Optimal Resilience Asynchronous Approximate Agreement

Ittai Abraham, Yonatan Amit, and Danny Dolev

School of Computer Science and Engineering,
The Hebrew University of Jerusalem, Israel
{ittaia, mitmit, dolev}@cs.huji.ac.il

Abstract. Consider an asynchronous system where each process begins with an arbitrary real value. Given some fixed $\epsilon > 0$, an approximate agreement algorithm must have all non-faulty processes decide on values that are at most ϵ from each other and are in the range of the initial values of the non-faulty processes.

Previous constructions solved asynchronous approximate agreement only when there were at least $5t+1$ processes, t of which may be Byzantine. In this paper we close an open problem raised by Dolev et al. in 1983. We present a deterministic optimal resilience approximate agreement algorithm that can tolerate any t Byzantine faults while requiring only $3t+1$ processes.

The algorithm's rate of convergence and total message complexity are efficiently bounded as a function of the range of the initial values of the non-faulty processes. All previous asynchronous algorithms that are resilient to Byzantine failures may require arbitrarily many messages to be sent.

Keywords: approximate agreement, Byzantine agreement, asynchronous systems.

1 Introduction

In the classical Byzantine Generals problem a set of processes begin with some initial value and must reach agreement on one of the initial values is spite of having some faulty processes. In the approximate version it is required that the values of all non-faulty processes eventually converge to a range that is bounded by some predefined $\epsilon > 0$.

It is well know that in asynchronous communication models reaching agreement is impossible under the possibility of having even one faulty process [8]. In sharp contrast, Dolev et al. [3, 4], show that approximate agreement is possible in asynchronous systems that have $5t+1$ processes, t of which may be Byzantine.

In this paper we solve the open question raised by [3, 4]. We show that Approximate Agreement can be reached with $3t+1$ processes, t of which may be Byzantine. Fischer et al. [7] show that there is no approximate agreement protocol with $3t$ or less processes that can tolerate t Byzantine failure. Hence our algorithm has optimal resilience.

T. Higashino (Ed.): OPODIS 2004, LNCS 3544, pp. 229–239, 2005.

The results are further strengthened by bounding the total number of rounds and the total number of messages sent. We bound the number of rounds until termination as a function of the range of initial values of the non-faulty processes. Our round and message efficiency is in contrast to all previous asynchronous solutions [4] in which the faulty processes can cause the protocol to run for an arbitrarily high (yet final) number of rounds.

The results presented in this paper are obtained using two building blocks. One is an asynchronous version of the Reliable-Broadcast protocol of Srikanth and Toueg [11]. The other building block is a novel *witness* technique. The witness technique limits the ability of faulty processes to lie about the range of values they were able to collect. This building block seems to be very powerful and it enables the non-faulty processes to rapidly converge their values.

There is a large body of work on stronger assumptions or weaker properties of the Approximate Agreement problem. In the synchronous version of approximate agreement, Mahaney and Schneider [10] improve the time complexity by using the Crusader Agreement protocol of Dolev [2] as a building block. Fekete [5] gives algorithms with asymptotically optimal convergence rates for the synchronous version of the problem. Fekete [6] also gives efficient algorithms for the asynchronous approximate agrement problem that is resilient against weaker adversaries of failure by omission and crash-failures. Kieckhafer and Azadmanesh [9] give a hybrid synchronous algorithm that can withstand both Byzantine and benign failures.

Another alternative for weakening the properties of Agreement is to require a probabilistic termination property that only guarantees a finite expectancy of termination (but there may exist infinite executions). Bracha [1] presents a randomized $3t + 1$ resilient Byzantine Agreement protocol with probabilistic termination. In contrast, our protocol is deterministic and always guarantees termination.

1.1 Model and Problem Definition

Consider a set of n processes. Processes communicate by a fully connected asynchronous network with reliable FIFO channels. Messages sent will eventually arrive after a finite unbounded amount of time. The channels between any two processes maintain FIFO property, if p sends to q message m and later sends message m' then q will first receive m and only later receive m'.

We assume that t of the processes may be Byzantine. All other processes follow the algorithm and are denoted as non-faulty.

Assume each non-faulty process begins with an arbitrary real input value and fix some (arbitrarily small) pre-agreed $\epsilon > 0$. An *Approximate Agreement Algorithm* must satisfy the following two conditions:

Agreement. All non-faulty processes eventually halt with output values that are within ϵ of each other;

Validity. The value output by each non-faulty process must be within the range of the initial values of non-faulty processes.

1.2 Notations

Let V denote the set of processes, and G the set of non-faulty processes. Hence $n = |V|$ and $|G| \geq n - t$.

Let S denote a finite multiset of reals. Intuitively S can be thought of as a set of real numbers in which repetitions are considered. For example $\{1, 1, 3\}$ equals $\{1, 3, 1\}$ but differs from $\{1, 3\}$. Formally let $\mathbb{R}$ denote the set of reals and $\mathbb{N}$ the set of natural numbers then S is a function $S : \mathbb{R} \mapsto \mathbb{N}$ such that $\{r \in \mathbb{R} \mid S(r) \neq 0\}$ is finite. Define $|S| = \sum_{r \in \mathbb{R}} S(r)$, $\min S = \min_{S(r) \neq 0}\{r \in \mathbb{R}\}$, $\max S = \max_{S(r) \neq 0}\{r \in \mathbb{R}\}$, and the *range* of S as $\delta(S) = \max S - \min S$.

Given a multiset S denote $s_1, s_2, \ldots, s_{|S|}$ the values of S ordered in a non decreasing order. For any $t < |S|/2$, define $trim(S, t)$ as the multiset containing the values $s_{t+1}, s_{t+2}, \ldots, s_{|S|-t-1}$ (removing the t largest and t smallest values from S). Define

$$reduce(S, t) = \frac{\max(trim(S, t)) + \min(trim(S, t))}{2} .$$

Given a set of processes V, let P be a set of (process,value) pairs. Formally, $P \subset (V \times \mathbb{R})$. Define $P_{|2}$ as the multiset of values of the second coordinate in P. To shorten notations, we extend the multiset operators to P, for example $\max P = \max P_{|2}$, $reduce(P, t) = reduce(P_{|2}, t)$.

We use the following conventions for defining the value of a variable during the execution of a protocol. All our protocols have explicit round numbers starting with 1 and incrementing by one each iteration. Given a variable x we denote x_p^h as the value of the variable x on process p when p completes its h-th round.

2 Reliable Broadcast and a $4t + 1$ Resiliency

The basic idea of [4] is to gather at least $n - t$ values, trim the t largest and t smallest of the gathered values, and then compute some averaging function of the remaining values as the next approximation. We begin by noting why the algorithm of Dolev et al. [4] fails for $4t + 1$ processes of which at most t may be Byzantine. Suppose $t + 1$ non faulty processes begin with 0 and another $2t$ non faulty processes begin with 1. The problem is that the remaining t Byzantine processes may send conflicting values to different processes. Specifically, all processes that begin with value $i \in \{0, 1\}$ may gather at least $2t + 1$ values that equal i so the trimming will cause them to see the same value i and never to converge. We overcome this difficulty by using Reliable-Broadcast.

Instead of gathering directly $n - t$ values, the simple $4t + 1$ algorithm gathers $n - t$ values that have been sent by Reliable-Broadcast.

The properties of the Reliable-Broadcast above are a variation of the asynchronous Reliable-Broadcast of [11]:

Correctness. If a non-faulty process p with a message m on round h performs Reliable-Broadcast(m, h) then all non-faulty processes will eventually Reliable-Accept(p, m, h).

Reliable-Broadcast code for process p with message m on round h:
send (p, m, h) to all processes;

Echo() method for process q:
upon receiving (p, m, h) from p
if q never sent a message of the form $(p, \cdot, h)$ then
send (p, m, h) to all processes;
upon receiving (p, m, h) from at least $t + 1$ unique processes;
if q never sent a message of the form $(p, \cdot, h)$ then
send (p, m, h) to all processes;

Condition for **Reliable-Accept**(p, m, h) at process q:
Received (p, m, h) from at least $n - t$ unique processes;

Fig. 1. Code for Reliable-Broadcast(m) and Reliable-Accept(p, m)

Non-forgeability. If a non-faulty process p does not perform at round h the task Reliable-Broadcast(m, h) then no non-faulty process will ever perform Reliable-Accept(p, m, h).

Uniqueness. If a non-faulty process performs Reliable-Accept(p, m, h) and another non-faulty process performs Reliable-Accept(p, m', h) then $m = m'$;

Lemma 1. *The algorithm in Figure 1 realizes Correctness, Non-forgeability, and Uniqueness.*

Proof. Non-forgeability holds since non-faulty processes will never receive the non existent message directly and hence may receive at most t indirect messages. Therefore a non-faulty process will never echo a nonexisting messages and clearly will never accept such a message.

Correctness holds since eventually every non-faulty process will receive either a direct message from p or $t + 1$ indirect messages and due to non-forgeability these are the only two options.

For uniqueness, suppose that the condition Reliable-Accept(p, m, h) holds for a non-faulty process q then at least $t+1$ non-faulty processes have sent (p, m, h) hence any other non-faulty process can gather at most $n - (t + 1)$ messages of the form (p, m', h) with $m' \neq m$, hence such m' will never be accepted. □

Given the Reliable-Broadcast primitive we present a simple $4t + 1$ resilient Approximate Agreement protocol. In each round, each process waits until it performs Reliable-Accept on $n - t$ different values.

Theorem 1. *Let U denote the multiset of initial values of non-faulty processes. If all non-faulty processes run the algorithm in Figure 2 for at least $\log_2(\delta(U)/\epsilon)$ rounds then their values are at most ϵ for each other and in the range of the initial values.*

```
Code for process p:

Local variables:
    values ⊂ (V × ℝ) initially values = ⊥;
    init ∈ ℝ; // the initial value;
    val ∈ ℝ initially val = init;
    round ∈ ℕ initially round = 1;

repeat:
    Reliable-Broadcast('value', p, val, round);
    values := ⊥;
    repeat
        upon Reliable-Accept('value', q, u, h) and h = round // the first time
            values := values ∪ (q, u);
    until |values| ≥ n − t;
    val := reduce(values, t);
    round := round + 1;
```

Fig. 2. The simple $4t + 1$ algorithm

The proof of this theorem can be derived as a simple exercise from the lemmata given for the $3t + 1$ algorithm and the fact that due to the Reliable-Broadcast mechanism, every two non-faulty processes accept at least $n-2t \geq 2t+1$ common values in each round.

3 The $3t + 1$ Algorithm

We note that for $3t+1$ resilience, simply using Reliable-Broadcast and trimming is not enough. In the worst case, two processes may accept a multiset of values that intersect only at one value, and after trimming the resulting multisets will not intersect. For example, suppose $n = 4, t = 1$ and let the values be $0, 0, 1, 1$; the faulty process can arrange that all processes with value $i \in \{0, 1\}$ will receive 3 values and after trimming the median will equal i and no progress will be made.

Hence, for the $3t + 1$ resilient algorithm we use an additional mechanism of gathering witnesses. A witness for process p is a process whose first $n-t$ accepted values were also accepted by p. Process p waits to gather $n - t$ witnesses. Since each process gathers $n - t$ witnesses, every two processes have at least $t + 1$ common witnesses, and thus at least one non-faulty witness. Having a common non-faulty witness implies that every pair of non-faulty processes have at least $n - t$ commonly accepted values.

Each message is associated with a specific round. Given a message with a higher round number than the current round, the receiving process saves it and will treat it as a new message when the process will reach the relevant round.

We also need a mechanism that allows processes to know when to decide on their value and halt. Let U denote the multiset of initial values of non-faulty

processes, ideally we aim to bound the number of rounds (and hence the number of messages sent) as a function of $\delta(U)$, the range of the initial values of the non-faulty processes (non-faulty range).

We note that in the asynchronous algorithm of [4] the Byzantine process can induce arbitrarily high and low values that will cause the protocol to run for an arbitrarily large (but finite) number of rounds.

In order to achieve round and message efficiency we employ a special initial round protocol that estimates the non-faulty range. The idea is to force all processes (even Byzantine ones) to Reliable-Broadcast the vector of values they gathered. This enforces a process to send values that are all inside the range of the initial values U. We show that the estimation of $\delta(U)$ by any non-faulty process is adequate to ensure that the resulting values are within ϵ of each other.

Different processes may have different estimations on the number of rounds required. Hence, care should be taken so that processes do not halt too early and cause others never to terminate. Specifically, a process waits until it Reliable-Accepts at least $t + 1$ 'halt' messages and it reaches a round larger

```
Local variables:
    values ⊂ (V × ℝ) initially values = ⊥;
    init ∈ ℝ; // the initial value;
    val ∈ ℝ initially val = init;
    (∀x ∈ V) : report[x], proof[x] ⊂ (V × ℝ) initially proof[x] := ⊥;
    witnesses, proven ⊂ V;
    round, enough ∈ ℕ initially round = 1;
    L ⊂ ℕ initially L = ⊥;

Code for process p:
init();
repeat
    Reliable-Broadcast('value', p, val, round);
    values := ⊥;
    (∀x ∈ V) : report[x] := ⊥;
    repeat
        // delay high round messages, discard low round messages
        upon Reliable-Accept('value', q, u, h) and h = round
            FIFO-Broadcast('report', q, u, h) to all;
            values := values ∪ (q, u);
        upon FIFO-Accept('report', q, u, h) from process r and h = round
            report[r] := report[r] ∪ (q, u);
        witnesses := {x ∈ V | report[x] ⊆ values and |report[x]| ≥ n − t} ;
        check/decide();
    until |witnesses| ≥ n − t;
    val := reduce(values, t);
    round := round + 1;
```

Fig. 3. The $3t + 1$ algorithm

Code for **init()**
    Reliable-Broadcast('init', p, val);
    repeat
        upon Reliable-Accept('init', q, u) (The first value from q)
                then $values := values \cup (q, u)$;
    until $|values| \geq n - t$;
    Reliable-Broadcast('proof', $p, values$);
    repeat
        upon Reliable-Accept('init', q, u) (The first value from q)
                then $values := values \cup (q, u)$;
        upon Reliable-Accept('proof', $q, vals$) (The first proof from q)
                then $proof[q] := vals$;
        $proven := \{v \in V \mid proof[v] \neq \perp$ and $proof[v] \subseteq values\}$;
    until $|proven| \geq n - t$;
    $values := \{(q, reduce(proof[q], t)) \mid q \in proven\}$;
    $val := reduce(values, t)$;
    $enough := \lceil \log_2(\delta(values)/\epsilon) \rceil + 1$;

Code for **check/decide()**
    if $(round = enough)$ then Reliable-Broadcast('halt', $p, round$) to all;
    upon Reliable-Accept('halt', q, h) (the first halt from q) then $L := L \cup \{h\}$;
    if $|L| \geq t + 1$ and $round > \min(trim(L, t))$ then **decide** val and **halt**;

Fig. 4. The **init()** and **check/decide()** methods for process p

than the estimation of at least one non-faulty process whose 'halt' message it accepted.

The code for the $3t + 1$ algorithm appears in Figure 3 and Figure 4.

4 Analysis

4.1 Informal Properties of Witness:

In order to advance in a round a process p requires at least $n - t$ witnesses. Process x is a witness for process p if the first $n - t$ values that x claimed to accept were accepted by p.

Since 'report' messages are sent via FIFO-Broadcast then if x is a non-faulty witness for both p, q then both p and q must have accepted the *first* $n - t$ values that x has accepted.

4.2 Liveness

Lemma 2. *If no non-faulty process halts before or during round h and all of them reach round h, then all non-faulty processes eventually advance to round $h + 1$.*

Proof. Seeking a contradiction, let $S \subseteq G$ be the set of non-faulty processes that never advance to round $h + 1$.

Eventually every $p \in G$ will Reliable-Broadcast its value. Hence eventually every $p \in G$ will Reliable-Accept at least $n - t$ values. Therefore every $p \in G$ will send at least $n - t$ 'report' messages. Hence eventually all $p \in S$ will Reliable-Accept each value in these 'report' messages. Hence all $p \in G$ will eventually have at least $n - t$ witnesses, and must advance. □

Lemma 3. *All non-faulty processes eventually decide and halt.*

Proof. Seeking a contradiction, suppose some set of non-faulty processes $S \subseteq G$ never decides.

We begin by showing that at least one process must halt. Eventually by Lemma 2 the round number will be higher than the *enough* values of $t+1$ non-faulty processes and so at least $t + 1$ 'halt' messages will be sent. Recall that a process p halts when its round number is larger than $\min(trim(L_p, t))$ and $|L_p| \geq t + 1$ (see the last line of the *check/decide* method). Hence eventually some non-faulty process will halt.

Let h be the minimum round that some process $p \in G$ halts at, hence by Lemma 2 all non-faulty processes will eventually reach round h. Since 'halt' messages are sent via Reliable-Broadcast, all other non-faulty processes will eventually receive the same set of 'halt' messages (with the same round values) that caused p to halt. Hence all non-faulty processes will eventually have $\min(trim(L_p, t)) \leq h$ and so must eventually halt. □

4.3 Safety

Lemma 4 (Validity). *For all $p \in G$ and round h,*

$$\min U \leq val_p^h \leq \max U \ .$$

Proof. The proof is by induction on round numbers. Clearly the initial values are in U by definition. Assuming that all the values of the previous round (or the initial values for $h = 1$) for all $p \in G$ are in the range, then the next value val^h is a product of $reduce(values^{h-1}, t)$ for some set of values that were sent by Reliable-Broadcast (or in the **init** method, their proofs were sent via Reliable-Broadcast). Since there are at most t Byzantine processes, and *reduce* trims the t largest and t smallest accepted values, the maximal and minimal remaining values will always be inside the range of the maximal and minimal values of set of values of the non-faulty processes at the previous round. Hence the averaging in $reduce(values, t)$ will be on values that are in the range of U by the induction hypothesis. □

The witness property is stated as follows:

Lemma 5. *Every pair of non-faulty processes p, q that complete round h, maintain that*

$$\left|values_p^h \cap values_q^h\right| \geq n - t \ .$$

Proof. If non-faulty processes p, q finish round h, they have at least $t+1$ common witnesses. This follows from the fact that each has at least $n-t$ witnesses, and every $n-t$ quorum has a $t+1$ intersection with every other quorum. Hence p, q have at least one common non-faulty witness r.

By the definition of witnesses and the FIFO properties of the 'report' messages, the first $n-t$ values accepted by r will appear both in $values_p$ and in $values_q$. □

Define $U_i = \bigcup_{p \in G} val_p^i$ be the multiset containing the *val* values of all the non-faulty processes after they all completed round i. We now show an exponential decrease in the range.

Lemma 6. *The range of non-faulty processes is cut by at least a half*

$$\delta(U_i) \leq \frac{\delta(U_{i-1})}{2} \ .$$

Proof. By Lemma 5 we know that every two processes have in common at least $n-t$ accepted values. Let p, q be two arbitrary non-faulty processes, with $values_p^i, values_q^i$ as their multiset of values. Without loss of generality we assume that $val_p^i \geq val_q^i$. Denote $m = \min(U_{i-1})$ and $M = \max(U_{i-1})$. It is sufficient to prove the following:

Claim: $val_p^i - val_q^i \leq \frac{M-m}{2}$

Proof: Denote $R = values_p^i \cap values_q^i$, thus $|R| \geq n-t$ and denote $V_p = trim(values_p^i, t)$, $V_q = trim(values_q^i, t)$.

Let x be the median of R, then $x \in V_q$ because R has at least $n-t$ elements and we only trim t from each side. Hence $\max(V_q) \geq x$. In addition, $\min(V_q) \geq m$ because $trim$ removes the t smallest elements in $values_q^i$. Therefore $val_q \geq \frac{m+x}{2}$.

In a similar fashion, $x \in V_p$, which implies $\min(V_p) \leq x$ and $\max(V_p) \leq M$ and thus $val_p \leq \frac{M+x}{2}$. Combining with the above we get $val_p - val_q \leq \frac{M-m}{2}$. □

4.4 Termination Detection

We now show that the algorithm runs for sufficiently many rounds to ensure non-faulty values are at most ϵ of each other.

Lemma 7. *Let k denote the minimal round estimation $k = \min\{enough_r \mid r \in G\}$. Then if all $p \in G$ complete round k then*

$$(\forall p, q \in G)\ |val_p^k - val_q^k| \leq \epsilon \ .$$

Proof. Let p be a process such that $enough_p = k$. Examine the $n-t$ values in $values_p$ at the end of the **init** method. Consider a non-faulty process q, it also gathers $values_q$ by the end of its **init** method. Due to the fact that the values and proofs are sent via Reliable-Broadcast, process q can receive and accept at most t values that are not in $values_p$, all other values that q accepts must agree

with $values_p$. Hence if we trim the t largest and t smallest values in $values_q$ the range of the remaining values is inside the range of $values_p$. Formally, we have

$$\min values_p \leq reduce(values_q, t) \leq \max values_p \ .$$

Then by iteratively applying Lemma 6, after all non-faulty processes run for $\log_2(\delta(values_p)/\epsilon)$ rounds, their values will be close enough. □

Let U be the set of initial values of non-faulty processes and A be the set of all the initial values that are eventually accepted by Reliable-Accept by the end of the initial round (so $U \subseteq A$). Let $C = trim(A, t)$. Clearly $\delta(C) \leq \delta(U)$ because removing the t largest and t smallest elements from A results in a range that is at most the range of U.

Lemma 8. *The number of rounds that any non-faulty process completes is at most*

$$\log_2 \left(\frac{\delta(U)}{\epsilon} \right) \ .$$

Proof. For any process q, and prover $r \in proven_q$ we have

$$\min C \leq reduce(proof[r]_q, t) \leq \max C$$

Notice that r may be faulty. This is true since $proof[r]$ is a set of $n - t$ values that were sent by Reliable-Broadcast, hence $proof[r] \subseteq A$ and trimming $proof[r]$ results in a range that is smaller than $\delta(C)$. Hence $\min U \leq \min values_q$ and $\max values_q \leq \max U$, because $values_q$ is set in **init** to be the set of $reduce(proof[r], t)$, for all $r \in proven_q$. Therefore $enough_p \leq \log_2 \left(\frac{\delta(U)}{\epsilon} \right)$ for all $p \in G$.

Let $E = \bigcup_{p \in G} enough_p$ then all $p \in G$ halt after at most $\min(trim(E, t))$ rounds. This is true because they will eventually receive $t + 1$ 'halt' messages and decide. However after round $\min(trim(E, t))$ no process can gather $n - t$ replies and hence cannot advance further. □

Theorem 2. *All non-faulty processes terminate after at most* $\log_2(\delta(U))/\epsilon)$ *rounds, with values that are at most* ϵ *of each other, in the range of the initial values.*

Proof. All non-faulty processes halt by Lemma 3. Termination is in at most $\log_2(\delta(U)/\epsilon)$ rounds, deduced from Lemma 8. Since termination requires $t + 1$ *halt* messages, it occurs at a round that is larger than $enough_p$ for some $p \in G$, hence from Lemma 6 and Lemma 7 the decision values are ϵ from each other. Finally, Lemma 4 proves that the decision values are inside the initial values of the non-faulty processes. □

5 Conclusions

In this paper we solve the open question left from the original paper solving the Approximate Agreement problem. The protocol presented limits the ability of the faulty processes to influence the convergence of the non-faulty processes.

The novel witness technique used in the paper seems to be very powerful. We wonder how useful it is in solving other problems. For example, what impact can it have for solving clock synchronization problems?

An interesting topic is the bounds on the rate of convergence. Now that it is only depends on the range of values of the non-faulty processes, one can look for an optimal convergence rate.

References

1. G. Bracha. An asynchronous $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol. In *Proceedings of the third annual ACM symposium on Principles of distributed computing*, pages 154–162. ACM Press, 1984.
2. D. Dolev. The byzantine generals strike again. *J. Algorithms*, 3(1), 1982.
3. D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl. Reaching approximate agreement in the presence of faults. In *Proceedings of the 3rd Symposium on Reliability in Distributed Systems*, 1983.
4. D. Dolev, N. A. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl. Reaching approximate agreement in the presence of faults. *J. ACM*, 33(3):499–516, 1986.
5. A. D. Fekete. Asymptotically optimal algorithms for approximate agreement. In *Proceedings of the fifth annual ACM symposium on Principles of distributed computing*, pages 73–87. ACM Press, 1986.
6. A. D. Fekete. Asynchronous approximate agreement. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 64–76. ACM Press, 1987.
7. M. J. Fischer, N. A. Lynch, and M. Merritt. Easy impossibility proofs for distributed consensus problems. In *Proceedings of the fourth annual ACM symposium on Principles of distributed computing*, pages 59–70. ACM Press, 1985.
8. M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, 1985.
9. R. M. Kieckhafer and M. H. Azadmanesh. Reaching approximate agreement with mixed-mode faults. *IEEE Trans. Parallel Distrib. Syst.*, 5(1):53–63, 1994.
10. S. R. Mahaney and F. B. Schneider. Inexact agreement: accuracy, precision, and graceful degradation. In *Proceedings of the fourth annual ACM symposium on Principles of distributed computing*, pages 237–249. ACM Press, 1985.
11. T. K. Srikanth and S. Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2(2):80–94, 1987.