Bounds on Information Exchange for Byzantine Agreement

DANNY DOLEV AND RÜDIGER REISCHUK

IBM Research Laboratory, San Jose, California

Abstract. Byzantine Agreement has become increasingly important in establishing distributed properties when errors may exist in the systems. Recent polynomial algorithms for reaching Byzantine Agreement provide us with feasible solutions for obtaining coordination and synchronization in distributed systems. In this paper the amount of information exchange necessary to ensure Byzantine Agreement is studied. This is measured by the total number of messages the participating processors have to send in the worst case. In algorithms that use a signature scheme, the number of signatures appended to messages are also counted.

First it is shown that $\Omega(nt)$ is a lower bound for the number of signatures for any algorithm using authentication, where n denotes the number of processors and t the upper bound on the number of faults the algorithm is supposed to handle. For algorithms that reach Byzantine Agreement without using authentication this is even a lower bound for the total number of messages. If n is large compared to t, these bounds match the upper bounds from previously known algorithms. For the number of messages in the authenticated case we prove the lower bound $\Omega(n + t^2)$. Finally algorithms that achieve this bound are presented.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*network operating systems*; D.4.5 [**Operating Systems**]: Reliability—*verification*

General Terms: Algorithms, Theory

Additional Key Words and Phrases: Distributed systems, reliability, Byzantine Agreement

1. *Introduction*

Reaching agreement in a distributed system is essential for maintaining coordination and synchronization among the participating processors. This problem has been modeled in the following way.

The distributed system consists of n processors, of which up to t may be faulty. The processors are completely interconnected. One of these processors, called the *transmitter*, sends a private value v to the other processors. They then have to reach agreement on which value the transmitter has sent.

The type of agreement studied in this paper is called the *Byzantine Agreement* [14] and it is achieved when

(i) all correctly operating processors agree on the same value;
(ii) if the transmitter is correct, then all correctly operating processors agree on its value.

Authors' present addresses: D. Dolev, Institute of Mathematics and Computer Science, Hebrew University, Givat Ram, 91904 Jerusalem, Israel; R. Reischuk, Fakultät Mathematik, Universität Bielefeld, 4800 Bielefeld 1, West Germany.

For establishing the agreement, information has to be exchanged. In this paper we present lower bounds on the amount of information exchange to ensure that agreement is reached.

Several algorithms for obtaining Byzantine Agreement have been published [3, 4, 6, 9, 10, 12–15]. Without authentication, the best algorithm is the one presented in [10], in which the agreement is achieved within $2t + 3$ phases while exchanging $O(nt + t^3)$ messages in the worst case. The best solution using authentication is presented in [9]; it requires $t + 1$ phases and $O(nt + t^2)$ messages, where each message may contain several signatures. This algorithm may exchange $O(nt^2 + t^3)$ signatures; by a slight modification and one additional phase, this number can be reduced to $O(nt + t^3)$. Previous papers give lower bounds either on the ratio between correct and faulty processors [3, 4, 15] or on the number of phases [1, 7–9, 11].

In this paper we concentrate on algorithms using authentication and analyze how many messages have to be exchanged to reach Byzantine Agreement. We prove that in the worst case any algorithm must exchange $\Omega(n + t^2)$ messages and $\Omega(nt)$ signatures. The lower bound for the number of messages in the authenticated case differs from the known upper bound. To close this gap, we present an algorithm that, within $O(t)$ phases, sends only $O(n + t^2)$ messages. For n much larger than t there is an even simpler and better solution with $t + 3 + t/\alpha$ phases and $O(\alpha n)$ messages for $1 \leq \alpha \leq t$. The solution introduces a trade-off between the number of messages and the number of phases. This paper is a complete version of [5].

Concentrating on algorithms that use authentication does not mean sacrificing practicality, since in a real distributed system one can assume that no processor sends wrong information on purpose, and a simple error correction code instead of a signature scheme can be used to get applicable results.

The lower bound on the number of signatures implies a lower bound on the number of messages in the unauthenticated case. Therefore, for the amount of information exchange, the algorithm in [10] is the best possible to within a constant factor in the case in which n is bigger than t^2.

2. *Histories*

We first have to define a formal model for the class of algorithms we are looking at. Because we want to establish lower bounds, we use a more general notation than the one in [6].

Let PR be a set of n processors. A *phase* for PR is a directed graph with nodes corresponding to processors in PR and with labels on the edges. A label on edge (p, q) represents the information sent from processor p to processor q during the given phase. We assume that when no message is sent there is no edge. A *history* for PR is a finite sequence of n node phases. There is also a special initial phase called *phase 0*, which contains only a single inedge. This edge goes to a special processor called *transmitter* and it is labeled with exactly one value v from a set V of values. (The assumption is that the inedge at phase 0 carries the value that the transmitter is to send and that V is the set of all values it might send.)

A *subhistory* of a history H is an initial segment of H. For each subhistory H' of H and every processor p there is a unique subhistory pH' called the *individual subhistory of H' for p*. It consists of only those edges in H' with target p. We assume that, for each labeled edge, processor p knows the source of that edge. This means no processor can send a message to p claiming to be somebody else. Note that at the beginning of phase k the individual subhistory pH_{k-1}, which is derived

from the first $k-1$ phases, is all that processor p has to work with; it cannot have any other information about the states of other processors. Let the set ISH contain all possible individual subhistories and MSG be a (large enough) set of labels (messages).

An *agreement algorithm* for PR consists of a set of

$$\textit{correctness rules } R_p\text{: ISH} \times \text{PR} \to \text{MSG},$$
$$\textit{decision functions } F_p\text{: ISH} \to 2^V.$$

Given a $(k-1)$-phase individual subhistory for p and a processor q, the correctness rule R_p produces a label for the edge in phase k from p to q. The decision function F_p maps individual subhistories for p into the power set of V. If F_p is a singleton set, we say that p has *decided* on the value.

With respect to a given correctness rule, a processor p is said to be *correct at phase k of history H* if the following condition holds: Each edge from p to a processor q in phase k has a label as specified by the correctness rule for p when it is applied to the individual subhistory of H for p consisting of the previous $k-1$ phases. A processor p is *correct in history H* if it is correct at each phase of H. We call a history *t-faulty* if at most t of its processors are incorrect.

We say *Byzantine Agreement can be achieved for n processors with at most t faults* if there is an agreement algorithm for PR such that, for any t-faulty history H, the decision functions F_p obey the conditions of Byzantine Agreement:

(i) if processors p and q are correct in H, then $F_p(pH) = F_q(qH)$;
(ii) if the transmitter and processor p are correct in H, then $F_p(pH) = v$, where v is the transmitter's value.

3. *A Lower Bound on the Number of Signatures in the Authenticated Case*

We consider the worst-case behavior in which a faulty processor can invent any unauthenticated information. But we allow the processors to share a signature scheme that enables each one to sign its messages so that every receiver will recognize them as being signed by it and no one can change the contents of a message or the signature undetectably. Such a scheme is the one suggested in [2] and [16] and its use for Byzantine algorithms is described in [6], [9], [14], and [15].

We allow faulty processors to collude for cheating. Therefore every message that contains only signatures of faulty processors can be produced by them.

Since there exists an algorithm for reaching Byzantine Agreement without using signatures, the lower bound is meaningless unless we somehow count the messages that do not contain signatures. We make the technical assumption that every message in an authenticated algorithm carries at least the signature of its sender. Alternatively, the lower bound given below holds if one counts the number of signatures together with the number of messages without signatures.

THEOREM 1. *If Byzantine Agreement is achieved by an agreement algorithm that handles up to t ($t < n-1$) faults, by using authentication, then there exists a history without any faults in which the total number of signatures being sent by correct processors is at least $n(t+1)/4$.*

PROOF. Let H be the history in which all processors are correct and the transmitter's value is 0, and G the one in which all are correct and the transmitter's value is 1. Let p be any processor. By condition (ii) for Byzantine Agreement,

$F_p(pH) = 0$ and $F_p(pG) = 1$ (The decision rule for p has to yield 0 in history H and 1 in history G).

Denote by $A(p)$ the set of all processors that either receive the signature of p or p receives their signatures in at least one of the two histories. If for each processor p the set $A(p)$ contains at least $t + 1$ processors, then the theorem obviously holds. Notice that the transmitter is not necessarily in $A(p)$.

Assume to the contrary that there exists a processor p for which the cardinality of $A(p)$ is at most t. Let H' be the history in which the processors in $A(p)$ are faulty behave toward p as in H and toward all the rest of the processors as in G.

The processors in $A(p)$, as faulty processors, are able to do so, because all the messages to correct processors, other than p, do not contain p's signature and all the messages to p contain only signatures of processors in $A(p)$.

Therefore, in H' the correct processor p sees the same subhistory as in H ($pH = pH'$). Since pH' is all the information available to p when applying its decision function F_p, we can conclude that $F_p(pH') = F_p(pH) = 0$. On the other hand, each correct processor q other than p sees the subhistory it saw in G and by the same argument $F_p(qH') = F_p(qG) = 1$ must hold. Notice that there is such a correct processor q since we assume $t < n - 1$. This violates condition (i) of Byzantine Agreement. Therefore, a correctly operating Byzantine Agreement algorithm cannot allow any processor to "exchange" less than $t + 1$ signatures with other processors in H and G altogether. Notice that we have established this (worst case) lower bound not for some weird history, but for a history in which every processor behaves correctly. □

If authentication is not available, this lower bound applies directly to tne number of messages that have to be sent.

COROLLARY 1. *If Byzantine Agreement is achieved by an agreement algorithm that handles up to t ($t < n - 1$) faults, without using authentication, then there exists a nonfaulty history in which the total number of messages being sent by correct processors is at least $n(t + 1)/4$.*

PROOF. The basic assumption for algorithms that reach Byzantine Agreement without using authentication is that a processor can identify only the immediate source of every message it receives. Any processor p can claim to have received a certain message from another processor q, and a processor z, different from p and q, cannot decide whether this is true or not (except in the special case where z has already detected t faulty processors and p is not among them).

This is equivalent to the assumption that every message carries exactly one signature, the signature of the last sender of that message. Therefore, we can conclude from Theorem 1 that at least $n(t + 1)/4$ messages are necessary in any algorithm that does not use authentication. □

4. *A Lower Bound on the Number of Messages in the General Case*

Sometimes the overhead for sending a message costs more than the message itself; and, therefore, it makes sense to find algorithms that minimize the number of messages. Since a message with several different signatures appended can contain much more verifiable information than the same message without these signatures, we do not necessarily need the same number of messages as in the unauthenticated case. In this section we present a lower bound on the number of messages, independent of the size of a message or the information it carries.

As we have proved in the previous section, if we use fewer than $\Omega(nt)$ messages, some must carry several signatures. In this section we show that in certain histories at least $\Omega(n + t^2)$ messages have to be sent to ensure that agreement has been reached.

THEOREM 2. *If Byzantine Agreement is achieved by an agreement algorithm that handles up to t ($t < n - 1$) faults, then there exists a history H in which the correct processors send at least $\max\{(n-1)/2, (1 + t/2)^2\}$ messages.*

PROOF. As in the previous proof, let H (respectively, G) be the history in which all processors are correct and the transmitter sends the value 0 (respectively, 1). One of these values, let us say 0, must have the property that there exists a set Q of at least $\lceil (n-1)/2 \rceil$ processors different from the transmitter such that each one does not agree on 0 if it receives no messages at all. This implies that the correct processors in H must have sent at least $\lceil (n-1)/2 \rceil$ messages.

Now assume that the maximum is achieved by the second term. Let B be a subset of Q of size $\lfloor 1 + t/2 \rfloor$ and let A be the remaining processors. We cannot prove that every processor has to send or receive a certain number of messages increasing with t, because efficient authenticated algorithms tend not to be homogeneous. But by playing with histories we shall show that there exists a history H' in which each processor in B is faulty and can force the correct processors in A to send it at least $\lceil 1 + t/2 \rceil$ messages.

Let H' be the following history: Every processor in A is correct; the transmitter correctly sends the value 0. Each processor q in B never sends a message to other processors in B. Toward processors in A, such a processor q behaves like a correct processor with one exception—it ignores the first $\lceil t/2 \rceil$ messages received from processors in A (it ignores all of them if it gets fewer than $\lceil t/2 \rceil$). This defines a valid history with $\lfloor 1 + t/2 \rfloor$ faulty processors in which the correct processors in A have to agree on value 0, because the transmitter is correct and has sent this value.

Assume that there is a faulty processor p in B that gets at most $\lceil t/2 \rceil$ messages from processors in A. Let $A(p)$ be the set of processors of A that have sent messages to p in H. To obtain the contradiction, we change H' into history H'' in which p is correct and all the processors in $A(p)$ are incorrect. They behave like correct processors except that they do not send any messages to p. Processors in B different from p ignore any message they get from p.

By definition, the faulty processors in $B - \{p\}$ and $A(p)$ behave toward the correct processors in A in history H'' in exactly the same way as they do in H'. Since p in H' simulates the behavior of a correct processor that has not got the first $\lfloor t/2 \rfloor$ messages it was supposed to get, there is also no difference between the behavior of p toward processors in A in H' and H''. Therefore, each correct processor q other than p sees the same subhistory in both cases ($qH' = qH''$), and at the end it must agree on value 0 because $F_p(pH') = 0$.

But the correct processor p receives no messages at all in H''. Therefore, by definition of the set B, it does not agree on 0. This leads to a contradiction, which proves that in H' every processor in B must receive at least $\lceil 1 + t/2 \rceil$ messages from correct processors. □

Theorem 2 also holds in the case when faults are much more limited because the proof only uses the ability of a faulty processor to send to some and not to others. This observation is not true, in general, for Theorem 1.

5. *A Linear Algorithm for $n \geq t^3$*

None of the known authenticated algorithms uses authentication to substantially reduce the number of messages required to be sent in the case in which n is much larger than t. In this case the best-known algorithms, with and without authentication, require $\Theta(nt)$ messages [9, 10] in the worst case. However, for large n, Theorem 2 implies only a linear lower bound, and the $\Omega(nt)$ lower bound of Theorem 1 only holds for the total number of signatures that have to accompany messages. Since we can append many signatures to a message, there may exist an authenticated algorithm that, after reaching the agreement among some set of active processors, sends only a linear number of messages to inform the rest about the agreement.

In this section we present such an algorithm. The number of phases it needs does not exceed the minimal number $t + 1$ by more than a small constant factor. But many messages carry $\Omega(t)$ signatures.

We may assume that n is at least $2t + 1$; otherwise, the algorithm in [9] sends as many messages as the lower bound in Theorem 2. We consider the case $n = 2t + 1$ and describe two algorithms. The first, working in $t + 2$ phases, sends fewer messages than the previously known algorithms. The second uses more phases, but at the end every correct processor agrees on the same value and also has a one-message proof for the outside world. It possesses a string that says what the common value is, and this statement is signed by at least t other processors.

Throughout this section we assume that all processors are completely synchronized and that, as in the lower bound proofs, the values the transmitter may send are 0 or 1. If the transmitter can send more than two values, one has to modify the algorithms slightly.

Instead of using the formal model defined in Section 2 (which seems to be appropriate for lower bound proofs), we prefer a more informal description for the following algorithms.

For the first algorithm let q be the transmitter and partition the $2t$ remaining processors into two sets A and B, each of size t. Let G be the graph that is formed by the complete bipartite graph with A and B as the sets of nodes, to which we add a node q and edges from q to every other node. We call the message a processor p receives at phase k, for $k = 1, \ldots, t + 2$, *a correct 1-message* if it consists of value 1 with signatures appended to it and if the sequence of processors that signed that message, together with p, forms a simple path of length k from q to p in G.

ALGORITHM 1

Phase 1. The transmitter signs and sends its value to every other processor.

Phases 2 *to* $t + 2$. Whenever a processor in A (respectively, B) gets a correct 1-message for the first time, it signs and sends this message to everybody in B (respectively, A).

Decision Function: A processor in A or B decides that the value is 1 if by phase $t + 2$ it has received a correct 1-message; otherwise, it agrees on value 0.

THEOREM 3. *For $n = 2t + 1$ Algorithm 1 is a $(t + 2)$-phase authenticated algorithm for reaching Byzantine Agreement among n processors with at most t faults that does not require sending more than $2t^2 + 2t$ messages.*

PROOF. We first prove the correctness of Algorithm 1. If the transmitter correctly sends value 1 at the first phase, then every processor gets a correct 1-message at phase 1 and each correct one will decide value 1 according to the decision function. If, on the other hand, the transmitter correctly sends value 0 at the first phase, no processor will ever be able to send a correct 1-message; therefore, the correct processors will agree on value 0.

There remains the case in which the transmitter is faulty and sends different values to different processors. If, in this case, a correct processor p agrees on value 1, p must have received a correct 1-message the first time at some phase $k \leq t + 2$. Assume first that $k \leq t$. At phase $k + 1$, p will send a correct 1-message to every processor in the set not containing p; let us assume that this is A. At least one processor z of the t processors in A must be correct, since we assumed the transmitter to be faulty and the total number of faulty processors is bounded by t. Now z will send a correct 1-message to every processor in B at phase $k + 2$, and hence every processor has got a correct 1-message by phase $t + 2$.

If p receives the first correct 1-message at phase $t + 1$ or $t + 2$, then among the first $t + 1$ processors having signed this message there must be at least one correct processor p'. Processor p' must have received a correct 1-message by phase t. Therefore, this case reduces to the former one.

The transmitter has to send exactly $2t$ messages. Each of the remaining $2t$ processors can correctly send at most one message to t other processors. Therefore, the number of messages is bounded by $2t^2 + 2t$. □

The second algorithm is an extension of the first one. It has $2t + 1$ additional phases. Let the processors be $p(1), \ldots, p(2t + 1)$. We call a message that is received by some processor $p(j)$ after phase $t + 2$ *increasing* if it consists of the value to which $p(j)$ has committed in phase $t + 2$ together with signatures of processors with labels less than j in an increasing order.

ALGORITHM 2

Phases 1 *to* $t + 2$. Run Algorithm 1 and decide on the common value.

Phase $t + 2 + j$ *for* $1 \leq j \leq 2t + 1$. Let $m(j)$ be one of the increasing messages $p(j)$ has received so far with a maximum number of signatures appended to it. If it has not received any message, then $m(j)$ is only the common value. Processor $p(j)$ signs $m(j)$ and, if $m(j)$ carries at least t signatures, $p(j)$ sends this message to every other processor, else only to processors with labels between $j + 1$ and $j + t + 1$.

THEOREM 4. *For* $n = 2t + 1$ *Algorithm* 2 *is an authenticated algorithm that reaches Byzantine Agreement among n processors with at most t faults such that after* $3t + 3$ *phases each processor possesses a message containing the common value with at least t different signatures of other processors appended to it. No processor can have such a message with a value different from the common value. The algorithm requires sending at most* $5t^2 + 5t$ *messages.*

PROOF. After phase $t + 2$, each of the processors with labels between 1 and t sends at most $t + 1$ messages to other processors, whereas the remaining ones might send up to $2t$ each. This adds up to

$$2t^2 + 2t + t(t + 1) + (t + 1)2t = 5t^2 + 5t.$$

It is easy to see that faulty processors cannot produce a message with $t + 1$ signatures on a value other than the correct one. It remains to be shown that by phase $3t + 3$ each processor has received the common value with at least t other signatures appended.

Let $p(i_1), \ldots, p(i_r)$ be the sequence of correct processors ordered by their labels ($r \geq t + 1$). Since between two succeeding ones there can be at most t faulty processors, one can easily prove that for any $1 < j \leq r$, $p(i_j)$ receives from $p(i_{j-1})$ the common value with at least the $j - 1$ signatures of $p(i_1), \ldots, p(i_{j-1})$ by phase $t + 2 + i_j$. (A faulty processor signing an increasing message obviously does not hurt the correctness of the algorithm.) Therefore, $p(i_r)$ will send an increasing message with at least $t + 1$ different signatures to every other processor. □

Now assume that n is bigger than $2t + 1$ and that the processors are ordered in some arbitrary way starting with the transmitter. Let α be the smallest quadratic number bigger than $6t$. If n happens to be smaller than α as in [9], one can extend the first Algorithm by 1 phase and $(t + 1)(n - 2t - 1) = O(t^2)$ messages and still achieve an $O(n + t^2)$ upper bound. Therefore, assume $n \geq \alpha$. We first describe a simple algorithm for the case $n \geq t^3$.

Let us call the first $2t + 1$ processors including the transmitter *active processors* and the $m := n - (2t + 1)$ remaining ones *passive.*

Divide the passive processors into r disjoint sets of size s, where $r = \lceil m/s \rceil$. The algorithm is parameterized by the size s of these sets. Every set has a "root," which will receive the value from the active processors. Every processor knows the other members of its set. For a set C, let $c(1), c(2), \ldots, c(s)$ be the elements in C, where $c(1)$ is the root.

The algorithm given below informs the processors in each set about the value on which the active processors have agreed on while running Algorithm 1. Within the algorithm we call this value the *correct* value.

ALGORITHM 3

Phases 1 *to* $t + 2$. The active processors run Algorithm 1.

Phase $t + 3$. Each active processor sends the correct value to the root $c(1)$ of each set C. Each root defines message $m(1)$ to be that (unique) value v it received from at least $t + 1$ active processors.

For each set C and for every $2 \leq j \leq s$:

Phase $t + 2j$. $c(1)$ sends $m(j - 1)$ to $c(j)$.

Phase $t + 2j + 1$. If at the previous phase processor $c(j)$ has received exactly one value from its root with possibly some signatures of processors $c(2), \ldots, c(j - 1)$ appended to it, it signs this message and returns it to $c(1)$. If $c(1)$ receives $m(j - 1)$ back from $c(j)$ with the signature of $c(j)$ appended, it defines this message to be $m(j)$; else, $m(j) := m(j - 1)$.

Phase $t + 2s + 2$. $c(1)$ sends $m(s)$ to every active processor.

Phase $t + 2s + 3$. For each active processor p and each set C, let $m(p, C)$ be the message p received from the root of C at the previous phase. Active processor p sends the correct value to every processor $c(j)$ $(1 < j \leq s)$ for which $m(p, C)$ does not contain the correct value and the signature of $c(j)$ appended to it.

Decision Function. An active processor agrees according to Algorithm 1. A root of a set agrees on $m(1)$ as defined in phase $t + 3$. If in the last phase a processor $c(j)$ with $j > 1$ receives a value v from at least $t + 1$ active processors, it agrees on v; otherwise, it agrees on that value that it received from its root at phase $t + 2j$.

LEMMA 1. *Algorithm* 3 *is an authenticated algorithm that reaches Byzantine Agreement among n processors with at most t faulty processors within* $t + 2s + 3$ *phases by sending at most* $2n + 4tn/s + 3t^2s$ *messages.*

PROOF. The correctness of Algorithm 3 is evident from the following fact.

Fact. If the root of a set C is correct (a correct processor) then, for each j, $1 \leq j \leq s$, $m(j)$ contains the correct value and $m(s)$ contains the signature of each correct processor in C (except the root) and will be received by each active processor at phase $t + 2s + 2$.

Therefore, if a correct processor $c(j)$, $j > 1$, of a set C receives a value from at least one correct active processor at the last phase, then the root of C must be incorrect. If $c(j)$ gets a value from at least $t + 1$ active processors at the last phase, at least one of them must be correct; hence that value must be the correct value. Otherwise, there must be at least one correct active processor p such that $m(p, C)$

contains the correct value and the signature of $c(j)$. This implies that the message $m(j-1)$ sent from $c(1)$ to $c(j)$ contains the correct value and $c(j)$ will agree on it.

The number of messages sent by correct processors does not exceed $2t^2 + 2t$ for phases 1 to $t+2$, $(2t+1)r$ for phase $t+3$, $2(m-r)$ for phases $t+4$ to $t+2s+1$, $(2t+1)r$ for phase $t+2s+2$, and $(2t+1)t(s-1)$ for the last phase. The last bound holds because there are at most t sets with incorrect roots, each one missing at most $s-1$ signatures of processors in that set. Altogether this can be bounded by $3t^2s + 2n + 4tn/s$. □

THEOREM 5. *There exists an authenticated algorithm that reaches Byzantine Agreement among n processors with at most t faults while sending at most* $O(n + t^3)$ *messages.*

PROOF. Lemma 1 implies that, by choosing $s = 4t$ in Algorithm 3, the number of messages is bounded by $O(n + t^3)$. □

6. *An* $O(n + t^2)$ *Algorithm for the General Case*

In this section we present a more complicated algorithm for a further reduction of the number of messages. If n is sufficiently smaller than t^3 and the active processors behave as in Algorithm 3, they would send more than $O(n + t^2)$ messages in the worst case. In this situation Algorithm 5 (described below) will match the lower bound. This algorithm is more intricate and requires sending long messages. Such an algorithm may be of less practical interest, but at least it shows that for any ratio between correct and incorrect processors the lower bound of Theorem 2 is tight.

First let us consider a slightly different problem. Assume that there are $N = m^2$ processors; each wants to send a value to everybody else and the object is to minimize the number of messages that have to be sent. There is the obvious solution by which $N(N-1)$ messages are sent in one phase.

If, at most, t processors are faulty, this problem could be solved by the following two-phase algorithm:

Select $t+1$ processors; they will play the role of relay processors. At phase 1 each processor signs and sends its value to every relay processor. A relay processor combines all the incoming messages and its own value to one long message and sends it to every nonrelay processor at phase 2.

This procedure requires sending at most $(N-1)(t+1) + (N-t-1)(t+1)$ messages. It is easy to show that $\Omega(Nt)$ is also a lower bound for the number of messages in case each correct processor is required to receive the value of every other correct processor. If only a high percentage of correct processors are required to exchange their values, then, as Algorithm 4 shows, one can substantially save in the number of messages for small values of N.

Denote the m^2 processors by $p(i, j)$, where $1 \le i, j \le m$, and let $M(i, j)$ be the value of $p(i, j)$. We say that a message has a *correct format* if it contains a value followed by signatures, as required by the algorithm. A processor ignores messages that do not have a correct format.

ALGORITHM 4

Phase 1. $p(i, j)$ signs and sends its value $M(i, j)$ to $p(i, k)$ for all $1 \le k \le m$. Define $M1(i, j, k)$ to be the message $p(i, j)$ received from $p(i, k)$, if the message does not have a correct format (a value signed by $p(i, k)$) define $M1(i, j, k)$ to be the empty string.

Phase 2. $p(i, j)$ sends $[M1(i, j, 1), \ldots, M1(i, j, m)]$ to $p(l, j)$ for all $1 \le l \le m$. If the message $p(i, j)$ received from $p(l, j)$ at this phase has a correct format (a list of up to m

strings in which each string is a value signed by one of the processors $p(l, 1), \ldots, p(l, m)$), then it defines $M2(i, j, l)$ to be this message, otherwise $M2(i, j, l)$ is the empty string.

Phase 3. $p(i, j)$ sends $[M2(i, j, 1), \ldots, M2(i, j, m)]$ to $p(i, k)$ for all $1 \leq k \leq m$. Denote by $M3(i, j)$ the set of messages $p(i, j)$ received from processors $p(i, 1), \ldots, p(i, m)$.

LEMMA 2. *If there are at most t faulty processors, then there is a set P of at least* $N - 2t$ *correct processors such that for each* $p(i, j)$, $p(l, k) \in P$ *the set of messages* $M3(i, j)$ *includes the value* $M(l, k)$ *signed by* $p(l, k)$.

PROOF. Let P be the set of all correct processors $p(i, j)$ of which the set $p(i, 1), \ldots, p(i, m)$ contains less than $m/2$ faulty processors. At most t correct processors do not appear in P; hence the size of P is at least $N - 2t$. A correct processor $p(i, j)$ receives the value $M(l, k)$ signed by a correct processor $p(l, k)$ if for some x both $p(l, x)$ and $p(i, x)$ are correct processors. This clearly holds when each one of the sets, $p(l, 1), \ldots, p(l, m)$ and $p(i, 1), \ldots, p(i, m)$ contains less than $m/2$ faulty processors. □

The number of messages sent by this algorithm is bounded by $3(m - 1)m^2$, which is smaller than Nt for $t \geq m = \sqrt{N}$. Thus we have proved the following.

THEOREM 6. *In three phases a set of N processors containing at most t faulty ones can mutually exchange values such that a set of at least* $N - 2t$ *correct processors actually succeed while sending at most* $O(N^{1.5})$ *messages.*

We now present an algorithm that achieves the bound $O(n + t^2)$ for any ratio between n and t. As before, let α be the smallest quadratic number bigger than $6t$. The first α processors will be the *active* processors. The active processors use Algorithm 4 to inform each other about passive processors that have not received the value. The set of active processors that are able to exchange their messages without being blocked by the faulty processors (the set P of Lemma 2) will be called the set of *nonisolated* processors.

The remaining $m := n - \alpha$ passive processors will be divided into complete binary trees of size $s = 2^\lambda - 1$. When we refer to a subtrees of these binary trees, we only consider those subtrees whose leaves are the leaves of the original binary tree. Within each subtree the nodes are ordered by some order starting from the root. Define $l(x) := 2^x - 1$ to be the number of processors in a binary tree of depth x.

Let a *string* be an integer (an index of the string) followed by a list of some passive processors and signed by a single active processor. Let C be a subtree of depth x and M be a message containing a set of strings. For a passive processor q, denote by $\pi(M, q, x)$ the number of different active processors p for which M contains a string signed by p with index x which is followed by a list on which q appears.

In the following algorithm the active processors run Algorithm 2 to decide on a value and then they transmit this value to the passive processors. The active processors use Algorithm 4 to decide on the set of passive processors that may have not received the value yet. Each string we use in the algorithm contains a set of passive processors that have not received the decision value yet.

To save messages, we want to prevent faulty active processors from activating correct passive processors with no reason. On the other hand, we need to activate passive processors in case that either they or their successors have not received the value yet. A root of a subtree activates itself only after receiving a set of strings that proves that either enough active processors think that the root had not received the value, or two of its successors had not.

A message M is *a proof of work* for a binary tree C of depth x if either

(i) $x = \lambda$ and M is empty, or
(ii) $x < \lambda$ and either $\pi(M, c, x) \geq \alpha - 2t$, where c is the root of C, or there exist a processor q with $\pi(M, q, x) \geq \alpha - 2t$ in the right depth$(x-1)$ subtree of C and a processor q' with $\pi(M, q', x) \geq \alpha - 2t$ in the left one.

Let W be the set of values the transmitter may send. We may assume that no value in W contains the empty set or a name of a processor as a substring. A message is called *valid* if it consists of an element in W (a value) followed by at least $t + 1$ signatures of active processors and possibly some of passive ones. The idea is that a message is valid if at least one correct processor supports its value. Notice that the outcome of Algorithm 2 are valid messages.

ALGORITHM 5

Phase 1 *to* $3t + 3$. The first $2t + 1$ active processors run Algorithm 2.

Phase $3t + 4$. Each processor of the first $t + 1$ ones sends a valid message to the remaining $\alpha - 2t - 1$ active processors.

For each active processor p denote by $B(p, \lambda)$ the set of passive processors and by $C(p, \lambda)$ be the set of the $\lceil m/s \rceil$ binary trees of depth λ.

For $x = \lambda, \ldots, 0$ **do (Start of the phases in block** x**)**

Phase 1 *of block* x. Each active processor p sends a valid message and a proof of work to the root c of each binary tree C (of depth x) in $C(p, x)$. Assume C consists of processors $c = c(1), \ldots, c(l(x))$. If the root c of a binary tree C of depth x receives a valid message m and a proof of work for C from an active processor it sets $m(1) := m$ and starts sending messages for the next $2l(x)$ phases. In this case let us call c *activated.*

For each activated root c and for every $1 \leq j < l(x)$:

Phase $2j - 1$ *of block* x. c sends $m(j)$ to $c(j + 1)$.

Phase $2j$ *of block* x. If at the previous phase processor $c(j + 1)$ has received exactly one valid message from the root of the depth x subtree it belongs to, then it signs this message and sends it back. If c receives $m(j)$ back from $c(j + 1)$ with its signature appended, it defines this message to be $m(j + 1)$, else $m(j + 1) := m(j)$.

Phase $2l(x) - 1$ *of block* x. c sends $m(l(x))$ to every active processor.

For each active processor p let $F(p, x - 1)$ to be the set of all processors in $B(p, x)$ whose signature did not appear in any valid message processor p gets back from the roots of the corresponding subtrees. $F(p, x - 1)$ does not include the roots themselves.

Phases $2l(x)$ *to* $2l(x) + 2$. The active processors run Algorithm 4 with $[F(p, x - 1), x - 1]$ as the values to be exchanged.

At the end of Algorithm 4, every active processor p determines its new set $C(p, x - 1)$ as follows:

Denote by M the set of messages p receives at phase $2l(x) + 2$. Observe that these messages are strings because of our definition. Denote by $B(p, x - 1)$ the subset of $F(p, x - 1)$ that consists of those processors q for which $\pi(M, q, x) \geq \alpha - 2t$. The set $C(p, x - 1)$ consists of all those subtrees of depth $x - 1$ for which M is a proof of work.

End of block x

7. *Correctness and Complexity of Algorithm* 5

After taking part in Algorithm 5, each processor agrees as follows:

Decision Function. Agree on the value of the first valid message received.

Correct active processors sign values from W only in the beginning when they participate in Algorithm 2. Using Theorem 4, one can easily show that any valid message must have as its value the correct value and that by phase $3t + 4$ each active processor possesses a valid message. It remains to show that each passive processor gets a valid message at least once.

LEMMA 3. *If a passive processor q has not got a valid message by the end of block x ($x = \lambda, \ldots, 1$), then processor q belongs to $B(p, x-1)$, for each nonisolated active processor p.*

PROOF. If for each nonisolated processor p a passive processor q belongs to $B(p, x)$ and q has not signed a valid message in block x, its name will still appear on the list $F(p, x-1)$ of each p. Therefore, each nonisolated processor receives q's name on at least $\alpha - 2t$ different lists at phase $2l(x) + 2$. This implies that q also belongs to $B(p, x-1)$ for each such p. □

In the last block each subtree consists of only one processor; therefore, all passive processors that have not seen a valid message yet will get it directly from the nonisolated active processors at that phase.

NUMBER OF PHASES. Since block x consists of 2^{x+1} phases for $x > 0$ and 1 for $x = 0$, the total number of phases of Algorithm 5 is at most $3t + 4s + 2$.

NUMBER OF MESSAGES. The correct active processors send at most $5t^2 + 5t + (t+1)(\alpha - 2t - 1)$ messages up to phase $3t + 4$ and, afterward, among themselves, no more than $3(\alpha - 1)\alpha^2$ messages per block.

Let C be one of the binary trees (of size s and depth λ).

LEMMA 4. *If C contains $b(C)$ faulty processors, then the number of processors in C which either get activated or are faulty is bounded by $2b(C) + 1$.*

PROOF. Let $P(C)$ be the set of activated nodes of C and consider the tree $T = T(C)$ obtained by "shrinking" C to $P(C)$. $P(C)$ is the set of nodes of T and, for each such node except that of the root of T (which equals the root of C), there is an edge to its nearest ancestor in T. T is not necessarily a complete binary tree. For a node z of T, denote by $h(z)$ the height of z in C.

We first show that each node z in T has at most two successors. If z has three successors in T, there must be a pair Ψ, Φ among them such that their nearest common ancestor w in C is a proper ancestor of z. Let p_Ψ (respectively, p_Φ) be one of the active processors that sent a valid message and a proof of work to Ψ (respectively, Φ).

There must be processors ψ and ϕ in the subtree with root Ψ (respectively, Φ) such that ψ (respectively, ϕ) appears in at least $\alpha - 2t$ sets $F(q, h(\Psi))$ (respectively, $F(q, h(\Phi))$) that are part of the proof of work that was sent from p_Ψ to Ψ (respectively, p_Φ to Φ). Among the processors signing these sets (as strings) at least $\alpha - 3t$ must be correct active processors. Since $2(\alpha - 3t) > \alpha$, at least one correct active processor p appears on both lists. This implies $\phi \in B(p, h(\Phi) + 1)$ and $\psi \in B(p, h(\Psi) + 1)$. For any active processors p and any $0 \le y < x \le \lambda$, $B(p, y)$ is a subset of $B(p, x)$; therefore, $\phi, \psi \in B(p, h(w))$. This means that p had a proof of work for the subtree with root w, which contradicts the assumption that w did not get activated.

Nodes in T may represent correct or incorrect processors. Let U be the subtree of T, which consists of all the incorrect nodes q from which the path from q to the root of T consists of incorrect nodes only. T–U forms a forest of binary trees S_i with correct roots s_i, $i = 1, \ldots, j$. Since T is binary, j cannot exceed the size of U by more than 1.

After the root s of such a tree S has been activated in block $h(s)$, each correct node q of S receives the value from s, signs it and every active processor gets the signature of q from s. Hence q does not belong to the set $F(p, h(s) - 1)$ for any

correct active p, and the sets $B(p, y)$ for $y < h(s)$ contain only incorrect nodes of S. Therefore, a proof of work for a subtree of S can only rely on incorrect processors of S. This implies that the number of correct nodes in S that are different from s is less than the number of incorrect nodes in S.

The number of correct nodes in T can therefore be bounded by $1 + |U|$ plus the number of incorrect nodes in each S_i. Hence the number of correct nodes in C that get activated plus the number of incorrect nodes in C does not exceed $2b(C) + 1$. □

This bounds the number of messages between active processors and passive processors in C of which the sender is correct by $2\alpha(2b(C) + 1)$.

In addition the sum of the sizes of all subtrees of C with an activated or faulty processor as the root is bounded by $s(1 + \log(2b(C) + 1))$. Then the number of messages among processors in C with a correct sender is at most twice as much. This gives a total of $2\alpha(2b(C) + 1) + 2s(1 + \log(2b(C) + 1))$ for each tree C.

To estimate the total number of messages, we have to sum over the $r = \lceil m/s \rceil$ binary trees. Recall that the total number of faulty processors is bounded by t, which implies that the summation of $\log(2b(C) + 1)$ over all the binary trees cannot exceed $t \log 3$. Hence the total number of messages sent by correct processors in Algorithm 5 is bounded by

$$O(t^2) + O(t^{1.5} \log s) + O(tn/s).$$

Therefore, we have proved the following:

LEMMA 5. *Assume* $1 \leq s \leq t < n/6$. *Then Algorithm 5 achieves Byzantine Agreement among n processors with at most t faults in at most* $3t + 4s + 2$ *phases while sending no more than* $O(t^2 + nt/s)$ *messages.*

THEOREM 7. *There exists an authenticated algorithm that reaches Byzantine Agreement among n processors with at most t faults while sending at most* $O(n + t^2)$ *messages.*

PROOF OF THEOREM 7. Lemma 5 implies that when we choose $s = t$ in Algorithm 5 the number of messages is bounded by $O(n + t^2)$. □

REFERENCES

1. DEMILLO, R. A., LYNCH, N. A., AND MERRITT, M. J. Cryptographic protocols. In *Proceedings of the 14th Annual ACM Symposium on the Theory of Computing* (San Francisco, Calif., May 5–7). ACM, New York, 1982, pp. 383–400.
2. DIFFIE W., AND HELLMAN, M. New direction in cryptography. *IEEE Trans. Inform. IT-22*, 6 (1976), 644–654.
3. DOLEV, D. Unanimity in an unknown and unreliable environment. In *Proceedings of the 22nd Annual Symposium on Foundations of Computer Science*. IEEE, New York, 1981, pp. 159–168.
4. DOLEV, D. The Byzantine generals strike again. *J. Algorithms 3*, 1 (1982), 14–30.
5. DOLEV, D., AND REISCHUK, R. Bounds on information exchange for Byzantine Agreement. In *Proceedings of the ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing* (Ottawa, Ontario, Canada, Aug. 18–20). ACM, New York, 1982, pp. 132–140.
6. DOLEV, D., AND STRONG, H. R. Polynomial algorithms for multiple processor agreement. In *Proceedings of the 14th ACM SIGACT Symposium on Theory of Computing* (San Francisco, Calif., May 5–7). ACM, New York, 1982, pp. 401–407.
7. DOLEV, D., AND STRONG, H. R. Distributed commit with bounded waiting. In *Proceedings of the 2nd Symposium on Reliability in Distributed Software and Database Systems* (Pittsburgh, Pa., July). IEEE Computer Society, 1982, pp. 53–60.
8. DOLEV, D., AND STRONG, H. R. Requirements for agreement in a distributed system. In *Proceedings of the Second International Symposium on Distributed Data Bases* (Berlin). 1982, pp. 115–129. Also in *Distributed Data Bases*, H. J. Schneider, Ed. North Holland, Amsterdam, 1982.

9. DOLEV, D., AND STRONG, H. R. Authenticated algorithms for Byzantine Agreement. *SIAM J. Comput. 12* (1983), 656–666.
10. DOLEV, D., FISCHER, M., FOWLER, R., LYNCH, N., AND STRONG, R. Efficient Byzantine Agreement without authentication. *Inf. Cont. 3* (1983), 257–274.
11. FISCHER, M., AND LYNCH, N. A lower bound for the time to assure interactive consistency. *Inf. Process. Lett. 14*, 4 (1982), 183–186.
12. FISCHER, M., FOWLER, R., AND LYNCH, N. A simple and efficient Byzantine generals algorithm. In *Proceedings of the Second Symposium on Reliability in Distributed Software and Database Systems* (Pittsburgh, Pa., July). IEEE Computer Society, 1982.
13. LAMPORT, L. The weak Byzantine generals problem. *J. ACM 30*, 3 (July 1983), 668–676.
14. LAMPORT, L., SHOSTAK, R., AND PEASE, M. The Byzantine generals problem. *ACM Trans. Program. Lang. Syst. 4*, 2 (July 1982), 382–401.
15. PEASE, M., SHOSTAK, R., AND LAMPORT. L. Reaching agreement in the presence of faults. *J. ACM 27*, 2 (Apr. 1980) 228–234.
16. RIVEST, R. L., SHAMIR, A., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM, 21*, 2 (Feb. 1978) 120–126.

RECEIVED SEPTEMBER 1982; REVISED JUNE 1984; ACCEPTED JULY 1984