ASYNCHRONOUS CONSENSUS AND BYZANTINE PROTOCOLS IN FAULTY ENVIRONMENTS*

Gabriel Bracha
Sam Toueg

TR 83-559
June 1983

Department of Computer Science
Cornell University
Ithaca, New York 14853

*This report supercedes an earlier report TR 83-550, entitled "Resilient Consensus Protocols".

Asynchronous Consensus and Byzantine Protocols in Faulty Environments

Gabriel Bracha

Sam Toueg

Cornell University
Ithaca, New York 14853

ABSTRACT

A consensus protocol enables a system of n asynchronous processes, some of which are faulty, to reach agreement. There are two kinds of faulty processes: fail-stop processes can only die, malicious processes can also send false messages. We investigate consensus protocols that terminate within finite time with probability 1 under certain assumptions on the behavior of the system. With fail-stop processes, we show that $\lceil (n+1)/2 \rceil$ correct processes are necessary and sufficient to reach agreement. In the malicious case, we show that $\lceil (2n+1)/3 \rceil$ correct processes are necessary and sufficient to reach agreement. This is contrasted with a recent result that there is no consensus protocol for the fail-stop case that always terminates within a bounded number of steps, even if only one process can fail. We also investigate the possibility of reliable broadcast (Byzantine Agreement) in an asynchronous system. We define the notion of Asynchronous Byzantine Agreement, and show that $\lceil (2n+1)/3 \rceil$ correct processes are necessary and sufficient to reach Asynchronous Byzantine Agreement.

1. Introduction

In this paper we consider protocols for reaching agreement in an unreliable asynchronous distributed system. Numerous variations of this question appeared in the literature. These differ in the assumed message system properties, the kind of failures accorded to the processes, and the notion of what constitutes a solution.

In our model all the processes are fully interconnected. The message system is reliable, though completely asynchronous, so messages can be delayed arbitrarily long. We consider two types of faulty processes. Fail-stop processes [Schl81] may simply "die", i.e., stop participating in the protocol. However, there is no way to detect the death of such process, and distinguish between a dead process and a merely slow one. Malicious processes [Dole81, Dole82, Lamp82], beside failing to send the required messages, may also send false and contradictory messages, even according to some malevolent plan.

Each process starts with some initial value. At the conclusion of the protocol all the correct processes must agree on the same value. However, we are ruling out the trivial case that the agreed value is fixed regardless of the processes' initial input.

This type of system (with fail-stop processors) was investigated in [Fisc83]. Fischer et al. showed the impossibility of a consensus protocol if only one failure may occur. However, in [Fisc83] the notion of an admissible solution is a protocol that always terminates within a finite number of steps. In this paper we are interested in a different kind of solution: we consider protocols that may never terminate, but this would occur with probability 0, and the expected termination time is finite. There are two ways to introduce probabilities on the possible executions of a protocol. In the first approach [BenO83], random steps are introduced in the protocol. The other approach, and the one we adopt in this paper, is to postulate some probabilistic behavior about the message system.

In the case of fail-stop processors, we describe a probabilistic protocol that can withstand up to $\lfloor (n-1)/2 \rfloor$ failures, where n is the number of processes. We also show that there is no consensus protocol that can overcome more than $\lfloor (n-1)/2 \rfloor$ failures. With malicious processes, we describe a protocol that can withstand up to $\lfloor (n-1)/3 \rfloor$ failures. We also prove the impossibility of such a protocol if more than $\lfloor (n-1)/3 \rfloor$ processes may fail.

We also investigate the possibility of reliable broadcast (Byzantine Agreement) in an asynchronous system. This problem was extensively investigated in synchronous systems [Dole81, Dole82, Dole83, Lamp82]. We define a weaker notion of agreement, the Asynchronous Byzantine Agreement, in which a broadcast by a malicious process may fail (i.e., not terminate), but if it succeeds then all the processes agree on the same value. We show that at least $\lceil (2n+1)/3 \rceil$ correct processes are necessary to reach Asynchronous Byzantine Agreement, and we describe a protocol that achieves this bound.

2. The fail-stop case

2.1. The model

We consider an asynchronous system of n fully interconnected processes. Processes communicate by sending *messages* via the *message system.* The message system maintains for each process a *message buffer* of messages sent to it but not yet received. It also supports the following primitives for each process q.

send(p,m):

instantaneously place the message m in process p's buffer.

receive(m):

remove some message from q's buffer and return it in m, or return the null value ϕ. This choice is made nondeterministically. (Returning ϕ, even if the buffer is not empty, is a device to model the arbitrarily long transmission delays incurred in a message system.)

Each process has an unbounded internal storage whose value constitutes its *state*. In an *atomic step* of the system, a process can try to receive a message, perform an arbitrary long local computation, and then send a finite set of messages. The computation and the messages sent are prescribed by the *protocol,* i.e., a function of the message received and the local state.

A *correct* process always follows the protocol until the protocol completion. A *fail-stop* process may *die* during the execution of the protocol, i.e., it may stop participating in the protocol. The death of a process occurs without warning messages. From our model, it is clear that such a death can not be detected by other processes. In particular, there is no way to distinguish between a dead process and a merely slow one.

Each process p has two distinguished memory locations, *input* i_p and *decision* d_p. The system starts with all the processes in some initial state, all the buffers empty, d_p undefined, and i_p having some value in $\{0,1\}$. The protocol can assign to d_p a value in $\{0,1\}$. Once d_p is assigned a value v, it can not be changed, and p is said to have *decided* v.

The *configuration* of the system is the collection of the states and the buffer contents of all the processes. Let C be a configuration and S be a subset of processes. A *sub-configuration* C_S is the restriction of C to the members of S. Henceforth, we reserve the notation F^i to denote any configuration where all the correct processes have decided i, and F_S^i to denote any sub-configuration where all the correct processes in S have decided i.

A sequence of atomic steps is called a *schedule*. If the execution of a schedule σ from a configuration C results in a configuration D we write $C \overset{\sigma}{\vdash} D$. If there is a schedule σ such that $C \overset{\sigma}{\vdash} D$ we write $C \vdash D$; if all the processes performing atomic steps in σ belong to a subset of processes S then we write $C_S \vdash D_S$, and say that D_S is *reachable from* C_S. The configuration D is said to be *reachable* if it is reachable from some initial configuration.

A *k-resilient consensus protocol* is a protocol that satisfies the following properties, provided that no more than k processes are faulty.

bivalence: if all the processes are correct, both F^0 and F^1 configurations are reachable†.

consistency: there is no reachable configuration where correct processes decide different values.

convergence: for any initial configuration,

$$\lim_{t \to \infty} \text{Probability [a correct process has not decided within } t \text{ steps]} = 0.$$

Note that, from the consistency and the convergence properties of k-resilient protocols, if $C_S \vdash F_S^i$ then $C \vdash F^i$.

† In Section 5, we discuss other interpretations of bivalence that appeared in the literature, and their implications on this paper's results.

2.2. A lower bound on the number of correct processes

The possibility of undetectable deaths during the execution of the protocol implies that, at any stage of the protocol, processes will have to act relying only on partial information about the state of the system. This is formalized by the following lemma.

Lemma 1. With a k-resilient consensus protocol, for any reachable configuration C, and for any subset S of processes that contains at least $n-k$ correct processes, either $C_S \vdash F_S^0$ or $C_S \vdash F_S^1$.

Proof: Let C be a reachable configuration, S be a subset of processes that contains at least $n-k$ correct processes, and $\overline{S}$ be the complement of S (i.e., the set of processes that are not in S). Note that $|\overline{S}| \leq k$. Assume first that all the processes in $\overline{S}$ are fail-stop. Suppose that, after reaching configuration C, all the processes in $\overline{S}$ die without sending warning messages. This results in a configuration C'. We have $C'_S = C_S$. From the consistency and the convergence properties of the k-resilient protocol, we must have $C'_S \vdash F_S^0$ or $C'_S \vdash F_S^1$. Since $C'_S = C_S$, and since the death of processes in $\overline{S}$ cannot be detected, we have $C_S \vdash F_S^0$ or $C_S \vdash F_S^1$. This must also hold even if there are correct processes in $\overline{S}$. □

Let C be a configuration, and S a subset of processes as in Lemma 1. If both $C_S \vdash F_S^0$ and $C_S \vdash F_S^1$ then C_S is ***bivalent.*** If $C_S \vdash F_S^0$ or $C_S \vdash F_S^1$, but not both, then C_S is ***univalent*** (***0-valent*** or ***1-valent,*** accordingly).

Lemma 2. [Fisc83] For $k \geq 1$, any k-resilient consensus protocol has a bivalent initial configuration.

Proof: Suppose all the processes are correct. Initial configurations differ only by the processes' input values. Two initial configurations differing by the input value of only one process are ***adjacent.*** Assume, for contradiction, there is a k-resilient protocol such that any initial configuration is either 0-valent or 1-valent. By the bivalence property of the protocol there must be one of each. Therefore, there must be two adjacent initial configurations, I^0 and I^1, that are 0- and 1-valent, respectively. These configurations differ only by the input value of some process p. Therefore $I_S^0 = I_S^1$, where S includes all the processes except p. From Lemma 1, either $I_S^0 \vdash F_S^0$ or $I_S^0 \vdash F_S^1$. If $I_S^0 \vdash F_S^0$, then $I_S^1 \vdash F_S^0$, and therefore we have $I^1 \vdash F^0$. But I^1 is 1-valent, a contradiction. A similar contradiction is obtained if we assume $I_S^0 \vdash F_S^1$. □

Theorem 1. There is no $\lceil n/2 \rceil$-resilient consensus protocol for the fail-stop case.

Proof: Assume there is such a protocol, and consider a system where all the processes are correct. Let C be any reachable configuration, S be any subset of processes of size $\lfloor n/2 \rfloor$, and $\bar{S}$ be the complement of S. We claim that C_S and $C_{\bar{S}}$ are either both 0-valent, or both 1-valent.

From Lemma 1, since the protocol is $\lceil n/2 \rceil$-resilient and $|S|$, $|\bar{S}| \geq n - \lceil n/2 \rceil$, we have $C_S \vdash F_S^i$ and $C_{\bar{S}} \vdash F_{\bar{S}}^j$, for some decision values i and j. Suppose that there exist two schedules σ_0 and σ_1 such that $C_S \overset{\sigma_0}{\vdash} F_S^0$ and $C_{\bar{S}} \overset{\sigma_1}{\vdash} F_{\bar{S}}^1$ (or vice-versa). Then we can apply the schedule $\sigma = \sigma_0 \cdot \sigma_1$ to configuration C, and this results in a configuration where processes in S decide 0, and processes in $\bar{S}$ decide 1 (or vice-versa). This contradicts the consistency of the protocol, and our claim is proved.

By lemma 2, there is a bivalent initial configuration I. From our claim, without loss of generality, both I_S and $I_{\bar{S}}$ are 1-valent. Let σ be a schedule such that $I \overset{\sigma}{\vdash} F^0$. We denote by I^t the configuration reached from I after the first t steps in σ. Note that $I^0 = I$ and $I^{|\sigma|} = F^0$. Clearly, both $I_S^{|\sigma|}$ and $I_{\bar{S}}^{|\sigma|}$ are 0-valent. Let t be the smallest index such that both I_S^t and $I_{\bar{S}}^t$ are 0-valent. Note that $t > 0$. From our initial claim, and the minimality of t, both I_S^{t-1} and $I_{\bar{S}}^{t-1}$ must be 1-valent.

Let p be the process that performs the atomic step s such that $I^{t-1} \overset{s}{\vdash} I^t$. Suppose p belongs to S, and therefore $I_S^{t-1} \vdash I_S^t$. Since I_S^t is 0-valent, we have $I_S^t \vdash F_S^0$. Then we must have $I_S^{t-1} \vdash F_S^0$. But I_S^{t-1} is 1-valent, and this is a contradiction. We obtain a similar contradiction if we assume that p belongs to $\bar{S}$. □

Note that the proof of the theorem holds for any type of protocol, even a probabilistic or a non-deterministic one.

2.3. A $\lfloor (n-1)/2 \rfloor$-resilient consensus protocol

In this section we describe a k-resilient consensus protocol, for $k=1, 2, \ldots, \lfloor (n-1)/2 \rfloor$. The protocol consists of phases repeatedly performed by each process until it reaches a decision. In each phase, a process p first sends a message to all the processes, and then it waits for messages from $n-k$ processes. These messages constitute its *view* of the system. Here we introduce our requirement on the behavior of the system: at any phase, we want every possible view to have some fixed probability of being the one seen. More formally, there is some ϵ such that, for any two processes p and q, and any phase t, Probability[p receives a message from q in phase t] $\geq \epsilon$.

Both the state of a process and the messages exchanged consist of a *phase number*, a binary *value*, and a *cardinality*. In each phase, a process first sends a message with its state to all the processes, then it waits for messages. When a process receives $n-k$ messages, it considers the sets of messages with value 0 and value 1, respectively. A message with value i and cardinality greater than $n/2$ will be called a *witness for* i. If a process receives a witness for i it changes its value to i (we will prove later that no process can receive a witness for both values). Otherwise, it changes its value to the value i with the largest message set. In both cases, it also changes its cardinality to the size of the message set with value i, and then it starts a new phase. A process decides i if it receives more than k witnesses for value i. This indicates that there are enough witnesses for that value in the message system to force the rest of the processes to reach the same decision.

Theorem 2. For any k, $0 \leq k \leq \lfloor (n-1)/2 \rfloor$, the protocol described in Figure 1 is a k-resilient consensus protocol for the fail-stop case.

Proof: We need the following few definitions. Each execution of the protocol outer loop is called a *phase*. A process is in phase t if at the beginning of this phase its variable *phaseno* has the value t. A message (witness for i) whose *phaseno* field is equal to t is called a t-message (t-witness for i). A process p *decides in phase* t if it sets the decision variable d_p while its *phaseno* variable is equal to t. The value of the variable *var* of process p, when p is at the beginning of phase t, is denoted by var_p^t.

We prove the theorem by showing the protocol's consistency, deadlock-freedom, convergence and bivalence, in the presence of up to k faulty processors.

consistency:

Let t be the smallest phase in which a process decides. We claim that, for any processes p and q, we cannot have both $witness\_count(0)_p^t > 0$ and $witness\_count(1)_q^t > 0$. Suppose for some i, $witness\_count(i)_p^t > 0$. Then process p, in phase $t-1$, must have received from some process r a $(t-1)$-witness for i. So r must have received, in phase $t-2$, more than $n/2$ $(t-2)$-messages with value i. Therefore, if both $witness\_count(0)_p^t > 0$ and $witness\_count(1)_q^t > 0$, since there are at most n processors, there must be a least one processor that sent $(t-2)$-messages with both values. This is impossible in the protocol described in Figure 1, and the claim is proved. From this claim and the description of the protocol, it is now easy to check that a process can never have both *witness_count*(0) and *witness_count*(1) greater than 0 in the same phase.

Let t be the smallest phase in which a process decides, say, process p decides 0 in phase t. We prove that no other process q can decide 1.

Since p decides 0 in phase t, we have $witness\_count(0)_p^t > k$. From our claim, we cannot have $witness\_count(1)_q^t > k$. Therefore, if q decides in phase t, it also decides 0.

We now show that all the t-messages sent are of the form $(t, 0, cardinality)$. Since $witness\_count(0)_p^t > k$, process p receives more than k $(t-1)$-witnesses for 0. Consider a process r that sends a t message. Process r must have received $n-k$ $(t-1)$-messages, and one of them must be a $(t-1)$-witness for 0. Then process r increments *witness_count*(0) in phase $t-1$. From our initial claim, process r sets its value to 0 in phase $t-1$, and it sends $(t, 0, cardinality)$ messages in phase t.

Consider a process q that decides in phase $t+1$. From the above remark, all the t-messages received by q have value 0, and therefore q must decide 0.

We now prove that all the $(t+1)$-messages sent are of the form $(t+1, 0, n-k)$. Consider a process r that sends $(t+1)$-messages. From the description of the protocol in Figure 1, we see that if r decides in phase t, the $(t+1)$-messages it sends are of the form $(t+1, 0, n-k)$. If r does not decide in phase t, it must have received $n-k$ t-messages in phase t. We already proved that all the t-messages have value 0. So, in phase t, process r sets its value to 0 and its cardinality to $n-k$. Therefore, it sends $(t+1, 0, n-k)$ messages in phase $t + 1$.

A process r that reaches phase $t+2$ must have received $n-k$ $(t+1)$-messages. From our remark above, all the $(t+1)$-messages are witnesses for 0, and therefore r decides 0 in phase $t+2$.

Since any process that reaches phase $t+2$ decides 0, no process can ever be in a phase higher than $t + 2$, and no process can decide 1.

deadlock freedom:

Since processes wait for each other's messages, the protocol might be exposed to deadlock. We prove that the protocol is deadlock-free.

Suppose, for contradiction, the protocol runs into a deadlock. Let D be the set of deadlocked processes. Each process q in D is deadlocked in phase t_q. Let $t_0 = \min_{q \in D} t_q$, and $p \in D$ be a process that is deadlocked in phase t_0. Let S be a set of $n-k$ correct processes. There are two possible cases.

(1) No process in S decides in a phase t, $t \leq t_0 - 2$. By the minimality of t_0, every process in S either decides in phase $t_0 - 1$ or t_0, or it reaches phase t_0 without deciding, in either case it sends t_0-messages to all the processes. Therefore, there will be at least $n-k$ t_0-messages in p's buffer, and p cannot be deadlocked in phase t_0; this is a contradiction.

(2) Some process decides in phase t, $t \leq t_0 - 2$. Let t be the smallest phase in which a process decides. In the proof of the protocol consistency, we showed that no process can ever be in a phase greater than $t + 2$. We also proved that every process that reaches phase $t + 2$ decides. Note that p is deadlocked in phase $t_0 \geq t + 2$. This is a contradiction, and the proof of deadlock-freedom is complete.

convergence:

Let S be a set of $n-k$ correct processes. Suppose no process in S decides in a phase t, $t < t_0$. We prove that there is a fixed θ such that, with probability greater than θ, all the processes in S decides in phase $t_0 + 2$.

Since there are no deadlocks, every process in S will reach phase t_0. Note that, for $t = t_0$, $t_0 + 1$, and $t_0 + 2$, from our assumption on system behavior, there is ρ such that, with probability greater than ρ, every process in S receives in phase t the set of $n-k$ t-messages sent by all the processes in S in phase t. In other words, with probability greater than $\theta = \rho^3$, for three consecutive phases all the processes in S exchange messages exclusively among themselves, oblivious to the rest of the system. It is clear from the protocol that, if this happens, then all the processes in S decide in phase $t_0 + 2$.

bivalence:

If all the processes start with the same input value, all the correct processes decide that value within two steps. □

Note that the protocol computes an "approximation" of the majority of the initial input values. If more than $(n+k)/2$ processes start with the same input value, every correct process decides that value in just three phases. If no input value appears in more than $(n+k)/2$ process, then the consensus value reached is not known a priori. However, the consensus value is still likely to be equal to the majority of the initial input values.

3. The malicious case

3.1. The model

In this section, we investigate a stronger failure behavior of the processes. A *malicious* process can send false and contradictory messages (even according to some malicious design), can fail to send messages, and can change its internal state to any other state.

However, the message system must provide a way for correct processes to verify the identity of the sender of each message. Otherwise, one malicious process can impersonate the whole system, leading the correct processes to conflicting decisions.

The rest of the model is as described in Section 2.1 with the following additional definitions. A schedule is *legal* if all its steps are according to the protocol. A configuration C is *legally reachable* if it is reachable by a legal schedule. Henceforth, we reserve the notation $\vdash$ to denote only transitions by legal schedules.

3.2. A lower bound on the number of correct processes

Lemma 3. With a k-resilient consensus protocol, for any reachable configuration C, and for any subset S of processes that contains at least $n-k$ correct processes, either $C_S \vdash F_S^0$ or $C_S \vdash F_S^1$ by some legal schedule.

Proof: The malicious processes can behave just like fail-stop processes and die. The proof follows from this observation and the proof of Lemma 1. □

Lemma 4. For $k \geq 1$, any k-resilient consensus protocol has a bivalent initial configuration I such that $I \vdash F_0$ and $I \vdash F_1$ by legal schedules.

Proof: The proof closely follows the proof of Lemma 2 (where Lemma 3 is used instead of Lemma 1). □

Theorem 3. There is no $\lceil n/3 \rceil$-resilient consensus protocol for the malicious case.

Proof: Suppose there is a $\lceil n/3 \rceil$-resilient protocol. Let S and T be subsets of processes of size $\lfloor 2n/3 \rfloor$ such that $| T \cup S | = n$. Note that $| T \cap S | \leq n/3$. Let C be a legally reachable configuration. All the malicious processes have followed the protocol so far. If they continue to follow the protocol, then there is no way in which they differ from correct processes. Therefore, by Lemma 3, $C_S \vdash F_S^i$ and $C_T \vdash F_T^j$, for some decision values i and j.

We claim that C_S and C_T are either both 0-valent, or both 1-valent. Suppose not, then, without loss of generality, there are legal schedules σ_0 and σ_1 such that $C_S \overset{\sigma_0}{\vdash} F_S^0$ and $C_T \overset{\sigma_1}{\vdash} F_T^1$. Suppose that all the processes in $T \cap S$ are malicious. The following schedule is possible. From C, by schedule σ_0, we first reach a configuration where all the correct processes in S decide 0. Then, the malicious processes in $S \cap T$ change their state and their buffer contents back to what they were in C, resulting in some configuration C'. The only difference between C'_T and C_T is that in C'_T the buffers of the processes in T may have additional messages (that were added during the execution of σ_0). Since $C_T \overset{\sigma_1}{\vdash} F_T^1$, the processes in T can now follow the legal schedule σ_1 from configuration C', until all the correct processes in T decide 1. This schedule violates the consistency of the protocol, and our claim is proven.

The rest of the proof follows closely the last part of the proof of Theorem 1. Let I be the bivalent initial configuration guaranteed by Lemma 4. From our claim, without loss of generality, both I_S and I_T are 1-valent. Let σ be a legal schedule such that $I \overset{\sigma}{\vdash} F^0$. We denote by I^t the configuration reached from I after the first t steps in σ. Clearly, both $I_S^{|\sigma|}$ and $I_T^{|\sigma|}$ are 0-valent.† Let t be the smallest index such that both I_S^t and I_T^t are 0-valent. Note that $t > 0$. From our initial claim, and the minimality of t, both I_S^{t-1} and I_T^{t-1} must be 1-valent.

---

† Note that our definition of bivalence is crucial to the proof here. Through Lemma 4, it allows us to claim that both decision values can be reached *by legal schedules* from I. Suppose F_0 can be reached from I only by an illegal schedule, and the faulty process is in S. In this case, $I_S^{|\sigma|}$ cannot be claimed to be 0-valent, since S does not satisfy the condition of Lemma 3.

Let p be the process that performs the atomic step s such that $I^{t-1} \overset{s}{\vdash} I^t$. Assume that p belongs to S. We have $I_S^t \vdash F_S^0$, and therefore $I_S^{t-1} \vdash F_S^0$. However, I_S^{t-1} is 1-valent, and this is a contradiction. We obtain a similar contradiction if we assume that p belongs to T. □

3.3. A $\lfloor (n-1)/3 \rfloor$-resilient consensus protocol

In this section, we make the same assumption on the system behavior as in Section 2.3, and we describe a k-resilient consensus protocol for the malicious case, for $k=1, 2, \ldots, \lfloor (n-1)/3 \rfloor$. The state of a process consists of a *phase number*, and a binary *value*. As in Section 2.3, the protocol consists of phases in which processes send each other their states. In order to overcome misleading messages from the malicious processes, the state information is sent in the following manner. There are two types of messages: *initial* and *echo*. A process sends to all the processes an initial message with its name and its state. Upon receiving an initial message, every process echoes it back to all the processes. Process p, at phase t, *accepts* a message with value i from process q if it receives more than $(n+k)/2$ messages of the form $(echo, q, i, t)$.

In each phase, a process first sends (through the procedure described above) its state to all the processes, and waits until it accepts messages from $n-k$ processes. Then, it changes its value to the majority of the values of the accepted messages. A process decides i if it accepts more than $(n+k)/2$ messages with value i. We will prove that, once a process decides i, thereafter all the other correct processes will have value i.

In the protocol described in Figure 2 processes do not exit the protocol after they decide. This was done for notational convenience only, and can be avoided in the following manner: When process p decides i, it sends to all the processes the message $(initial, p, i, *)$ and echoes of the form $(echo, q, i, *)$ for all q's. These last messages are special, and whenever a process receives them, it sends them back to itself. Once a correct process has decided i, all the correct processes will have value i. Therefore, this procedure will have the same effect as the actual participation of p in the protocol.

Theorem 4. For any k, $0 \le k \le \lfloor (n-1)/3 \rfloor$, the protocol described in Figure 2 is a k-resilient consensus protocol for the malicious case.

Proof: We show the protocol's deadlock-freedom, consistency, convergence and bivalence, in the presence of up to k faulty processes. We use the same notation and definitions as in the proof of Theorem 2.

deadlock-freedom:

We have to prove that it is always possible for a process to accept $n-k$ messages. Consider a correct process p in phase t, where t is the smallest phase among correct processes in the system. At least $n-k$ correct processes are in phase t or in a higher phase. Let q be such a process. Process q has already sent a $(initial,q,v,t)$ message to all the other processes. Since there are at least $n-k$ correct processes, p's buffer will receive at least $n-k$ $(echo,q,v,t)$ messages. Since $n-k > (n+k)/2$, then p, in phase t, eventually accepts this message with value v from q. Therefore, p accepts $n-k$ messages from correct processes, and p proceeds to the next phase.

consistency:

Consider any two processes p and q, in some phase t. We claim that, if p and q accept a message from some process r, then these messages must have the same value. Suppose not, then in phase t, p accepts a message with value 0 from r and q accepts a message with value 1 from r. Then more than $(n+k)/2$ processes sent $(echo,r,0,t)$ messages to p, and more than $(n+k)/2$ processes echoed $(echo,r,1,t)$ messages to q. Therefore, more than k processes have sent both $(echo,r,0,t)$ and $(echo,r,1,t)$. Since there are at most k malicious processes, then at least one correct process has sent both $(echo,r,0,t)$ and $(echo,r,1,t)$. From the description of the protocol, correct processes can not do that, and hence a contradiction.

Let t be the smallest phase in which a correct process decides. Let us say process p decides 0 in phase t. Process p must have accepted messages with value 0 from a set S of more than $(n+k)/2$ processes. By deadlock-freedom, any other correct process q will accept, in phase t, messages from $n-k$ processes. Therefore, it must accept messages from more than $(n+k)/2-k = (n-k)/2$ processes in S. By our claim, the value of the messages accepted by q from processes in S must be 0. So q accepts more than $(n-k)/2$ messages with value 0, and it changes its value to 0.

In phase $t+1$, all the correct processes will have value 0. Note that it takes at least $(n-k)/2$ messages with value 1 to change the value of a correct process to 1. Since there are only $k<n/3$ malicious processes, and $k<(n-k)/2$, this can never happen. Therefore, from phase t on, all the correct processes will have value 0 and they can not decide 1.

convergence:

Let S be a set of correct processes that have yet not decided. Suppose no process in S decides in a phase t, $t < t_0$. We prove that there is a fixed θ such that, with probability greater than θ, all the processes in S decide in phase $t_0 + 1$.

Since there are no deadlocks, every process in S reaches phase t_0. From our assumptions on the system behavior, there is θ such that in phases t_0, and $t_0 + 1$ the following happens with probability greater than θ. In phase t_0, every process in S accepts messages from the same set of $n-k$ processes. In phase $t_0 + 1$, every process in S accepts messages only from correct processes. It is clear from the protocol that all the processes in S decide in phase $t_0 + 2$.

bivalence:

If all the processes start with the same input value, within two phases all the correct processes decide that value. □

Note that if $k < n/5$, once a correct process decides, all the other process also decide within one phase. As in the fail-stop case, this protocol computes an "approximation" of the majority of the initial input values. If more than $(n+k)/2$ correct processes start with the same input value, every process decides that value in just two phases.

4. Performance analysis

In this section we bound the expected number of phases required to reach agreement. The protocols described in Fig. 1 and Fig. 2 are intended to overcome the maximal number of faulty processes. However this renders them either hard to analyze (as in Fig. 1), or very inefficient (as in Fig. 2). Therefore, we will investigate convergence times with fewer faulty processes than the maximum possible.

We assume that the faulty processes do their worst to slow convergence, i.e., they try to enable more divergent views of the system by the processes. Hence, in the fail-stop case none of them will fail, and in the malicious case they will try to balance the number of 1 and 0 messages in the system. In either case, in any phase, n messages will be sent to each process.

We further make the simplifying assumption that, in every phase, any set of $n-k$ messages has the same probability of being received.

4.1. The fail-stop case

We analyze a simple variant of the protocol in Fig. 2, that is a $\lfloor (n-1)/3 \rfloor$-resilient protocol. The outline of the protocol is as follows: In each phase processes send each other their value, and wait for $n-k$ messages. Processes change their values to the majority of the received message values, and decide a value when receiving more than $(n+k)/2$ messages with that value.

We can describe the execution of the protocol as a Markov chain P with states $0, \ldots, n$. The system is in state i if i processes have value 1. For $k=n/3$, we get the following transition probabilities.

$$P_{i,j} = \binom{n}{j} w_i^j (1-w_i)^{n-j} \tag{1}$$

$$\textit{where} \qquad w_i = \sum_{2n/3 \geq r > n/3} \frac{\binom{i}{r}\binom{n-i}{2n/3-r}}{\binom{n}{2n/3}} .$$

The absorbing states are $0, \ldots, n/3-1$ and $2n/3+1, \ldots, n$.

Let E_j denote the expected absorption time starting from state j. It is clear that $E_{n/2} \geq E_{n/2+1} \cdots \geq E_{2n/3+1} = 0$ and $E_{n/2} \geq E_{n/2-1} \cdots \geq E_{n/3-1} = 0$. This gives us a simple intuitive notion of the "distance" of a state from the absorbing states. We can use it to modify our matrix in a way which will increase the expected absorption time. The closer the state is to the center of the matrix, the further it is away from the absorbing states. Therefore, if for $i<j<n/2$ and $l<n/2$, we decrease $P_{n/2 \pm l, n/2 \pm j}$ and increase $P_{n/2 \pm l, n/2 \pm i}$, the resulting matrix will describe a Markov chain with slower convergence rate. As a particular instance of this procedure we can "identify" one state with another, and substitute state $n/2 \pm j$'s row with state $n/2 \pm i$'s row.

We partition the states into the following five groups:

$A = [0, \ldots, n/3-1]$

$B = [n/3, \ldots, n/2 - l\sqrt{n}/2 - 1]$

$C = [n/2 - l\sqrt{n}/2, \ldots, n/2 + l\sqrt{n}/2]$

$D = [n/2 + l\sqrt{n}/2 + 1 \,, \ldots, \, 2n/3]$

$E = [2n/3 + 1 \,, \ldots, \, n]$

We identify all B states with state $n/2 - l\sqrt{n}/2 - 1$, all C states with state $n/2$, and all D states with state $n/2 + l\sqrt{n}/2 + 1$. Having done that we can collapse each group into a single state, by adding all the columns in a group to a single column and eliminating the multiple rows. This leaves us with a 5 by 5 matrix M.

In order to compute M's entries we use the following approximation: Let the random variable $\mathbf{X}$ be the sum of n Bernoulli trials with success probability p, and $j \geq np$. We approximate $\mathbf{X}$'s distribution by the normal distribution, i.e.

$$\mathbf{Pr}\{\mathbf{X} \geq j\} \approx \Phi((j - np)/\sqrt{np(1-p)}) \quad where \quad \Phi(x) = \frac{1}{2\pi} \int_x^\infty e^{-t^2/2} dt \tag{2}$$

In state $n/2$ processes can decide 0 or 1 with equal probability, i.e. the expected number of the processes that decide 1 (0) in the next phase is $n/2$.

$$M_{C,C} = \sum_{j \in C} P_{n/2,j} \approx 1 - 2\Phi(l)$$

By setting $P_{C,A} = 0$ we decrease the expected absorption time, which leaves us with $M_{C,B} = \Phi(k)$.

In order to compute row B's entries, we need a bound on $w_{n/2 - k\sqrt{n}/2 - 1}$. Consider a population of n items, b of them are special, and a random sample of size r. Let the random variable $\mathbf{X}_{(n,b,r)}$ be the number of special items in the sample. $\mathbf{X}_{(n,b,r)}$ has the hyper-geometric distribution. by (1)

$$w_{n/2 + l\sqrt{n}/2 - 1} = \mathbf{Pr}[\mathbf{X}_{(n, n/2 - l\sqrt{n}/2 - 1, 2n/3)} > n/3] \tag{3}$$

the mean is

$$\mathbf{E}(\mathbf{X}_{(n,b,r)}) = rb/n \tag{4}$$

and the variance is

$$\mathbf{Var}(\mathbf{X}_{(n,b,r)}) = \frac{rb(n-b)(n-r)}{n^2(n-1)} \quad . \tag{5}$$

By Chebychev's inequality we have

$$\mathbf{Pr}\{\ |\mathbf{X}-\mathbf{E}(\mathbf{X})\,| > t\} \le \frac{\mathbf{Var}(\mathbf{X})}{t^2} \quad . \tag{6}$$

From (3), (4), (5), and (6) we get

$$w_{n/2-l\sqrt{n}/2-1} < \frac{1}{2l^2} \quad .$$

For $l^2 = 1.5$

$$w_{n/2-l\sqrt{n}/2-1} < 1/3 \quad . \tag{7}$$

In order to make a transition from B to C, more than $n/2 - l\sqrt{n}/2 - 1$ processes have to change their value to 1, therefore

$$M_{B,C} = \sum_{j \in C} P_{n/2-l\sqrt{n}/2-1,j} \quad . \tag{8}$$

Using (2) we get

$$M_{B,C} \le \Phi\left(\frac{n/2 - l\sqrt{n}/2 - 1 - n/3}{\sqrt{2n/9}}\right) \approx \Phi((\sqrt{n} + 3l)/\sqrt{8}) \tag{9}$$

and using (2) again

$$M_{B,A} = \sum_{0 \ge j > n/3} P_{n/2-l\sqrt{n}/2-1,j} > \Phi(0) = 1/2 \tag{10}$$

Because of the symmetry of (1), and of our partition we know that $M_{B,A} = M_{D,E}$ and $M_{B,C} = M_{D,C}$. We are interested in the expected absorption time to any absorbing state, and do not care to which one. Therefore we can collapse state A and E. In the resulting matrix B and D have exactly the same transition probabilities, and we can collapse them together too. Again, we can decrease probabilities of transition to AE, and increase probabilities of transition to C. This yields the following matrix R.

$$R = \begin{array}{c} C \\ BD \\ AE \end{array} \begin{pmatrix} 1-2\Phi(l) & 2\Phi(l) & 0 \\ \Phi((\sqrt{n} + 3l)/\sqrt{8}) & 1/2 - \Phi((\sqrt{n} + 3l)/\sqrt{8}) & 1/2 \\ 0 & 0 & 1 \end{pmatrix} \tag{11}$$

Let Q denote the leading 2 by 2 sub-matrix of R. Let $N = (I - Q)^{-1}$. By [Isaa76], the expected absorption time from a state in R is given by the sum of the corresponding row in N. Thus

$$I - Q = \begin{pmatrix} 2\Phi(l) & -2\Phi(l) \\ -\Phi((\sqrt{n} + 3l)/\sqrt{8}) & 1/2 + \Phi((\sqrt{n} + 3l)/\sqrt{8}) \end{pmatrix} \tag{12}$$

and

$$N = \frac{1}{\Phi(l)} \begin{pmatrix} 1/2 + \Phi((\sqrt{n} + 3l)/\sqrt{8}) & 2\Phi(l) \\ \Phi((\sqrt{n} + 3l)/\sqrt{8}) & 2\Phi(l) \end{pmatrix}$$

and the sum of N's first row is

$$\frac{2\Phi(l) + 1/2 + \Phi((\sqrt{n} + 3l)/\sqrt{8})}{\Phi(l)} \ . \tag{13}$$

After substituting the value of l we get that the expected number of phases is less than 7.

4.2. The malicious case

We analyze the protocol in Fig. 2, but restrict k, the number of faulty processes to $k \leq n/5$ and $k = l\sqrt{n}/2$, for some l. The protocol can be described as a Markov chain M, with states $0, \ldots, n-k$. The system is in state i if i correct processes have value 1. The absorbing states are $0, \ldots, (n-3k)/2-1$ and $(n+k)/2+1, \ldots, n-k$.

The worst that the malicious processes can do is to try to balance the number of 1- and 0- messages. This gives us the following transition probabilities:

$$M_{(n-k)/2 \pm i, j} = \begin{cases} P_{n/2 \pm (i-k), j} & i \geq k \\ P_{n/2, j} & i < k \end{cases} \tag{1}$$

where $P_{i,j}$ is as in section 4.1.

Starting from the balanced state $(n-k)/2$, the probability of making a transition to an absorbing state is given by:

$$\sum_{\substack{0 \leq j < (n-3k)/2 \\ (n+k)/2 < j \leq n-k}} M_{(n-k)/2, j} \approx 2\Phi(l) \tag{2}$$

Therefore the expected number of transitions to an absorbing state is bounded by $1/2\Phi(l)$. Therefore, for $k = o(\sqrt{n})$, the expected absorption time is constant.

5. The notion of bivalence

The natural requirement of *bivalence*, i.e., the requirement that a consensus protocol is capable of reaching different decision values, has been given several implicit interpretations throughout the literature. A strong interpretation of bivalence requires that for any number and any distribution of faulty processes in the system (within the bounds permitting decision), both decision values are reachable. With a weak interpretation of bivalence, it is possible that one of the two decision values can be reached only in the presence of faulty processes. Our interpretation of bivalence lies in between the above interpretations. It requires that both decision values are reachable if all the processes are correct. However, we allow a fixed decision if some of the processes are faulty. We feel that in a consensus protocol a decision value should depend on the initial input values of the processes, and not only on some aberrant behavior of the faulty processes.

These interpretations of bivalence are not equivalent, and in some contexts they give rise to different algorithms and impossibility results. One such example is the case where all the faulty processes are initially dead. A consensus protocol that overcomes up to $\lfloor (n-1)/2 \rfloor$ processes was described in [Fisc83]. This protocol satisfies the strong interpretation of bivalence, and can be easily proven to be optimal in the number of faulty processes it can withstand. However, with our interpretation of bivalence, there is a consensus protocol that can overcome any number of faulty processes. With this protocol, if all the processes are correct both decision values are attainable. But if one or more processes are faulty, the decision value is fixed†.

The consensus protocols described satisfy the strong interpretation of bivalence. Clearly, our lower bounds on the number of faulty processes also hold under this strong interpretation.

† This protocol is a modification of the protocol in [Fisc83]. The transitive closure G^+ is defined and constructed in the same way. If G^+ turns out to be strongly connected, and it contains all the processes, then all the processes will know it, and they will decide using an agreed bivalent function of all the inputs. Otherwise, they all decide 0.

6. Asynchronous Byzantine Agreement

A major problem in distributed systems is ensuring reliable broadcasts, commonly known as Byzantine Agreement [Lamp81], Unanimity [Dole81, Dole82, Dole83], or Interactive Consistency [Peas80]. All the previous treatments of the Byzantine Agreement are dealing with a synchronous system of n processes, where up to k processes can be malicious. Some specially designated process is a *transmitter* that sends a *value* to all the rest of the processes. A Byzantine Agreement is achieved if the following holds:

1. All the correct processes agree on the same value, and

2. If the transmitter is correct, all the correct processes agree on the transmitter's value.

Implicit is also the requirement that each process knows in which state the system is: "before broadcast", "executing the agreement protocol" and "after broadcast". Thus, queries about the transmitted value can be handled in a consistent manner by any correct process.

When considering asynchronous systems, we can no longer hold such a view of the system. Some correct processes can proceed with the protocol and reach agreement while others may not yet be aware that the protocol has begun. Even if a process receives a message from the transmitter or from other processes, it may be insufficient to start up the process on the protocol. Some threshold of activity is necessary to start up a process, a threshold which guarantees that all the other correct processes will also start the protocol and will agree on the same value.

The following two scenarios illustrate the necessity of such a scheme.

1. The transmitter is malicious. At time t_0 it sends 0-messages to a set S_1 of k correct processes, 1-messages to a different set S_2 of k correct processes, and no messages to the rest. All these messages are received at time t_1. After that, the transmitter stops participating in the protocol. If we regard this as a sufficient condition to start up a Byzantine Agreement protocol, then the system can proceed and agree, let us say on 1, at time t_2.

2. The transmitter is correct and sends 0-messages to all the processes. The k processes in S_1 are correct and they receive these 0-messages at time t_1, exactly as in the first scenario. Also at time t_1, k malicious processes receive the 0-messages sent by the transmitter, but they treat them as if they were 1-messages and they act exactly as the processes in S_2 in scenario 1. The other messages from the transmitter are received only at a time later than t_2. Consider the system during the interval $[t_1\ ,\ t_2]$. The view of

the system by the processes in S_1 is the same as in scenario 1, and therefore they can simulate it and agree on 1 at time t_2, thus violating requirement 2 of the Byzantine Agreement.

There are two ways to overcome this phenomenon. We can restrict the behavior of a malicious transmitter (it will be enough to force it to send $2k+1$ messages with the same value). Another way, the one we adopt, is to regard certain views of the system as insufficient to start the protocol. Processes may not start, unless presented with a view which guarantees starting up and agreement of all the correct processes.

We say that an Asynchronous Byzantine Agreement is achieved if the following holds:

1. If the transmitter is correct all the correct processes decide on its value.
2. If the transmitter is malicious then either no correct process will decide or they will all decide on the same value.

6.1. A lower bound on the number of correct processes

Theorem 5. It is impossible to achieve the Asynchronous Byzantine Agreement with $k \geq n/3$.

Proof: Suppose it is possible; since $k \geq n/3$, we can partition the processes to three disjoint set A, B and C, of size k or less. Let the transmitter be in A and consider the following scenarios:

1. The processes in A and B are correct, and the transmitter sends 0-messages. The processes in C are malicious, and they do not send any messages during the protocol. Since the transmitter is correct, and there are less than k malicious processes, the processes in A and B must agree on 0 within some time t.

2. Only the transmitter is malicious. It sends 0-messages to processes in A and B, and 1-messages to processes in C. Also, messages from C are delayed for a period longer than t before they are received. The processes in A and B have the same view of the system as in scenario 1, and therefore can agree on 0 at time t.

In a similar fashion we can construct scenario 3 with the following properties:

3. Only the transmitter is malicious. It sends 1-messages to A and C, and 0-messages to B. Messages from B are delayed for a period longer than t'. At time t' the processes in A and C agree on 1.

Now we can combine scenarios 2 and 3 to yield a contradiction.

4. The processes in A are malicious, the processes in B and C are correct. The processes in A send messages to processes in B as in scenario 2, and to processes in C as in scenario 3. All messages between processes in B and processes in C are delayed for a period longer than $max(t,t')$. In this scenario at time $max(t,t')$ the processes in B will agree on 0 and the processes in C will agree on 1, a contradiction. □

6.2. An Asynchronous Byzantine Agreement protocol

There are three types of messages in the protocol: *initial, echo* and *ready*. The protocol starts with the transmitter sending (*initial*,v) messages, where v is the transmitter value. Then processes report to each other the value they received via (*echo*,v) messages. If more than $(n + k)/2$ (*echo*,v) messages are received by a process, it sends (*ready*,v) messages to all the other processes. If a process receives $2k+1$ (*ready*,v) messages with the same value v, it decides v.

Theorem 6. The protocol in Fig. 3 achieves the Asynchronous Byzantine Agreement for $k=1, \ldots, \lfloor (n-1)/3 \rfloor$ malicious processes.

Proof: We have to show that if some correct process p decides some value, then all the correct processes also decide the same value, and that if the transmitter is correct then they all decide the transmitter's value.

First we claim that no two correct processes p and q can send (*ready*,v) messages with different values of v. Suppose this is possible, then p received more than $(n+k)/2$ (*echo*,1) messages, and q received more than $(n+k)/2$ (*echo*,0) messages. Therefore, some correct process r must have sent both (*echo*,1) and (*echo*,0) messages. But this impossible for a correct process, and our claim is proved. Since decision requires $2k+1$ (*ready*,v) messages with the same value v, it is also clear that no two correct processes can decide different values.

Suppose p decides i, then p received $2k+1$ (*ready*,i) messages. At least $k+1$ of them were sent by correct processes. Therefore every correct process will also receive at least $k + 1$ (*ready*,i) messages, and will send its (*ready*,i) message. Thus, at least $n-k$ processes will send (*ready*,i) messages. Therefore, every correct process will receive at least $2k + 1$ (*ready*,i) messages and will decide i.

It is clear that if the transmitter is correct then all the correct processes will decide on its value. □

7. Conclusion

We investigated probabilistic consensus protocols that terminate with probability 1, under certain assumptions about the behavior of the system. For a system with fail-stop processors, we showed that $\lceil (n+1)/2 \rceil$ correct processes are necessary and sufficient for achieving consensus. For a system with malicious processes, we showed that $\lceil (2n+1)/3 \rceil$ correct processes are necessary and sufficient for achieving consensus.

Probabilistic consensus protocol are also investigated in [BenO83]. The protocols are similar to those given in this paper, but randomization is incorporated in the protocol itself. They have an exponential expected termination time in the fail-stop case, and, in the malicious case, they can overcome up to $n/5$ malicious processes.

These are other examples of problems [Itai81, Lehm81] that do not have a deterministic algorithm, but have probabilistic algorithms that terminate with probability 1. From our analysis in Section 4, it seems that these probabilistic consensus protocols provide a viable solution.

We also investigated the possibility of Byzantine Agreement in an asynchronous system. We defined the notion of Asynchronous Byzantine Agreement, and showed that $\lceil (2n + 1)/3 \rceil$ correct processes are necessary and sufficient to reach Asynchronous Byzantine Agreement.

References

BenO83 M. Ben-Or Another advantage of free choice: Completely asynchronous agreement protocol, *Proc. 2nd Symposium on the Principles of Distributed Systems.*

Dole81 D. Dolev, Unanimity in unknown and unreliable environment, Proc. 22nd Annual Symposium on Foundations of Computer Science, Nashville, Tennessee, pp. 159-168, Oct. 1981.

Dole82 D. Dolev, Polynomial algorithm for multiple process agreement, *Proc. 14th Annual ACM Symposium on Theory of Computing,* San Francisco, California, pp. 404-407, May 1982.

Dole83 D. Dolev, M. Fischer, R. Fowler, N. Lynch and R. Strong, An efficient byzantine agreement without authentication, *Research report RJ3428, IBM San Jose, 1983.*

Fisc83 M. J. Fischer, N. A. Lynch, and M. S. Paterson, Impossibility of distributed consensus with one faulty process, *Proc. 2nd ACM SIGACT-SIGMOD Symposium on Principles of Database systems.*

Isaa76 D. L. Isaacson and R. W. Madsen, *Markov chains theory and practice* pp. 89-100, John Wiley & Sons, Inc, 1976.

Itai81 A. Itai and M. Rodeh, Symmetry breaking in distributive networks, *Proc. 22nd Annual Symposium on Foundation of Computer Science,* Nashville, Tennessee, pp. 150-158, Oct. 1981.

Lamp82 L. Lamport, R. Shostak, and M. Pease, The Byzantine Generals problem, *ACM Transactions on Programming Languages and Systems 4,* pp. 382-401, 1982.

Lehm81 D.J. Lehman and M.O. Rabin, On the advantages of free choice: A symmetric and fully distributed solution to the dining philosophers problem, *Proc. 8th ACM Symposium on the Principles of Programming Languages,* Jan. 1981.

Peas80 M. Pease, R. Shostak and L. Lamport, Reaching agreement in the presence of faults, *JACM vol. 27, no 2, pp. 228-234, 1980.*

Schl81 R. D. Schlichting and F.B. Schneider, Fail-stop processes: an approach to designing fault tolerant computing systems, *TR 81-479, Dept of Computer Science, Cornell University,* also to appear in TOPLAS.

```
process p: k-consensus

        value: integer init (i_p)
        cardinality: integer init (1)
        phaseno: integer init (0)
        witness_count: array[0..1] of integer init(0)
        message_count: array[0..1] of integer init(0)
        msg: record of
                phaseno: integer
                value: integer
                cardinality: integer

        while (witness_count(0) ≤ k and witness_count(1) ≤ k)
          message_count := witness_count := 0
          for all q, 1 ≤ q ≤ n, send(q,(phaseno,value,cardinality))
          while (message_count(0) + message_count(1) < n-k)
            receive(msg)
            case
              (msg.phaseno = phaseno):
                begin
                  message_count(msg.value) := message_count(msg.value) + 1
                  if msg.cardinality > n/2
                      then witness_count(msg.value) := witness_count(msg.value) + 1
                end
              (msg.phaseno > phaseno):
                send(p,msg)
            end
          if there is i such that witness_count(i) > 0
             then value := i
             else if message_count(1) > message_count(0)
                    then value := 1
                    else value := 0
          cardinality := message_count(value)
          phaseno := phaseno + 1
        end
        let i be such that witness_count(i) > k
        d_p := i
        for all q, 1 ≤ q ≤ n,
             begin
                send(q,(phaseno,value,n-k))
                send(q,(phaseno+ 1,value,n-k))
             end
```

Figure 1. A k-resilient consensus protocol for the fail-stop case.

```
process p: k-consensus

        value: integer init (i_p)
        phaseno: integer init (0)
        message_count: array[0..1] of integer init(0)
        echo_count:array[1..n : 0..1] of integer init(0)
        msg: record of
                type: (initial, echo)
                from: integer
                value: integer
                phaseno: integer

        while (true)
          message_count := 0
          echo_count := 0
          for all q, 1 ≤ q ≤ n, send(q,(initial,p,value,phaseno))
          while (message_count(0) + message_count(1) < n-k)
             receive(msg)
             if it is the first message received from the sender
                with these values of msg.type, msg.from and msg.phaseno then
               case
                (msg.type = initial):
                    for all q, 1 ≤ q ≤ n, send(q,(echo,msg.from,msg.value,msg.phasno))
                (msg.type = echo and msg.phasno = phasno):
                    begin
                       echo_count(msg.from,msg.value) := echo_count(msg.from,msg.value) + 1
                       if echo_count(msg.from,msg.value) = (n+ k)/2 + 1
                          then message_count(msg.value) := message_count(msg.value)+ 1
                    end
                (msg.type = echo and msg.phaseno > phaseno):
                    send(p,msg)
               end
          end
          if message_count(1) > message_count(0)
              then value := 1
              else value := 0
         if there is i such that message_count(i) > (n+ k)/2
              then d_p := i
          phaseno := phaseno + 1
        end
```

Figure 2. A k-resilient consensus protocol for the malicious case.

```
process p: byzantine
        msg_count: array of [types:0..1] of integer
        msg:record of type:(initial,echo,ready)
                 value:integer

        while (there is no i such that
                   msg_count(initial,i) ≥ 1 or
                   msg_count(echo,i) > (n+k)/2 or
                   msg_count(ready,i) ≥ k+1)
           receive(msg)
           if msg is the first message of that type and origin
                 then msg_count(msg.type,msg.value) = msg_count(msg.type,msg.value)+ 1
        end
        for all q, send(echo,msg.value)

        while (there is no i such that
                   msg_count(echo,i) > (n+k)/2 or
                   msg_count(ready,i) ≥ k+1)
           receive(msg)
           if msg is the first message of that type and origin
              then msg_count(msg.type,msg.value) = msg_count(msg.type,msg.value)+ 1
        end
        for all q, send(ready,msg.value)

        while (there is no i such that
                   msg_count(ready,i) ≥ 2k+1)
           receive(msg)
           if msg is the first message of that type and origin
              then msg_count(msg.type,msg.value) = msg_count(msg.type,msg.value)+ 1
        end
        decide msg.value
```

Figure 3. An Asynchronous Byzantine Agreement protocol for k malicious processes.