a tuned averaging function to collapse the remaining multiset to a single value. $Result := Mean(Select_{2b+o}(M_6))$

3.1 Analysis of Convergence

We are going to analyse the convergence of this algorithm, by taking two processors in an arbitrary execution, and calculating the difference between their final values, using intermediate analysis showing variance between the multisets produced in the steps of the calculation. To begin, we show that we can restrict our attention to executions where every processor sees at most o missing messages (represented by $\perp$ in its multiset). In essence, there is no advantage to the fault source, in making a byzantine node omit a message.

Lemma 7. *If the convergence rate of BSO is bounded above by K for all executions in which at every processor p_i $M_1(\perp) \leq o$, then its convergence rate is bounded above by K for all executions.*

Now we can prove a bound on the convergence of BSO

Theorem 8. *For any execution of BSO algorithm, we have*

$$K \leq \left(\left\lceil \frac{N - 2b - 2s - o}{b + o/2} \right\rceil \right)^{-1}$$

Proof. The following argument on the convergence of the algorithm hinges on analysing the variance that can develop between the sets held at to processors p_v and p_w at each step. Let V_1 be the multiset received at processor p_v and W_1 be the multiset received at processor p_w. Similarly we use V_j to denote the multiset computed as M_j by p_v at stage j of the calculation done after all values are received. Of course, W_j is used similarly.

The result of the above Lemma 7 is we can assume for our following analysis that the number of omissions received at p_v and p_w are less than o. Without loss of generality, we will assume that $V_1(\perp) \leq W_1(\perp)$.

Consider an execution, from the perspective of processors p_v and p_w. Nodes in the network can behave correctly towards both of them or omissively or byzantinely. Byzantine faulty processors may also behave omissively. For analysis we will count the number of each types of behaviour that occurs during this execution. Thus we denote by b_0 the number of processors from which p_v and p_w receive different incorrect values (of course, the algorithm can't determine this internally; it is an artifact in our analysis). For brevity we represent an incorrect value as x, so we write $(x1, x2) : b_0$ to indicate that there are b_0 such cases. Similarly, we use g for a correct value and $\perp$ for a missing value, giving a complete catalogue of cases which can be written as follows:

$(x1, x2)$	b_0
$(x1, \perp)$	b_1
$(\perp, x2)$	b_2
$(\perp, \perp)$	o_0
$(g, \perp)$	o_1
$(\perp, g)$	o_2
$(x1, x1)$	s_0
(g, g)	g_0

Notice that a processor need not suffer a symmetric fault, as defined in the fault model, in order to be counted under s_0, all that is needed is that it behave symmetrically to the two processors of interest. However, every symmetric fault will be counted under s_0 or o_0 or g_0.

For convenience we collect in Table 1 the exact sizes and bounds of the sets involved. In the following, we give names to important combinations of the counts: let $bo_1 = b_1 + o_1$, $bo_2 = b_2 + o_2$ and $bo_3 = bo_1 - bo_2$. Here are brief explanations for the values stated.

In Step 1 we gather V_1 and W_1. At this point V_1 can have $b_0 + max((b_1 + o_1), (b_2 + o_2))$ additional elements compared with $V_1 \cap W_1$, and since we assumed that V_1 had more $\perp$ entries than W_1, we see that $max((b_1 + o_1), (b_2 + o_2))$ is $(b_1 + o_1)$. W_1 will contain the same number of elements not in the intersection, however $(b_1 + o_1) - (b_2 + o_2)$ of these will be the value $\perp$.

In step 2, excess $\perp$ entries are replaced by defaults. Because we have assumed no processor has excess $\perp$ entries, we have $V_2 = V_1$ and $W_2 = W_1$.

In step 3, we remove extremal (potentially illegal) values from V and W leaving them with the same or fewer different values.

Step 4 calculates the number of $\perp$ entries in each case. Thus $j_v = bo_2 + o_0$, and $j_w = bo_1 + o_0$.

In step 5 the remaining $\perp$ entries are removed so V_5 contains $2bo_3$ elements more than W_5. Break V_5 into 2 new sets for analysis, X and V_c by placing $|W_5|$ elements of V_5 into X giving preference to the elements of V_5 present in $W_5 \cap V_5$; then we place the remaining $2bo_3$ elements into V_c. Now $|V_x| = |W_5|$ and $|W_5 - X| \leq 2b_0 + 2bo_2$, and $|X - W_5| \leq 2b_0 + 2bo_2$. By lemma 1 $variance(X, W_5) \leq 2b_0 + 2bo_2$.

Table 1. Key values at different steps in the Algorithm

Step	$\|V_i\|$	$\|W_i\|$	$\|V_i^*\|$
1,2	N	N	$N - bo_2 - o_0$
3	$N - 2(b + s)$	$N - 2(b + s)$	$N - 2(b + s) - bo_2 - o_0$
4	$2N - 4(b + s)$	$2N - 4(b + s)$	$2N - 4(b + s) - 2bo_2 - 2o_0$
5	$2N - 4(b + s) - 2bo_2 - 2o_0$	$2N - 4(b + s) - 2bo_1 - 2o_0$	$2N - 4(b + s) - 2bo_2 - 2o_0$

Step	$\|W_i^*\|$	$\|V_i^*\| - \|W_i^*\|$	bound on $\|V_i - W_i\|$	bound on $\|W_i^* - V_i^*\|$
1,2	$N - bo_1 - o_0$	bo_3	$b_0 + bo_1$	$b_0 + bo_2$
3	$N - 2(b + s) - bo_1 - o_0$	bo_3	$b_0 + bo_1$	$b_0 + bo_2$
4	$2N - 4(b + s) - 2bo_1 - 2o_0$	$2bo_3$	$2b_0 + 2bo_1$	$2b_0 + 2bo_2$
5	$2N - 4(b + s) - 2bo_1 - 2o_0$	$2bo_3$	$2b_0 + 2bo_1$	$2b_0 + 2bo_2$

Now we can complete the proof of our theorem. Note $|X| = |W_5|$, and thus $j_x = j_w$. We see that $variance(X, W_5) \leq 2b_0 + 2bo_2$. If we form X' from X just as M_6 is formed from M_5, we have $X' = reduce^{o - j_w}(X)$. Since $W_6 = reduce^{o - j_w}(W_5)$, by lemma 2, we see $variance(X', W_6) \leq variance(X, W_5)$. Thus $variance(X', W_6) \leq 2b_0 + 2bo_2$.

Now consider the relative behaviour of V and X. Step 6 does $V_6 = reduce^{o-j_v}(V_5)$, where $jv = bo_2 + o_0$, so that $o - jv = o - o_0 - bo_2 = o - o_0 - bo_1 + bo_3$. Also $V_5 = X + V_c$. So, $V_6 = reduce^{o-j_v}(V_5) = reduce^{o-o_0-bo_1+bo_3}(X + V_c) = reduce^{o-o_0-bo_1}(reduce^{bo_3}(X + V_c))$. Let $V_5' = reduce^{bo_3}(X + V_c)$. By Lemma 4, noting $|V_5| = 2bo_3$, we have $variance(V_5', X) \leq bo_3$. Also $X' = reduce^{o-o_0-bo_1}(X)$, $V_6 = reduce^{o-o_0-bo_1}(V_5')$ so $variance(X', V_6) \leq bo_3$, by Lemma 2. From above $variance(W_6, X') \leq 2b_0 + 2bo_2$, so by Lemma 3 we have

$$variance(V_6, W_6) \leq 2b_0 + 2bo_2 + bo_3 = 2b_0 + bo_1 + bo_2 = 2b_0 + b_1 + b_2 + o_1 + o_2 \leq 2b + o$$

Note $|V_6| = 2N - 4b - 4s - 2o$. Since $variance(V_6, W_6) \leq 2b + o$ by lemma 6 we see $mean(select_{2b+o}(V)) - mean(select_{2b+o}(w)) \leq \delta(V \cup W)/\lceil \frac{2N-4b-4s-2o}{2b+o} \rceil$. So the guaranteed convergence rate of the algorithm is

$$K = \left(\left\lceil \frac{N - 2b - 2s - o}{b + o/2} \right\rceil \right)^{-1}$$

4 Lower Bound on Convergence of Arbitrary Algorithm

Any algorithm can be represented as a full information protocol, in which values are exchanged and then some protocol-specific function is applied to the collected messages to determine the final value. Thus we define a view as the vector of values received at each processor after the communication step. It contains a list of values and the processor they came from. Typically only the values are used, the vector being converted immediately into a multiset. Consider an execution E in this fault model. It may be described by the starting value of each node and stating which nodes have which fault types, and how those faulty nodes behave. During E not all nodes need receive the same data vector (view) due to faulty behaviour of some nodes as specified in E. We say that two views V and W are directly compatible ($V \approx W$) if there can exist an execution E in which V can be received at one correct processor and W be received at another correct processor. In this situation we say that E produces V and W ($E \Rightarrow (V, W)$).

A valid approximation function for this model is one that takes any view/multiset and produces a new approximation guaranteed to be in the range of correct values. If there are b byzantine and s symmetric faults then the $(b + s)$ smallest and the $(b + s)$ largest values received are suspect. Thus a valid approximation function must always choose a value in the range of the $(b + s + 1)$th smallest to $(b + s + 1)$th largest elements received.

In our lower bound we will use a simple counting argument, expressed in the following result.

Lemma 9. *Let f be any valid approximation function. Suppose that $V_0, V_1, \ldots, V_c$ is a chain of views such that $V_i \approx V_{i+1}$ and where $f(V_0) = 0$ and $f(V_c) = 1$. There exists an i $(0 \leq i < c)$ such that*

$$|f(V_i) - f(V_{i+1})| \geq \frac{1}{c}$$

Proof. We argue by contradiction, and let $S = \langle s_1, s_2, \ldots, s_c \rangle$ where $s_i = f(V_i) - f(V_{i-1})$.

$$\sum_{i=1}^{c} s_i = (f(V_1) - f(V_0)) + (f(V_2) - f(V_1)) + \cdots + (f(V_c) - f(V_{c-1})) = f(V_c) - f(V_0) = 1$$

However, if $s_i < \frac{1}{c}$ for all i then $\sum_{i=1}^{c} s_i < c * \frac{1}{c} < 1$, which, since $\sum_{i=1}^{c} s_i = 1$, isn't true. So for some $i : s_i \geq \frac{1}{c}$

We can now use the lemma to establish our lower bound

Theorem 10. *If A is any algorithm that solves Approximate Agreement in all executions of a distributed systems with at most b byzantine faults, s symmetric ones and o omission failures, then the convergence of A is bounded below by*

$$K \geq \left(2 \left\lceil \frac{N - 2b - 2s - o}{2b + o} \right\rceil \right)^{-1}$$

Proof. We will produce a chain of views of length c, such that every consecutive pair are directly compatible, the final value produced from the first must be 0, the final value produced from the last must be 1, and every one contains only entries of 0, 1 or $\perp$. The lemma above will then show that one execution exists in which two neighbouring views are produced, where the final values differ by at least $\frac{1}{c}$. In this execution, all initial values are 0 or 1, showing that the interval of values is reduced to no less than $1/c$; that is, the convergence is no less than $1/c$.

We will now complete the lower bound by providing a specification of a chain of executions $E_0, \ldots, E_{c-1}$ which produces a chain of views $V_0, \ldots, V_c$ ($E_i \Rightarrow (V_i, V_{i+1})$) which satisfies the above conditions. The precise characterisation is given below, but we offer a diagram that may make things more clear. The chain starts with all entries being 0 except for some nulls and ones, whose total number is less than the maximum faults; thus the decision value must be 0, since possibly all non-zero values came from faulty processors and so were not initial values of correct ones. Similarly the decision value in the final view must be 1, with all entries (except a few that might be from faulty processors) being 1. In between views in the chain are made by gradually replacing zero-value entries with nulls and 1-value entries, as shown in Figure 1.

Now to make this precise, we indicate exactly what happens in each execution, giving different constrictions for alternating pairs in the chain. In the execution E_i containing nodes $\langle p_0, ..., p_{N-1} \rangle$ where $i \bmod 2 = 0$ the nodes behave as follows. (All ranges are inclusive.)

– Nodes p_0 to $p_{(\frac{i}{2}+1)*o-1}$ start with value 1.
– Nodes $p_{N-b*(i+1)-s}$ to p_{N-1} start with value 1.
– The remaining nodes all start with value 0.

In the message exchange, the following is observed

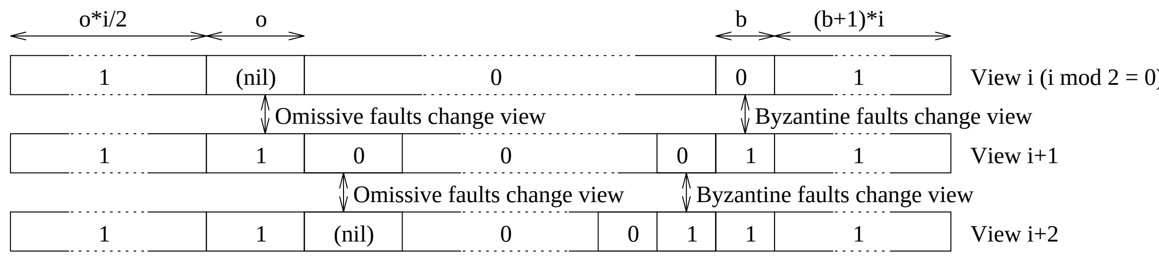

Fig. 1. Construction of the chain of views

- Nodes $p_{\frac{i}{2}*o}$ to $p_{(\frac{i}{2}+1)*o-1}$ are omissively faulty, they contribute $\perp$ to view V_i and 1 to the view V_{i+1}.
- Nodes $p_{N-b*(i+2)-s}$ to $p_{N-b*(i+1)-s-1}$ are byzantine faulty, contributing 0 to view V_i and 1 to the view V_{i+1}.
- The remaining nodes are non-faulty

In execution E_{i+1} where $i \bmod 2 = 0$ the nodes behave as follows.

- Nodes p_0 to $p_{(\frac{i}{2}+1)*o-1}$ start with value 1.
- Nodes $p_{N-b*(i+1)-s}$ to p_{N-1} start with value 1.
- The remaining nodes all start with value 0.

The behaviour in these executions is as follows

- Nodes $p_{(\frac{i}{2}+1)*o}$ to $p_{(\frac{i}{2}+2)*o-1}$ are omissively faulty, they contribute 0 to view V_{i+1} and $\perp$ to the view V_{i+2}.
- Nodes $p_{N-b*(i+3)-s}$ to $p_{N-b*(i+2)-s-1}$ are byzantine faulty, contributing 0 to view V_{i+1} and 1 to the view V_{i+2}.
- The remaining nodes are non-faulty

As can be seen $V_0(1) = b + s$ with the only other values being 0 and $\perp$, so that and valid f will have $f(V_0) = 0$. E_i and E_{i+1} both produce the same thing for V_{i+1}, so for any V_i and V_{i+1} we have $V_i \approx V_{i+1}$ because $(E_i \Rightarrow (V_i, V_{i+1}))$. We now need only show $f(V_c) = 1$ for any f.

Notice that the number of 0's in each view is reducing by $2b + o$ every two steps.$(V_{i+2}(0) = V_i(0) - (2b+0))$. We set $c = \left\lceil \frac{N-2b-2s-o}{2b+o} \right\rceil \times 2$. This is sufficient to ensure $V_c(0) \leq b + s$. Since the only other values in V_c are 1 and $\perp$ it follows that $f(V_c) = 1$ for any valid f.

This completes the construction of the chain, so by Lemma 9 we have for any approximation function f the convergence rate is bound by

$$K \geq \frac{1}{\left\lceil \frac{N-2b-2s-o}{2b+o} \right\rceil \times 2}$$

Since $\left\lceil \frac{N-2b-2s-o}{2b+o} \right\rceil \times 2$ is equal to or one greater than $\left\lceil \frac{N-2b-2s-o}{b+\frac{o}{2}} \right\rceil$. This lower bound demonstrates that Algorithm BSO is optimal or near optimal depending on the choice of b, s, o. Importantly Algorithm BSO is optimal in the maximum robustness cases, that is when $N = 3b + 2d + \frac{3}{2}o + 1$.

Also notice that if we set $c = \left\lceil \frac{N-2b-2s-\frac{o}{2}}{b+\frac{o}{2}} \right\rceil$ in the above chain of executions/views then $V_c(0)$ is still less than $b + s$. Setting $o = 0$ as in fault-model studied in [8] and we get

$$K \geq \left(\left\lceil \frac{N-2b-2s}{b} \right\rceil \right)^{-1}$$

This is guarantees exact optimality of our algorithm for that fault model.

5 Conclusion

We have analysed the Approximate Agreement problem for distributed systems subject to at most b byzantine faults, s symmetric ones and o omission failures. We have presented an algorithm, called BSO, and shown that it has convergence given by

$$K \leq \left(\left\lceil \frac{N-2b-2s-o}{b+o/2} \right\rceil \right)^{-1}$$

We have proved a lower bound of

$$K \geq \left(2 \left\lceil \frac{N-2b-2s-o}{2b+o} \right\rceil \right)^{-1}$$

which is asymptotically matching our algorithm.

In future work we plan to extend our analysis to asynchronous systems, and to the closely related clock synchronisation problem.

References

1. M. Azadmanesh, R. Kieckhafer, "New Hybrid Fault Models for Asynchronous Approximate Agreement" *IEEE Trans on Computers*, **45**(4):439–449, (1996).
2. D. Dolev, N. Lynch, S. Pinter, E. Stark, W. Weihl, "Reaching Approximate Agreement in the Presence of Faults", *Journal of the ACM*, **33**(3):499–516, (1986).
3. C. Dwork, Y. Moses, "Knowledge and Common Knowledge in a Byzantine Environment: Crash Failures", *Information and Computation*, **88**(2):156-186, (1990).
4. A. Fekete, "Asymptotically Optimal Algorithms for Approximate Agreement", *Distributed Computing*, **4**:9–29, (1990).
5. A. Fekete, "Asynchronous Approximate Agreement" *Information and Computation* **115**(1):95–124, (1994).
6. M. Fischer, N. Lynch, "A Lower Bound for the Time to Assure Interactive Consistency", *Information Processing Letters* **14**(4):183–186 (1982).

7. M. Fischer, N. Lynch, M. Patterson, "Impossibility of Distributed Consensus with One Faulty Process", *Journal of the ACM*, **32**(2):374–382 (1985).
8. R. Kieckhafer, M. Azadmanesh, "Reaching Approximate Agreement with Mixed-Mode Faults", *IEEE Trans on Parallel and Distributed Systems*, **5**(1):53–63, (1994).
9. M. Pease, R. Shostak, L. Lamport, "Reaching Agreement in the Presence of Faults", *Journal of the ACM* **27**(2):228–234 (1980).
10. P. Thambidurai, Y. Park, "Interactive Consistency with Multiple Faiilure Modes", in *Proc 7th IEEE Symposium on Reliable Distributed Systems*, (1988).