BOUNDS ON INFORMATION EXCHANGE FOR BYZANTINE AGREEMENT

Danny Dolev
Ruediger Reischuk

IBM Research Laboratory
San Jose, CA 95153

Abstract

Byzantine Agreement has become increasingly important in establishing distributed properties when there may exist errors 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 we study the amount of information exchange necessary to ensure Byzantine Agreement. This is measured by the number of messages and the number of signatures appended to messages (in case of authenticated algorithms) the participating processors need to send, in the worse case, in order to reach Byzantine Agreement. The lower bound for the number of signatures in the authenticated case is $\Omega(nt)$, where n is the number of participating processors and t is the upper bound on the number of faults. If n is large compared to t, it matches the upper bounds from previously known algorithms. The lower bound for the number of messages is $\Omega(n+t^2)$. We present an algorithm that achieves this bound and for which the number of phases does not exceed the minimum t+1 by more than a constant factor.

1. INTRODUCTION

Reaching agreement in distributed system is essential for maintaining coordination and synchronization among the participating processors. 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.

The type of agreement we study is called Byzantine Agreement (LSP) and it is achieved when:

(I) all correctly operating processes agree on the same value, and

(II) if the transmitter operates correctly, then all correctly operating processes agree on its value.

Several algorithms for obtaining Byzantine Agreement have been published (PSL), (LSP), (L), (Da), (Db), (DSa), (DSb), (FFL), (DFFLS). Without using authentication the best algorithm is the one presented in (DFFLS), in which the agreement is achieved within 2t+3 phases while exchanging, in the worst case, $O(nt + t^3 \log t)$ bits of information. The best solution using authentication is presented in (DSb), it requires t+1 phases and $O(nt+t^2)$ messages, where each message may contain several signatures. The maximum number of signatures required is $O(nt^2+t^3)$ for t+1 phases and $O(nt+t^3)$ for t+2 phases. Previous papers give lower bounds either on the ratio between correct and faulty processors (La) or on the number of phases (LF), (DSb), (DLM), and (DSc).

In this paper we concentrate on algorithms using authentication, and analyze how many messages

have to be exchanged to reach Byzantine Agreement. We will 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 which 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 $t+2+t/\alpha$ phases and $O(\alpha n)$ messages for $1 \leq \alpha \leq t$. The solution presents a tradeoff between the number of messages and phases.

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 in these cases a simple error correction code instead of signatures can be used and the results are applicable.

The lower bound on the number of signatures implies a lower bound on the number of messages in the unauthenticated case. This lower bound shows that when $n \geq t^2$ the algorithm in (DFFLS) is best possible to within a constant factor in this respect.

2. HISTORIES

We first review some notions from (DSa).

A phase is a directed graph with nodes corresponding to processors and with labels on the edges. A label represents the information sent from a given processor to another during the given phase. We assume that when no message is sent there is no edge. An n processor history is a finite sequence of n node phases, with nodes labelled by the names of the processors, together with a special initial phase called phase 0, such that phase 0 contains only a single inedge to one processor called the sender. (The assumption is that the inedge at phase 0 carries the value that the sender is to send.)

A subhistory of a history H is a copy of H with some edges removed. For each history H and processor p there is a unique subhistory **pH** called the subhistory of H according to p, consisting of only the edges with target p. Thus, the subhistory according to the sender includes the value it is supposed to send even if it sends nothing. Note that the subhistory pH is all that processor p has to work with, it cannot have any other information about the states of other processors.

An agreement algorithm on a class of histories C consists of a correctness rule **R** (a function which given a subhistory according to p and an edge in a phase to be added to the history as the next phase, produces a possibly empty label for that edge) and a decision function **F** (a function from subhistories according to processors of histories in C to the union of V with a special symbol representing "sender fault"). With respect to a given correctness rule, a processor p is said to be correct at phase k if each edge from p in phase k has the label produced by the correctness rule operating on the previous k-1 phases of the subhistory according to p. A processor p is correct for history H if it is correct at each phase of H. We call a history t-faulty (with respect to a correctness rule) if at most t of its processors are incorrect.

A correctness rule is actually a union of possibly distinct correctness rules, one for each processor. Likewise, the decision function is a union of individual decision functions.

We say Byzantine Agreement can be achieved for n processors with at most t faults within d phases if there is an agreement algorithm for the set C of n processor, t-faulty, d phase histories so that the decision function F obeys the rules for Byzantine Agreement:

(I) if processors p and q are correct for H in C then $FpH = FqH$, and

(II) if the sender is correct at the first phase of H and processor p is correct for H in C then $FpH = v$ where v is the sender's value.

Note that the class C of histories is assumed limited to those consistent with the semantics of authentication.

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 assume that the processors 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 alter the content of the message or the signature undetectably. Such a scheme is the one suggested in (DH) and (RSA), and the use of it for Byzantine algorithms is described in (DSa), (DSb), (LPS), and (PLS).

We allow faulty processors to collude for cheating, therefore we can assume that every message that contains only signatures of faulty processors can be produced by these processors. This is a worst case assumption, and it will be assume in the lower bound proofs. The assumption will be extended to assume that every subhistory in which only faulty processors are involved, can be produced by every faulty processor at any time.

There exists an algorithm to reach Byzantine Agreement without using signatures; therefore, the lower bound is meaningless unless we count somehow the messages that do not contain signatures. We make the technical assumption that every message in the authenticated algorithm carries at least the signature of its sender. Alternatively, the lower bound can count 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 H 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 sends the value 0, and G the one in which all are correct and the transmitter is sending 1. If the sum of the number of signatures each correct processor receives and the number of processors receiving its signature in both histories together is at least $t+1$, then the theorem holds.

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. Assume 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 $A(p)$ behave towards p as in H and towards 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' processor p sees the same subhistory as in H, which implies that $FpH'=FpH=0$, while all other correct processors q see the subhistory they saw in G and hence $FqH'=FqG=1$. Notice that there are other correct processors since we assumed $t<n-1$. This violates condition (I) of Byzantine Agreement. Therefore, there cannot be any processor which "exchanges" in H and G altogether at most t signatures with other processors. □

If authentication is not available this lower bound applies directly to the 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 history H 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 there is no way for a processor z different from p and q to 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 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 AUTHENTICATED CASE

Sometimes the overhead for sending a message costs more than the message itself; and therefore, it makes sense to find algorithms which minimize the number of messages. Since a message that has several different signatures appended can contain a lot 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.

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

Now assume that the maximum is achieved by the second term. Let B be a subset of Q of size $1+t/2$ 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 since efficient authenticated algorithms tend not to be homogeneous. But by playing with histories we will 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 at least $1+t/2$ messages to it.

Let H' be the following history:
Every processor in A is correct, the transmitter sends the value 0 . Each processor q in B never sends a message to other processors in B. Towards processors in A processor q behaves like a correct processor with one exception, it ignores the first $t/2$ messages received from processors in A, all of them if it gets fewer than $t/2$. This defines a valid history with $1+t/2$ faults 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 $t/2$ 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" : make p correct, and all the processors in A(p) incorrect. They behave like correct processors except for not sending any message 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 towards 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 t/2 messages it was supposed to get there is also no difference between the behavior of p towards processors in A in H' and H". Therefore, each correct processor other than p sees the same subhistory in both cases and at the end he must agree on value 0 . But the correct processor p receives in H" no messages at all, therefore by definition he doesn't agree on 0 . This leads to a contradiction, which proves that in H' every processor in B must receive at least 1+t/2 messages from correct processors. □

5. DECREASING THE NUMBER OF MESSAGES IN THE AUTHENTICATED CASE

None of the known authenticated algorithms makes use of authentication to substantially reduce the number of messages required to be sent in the case where n is much larger than t. In this case the best known algorithms, with and without authentication, require $\Theta(nt)$ messages (DSb,DFFLS) in the worst case. However for large n Theorem 2 only gives a linear lower bound, and the $\Omega(nt)$ lower bound in Theorem 1 only holds for the total number of signatures that have to accompany messages. Since we can append a lot of 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 this algorithm needs does not exceed the minimal number t+1 by more than a small constant factor. Of course in the worst case many messages have to carry $\Omega(t)$ signatures.

We may assume that n is at least 2t+1 , otherwise the algorithm in (DSb) sends a number of messages that is of the same order as the lower bound in Theorem 2. First 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 has the nice feature that at the end not only does every correct processor agree on the same value, but it also has a one-message proof for the outside world of what this value is. In practice this means that it possesses a string which says what the common value is and this statement is signed by at least t other processors.

In the following algorithms we assume that the processors have to decide whether the transmitter has sent value 0 or 1 . If the transmitter can send more than two values one has to modify the algorithms slightly. We also assume that whenever a processor sends a message to someone else, it appends its signature to it before sending. Throughout this section we assume that all processors are completely synchronized.

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 that graph which is formed by the complete bipartite graph with the set of nodes A,B. In addition connect node q to each node in A or B. We call a message a processor p receives at phase k, for k=1,...,t+2 <u>correct</u> if it consists of a value with signatures appended to it and the sequence of processors that signed that message together with p form a simple path of length k from q to p in G.

Algorithm 1:

Phase 1:

The transmitter sends its value to every other processor.

Phases 2 to t+2:

Whenever a processor in A (B) gets a correct message containing for the first time the value 1 it sends this message to everybody in B (A).

Decision function:

A processor in A or B decides that the value is 1 if by phase t+2 it has got a correct message with value 1, otherwise it agrees on value 0 .

Theorem 3: For n=2t+1 Algorithm 1 is a t+2 -phase authenticated algorithm to reach Byzantine Agreement among n processors with at most t faults that does not require sending more than $2t^2+2t$ messages. □

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 which 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 increasing order.

Algorithm 2:

Phase 1 to t+2:

Run algorithm 1 and decide on the common value.

For $1\leq j\leq 2t+1$ phase t+2+j :

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 its value. If m_j carries at least t signatures p_j sends this message to every other processor, else only to processors with a label 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 correct processor has received the common value together with at least t signatures of other processors appended to it. The algorithm requires sending at most $5t^2+5t$ messages. □

We will now consider the general case n>2t+1 . Arbitrarily choose a set A of 2t+1 processors that includes the transmitter to be the set of active processors and let B be the m:=n-(2t-1) remaining processors, called passive. The algorithm we will describe consists of two parts: in the first the active processors agree on a value, using one of the previous algorithms, and then in the second they use the following algorithm to share the agreement with the passive processors.

Divide the passive processors into r disjoint subsets with s processors each - except possibly the last one - where $r=\lceil m/s\rceil$. The algorithm is parameterized by the size s of each subset. Fix s to be $2^k-1\leq t$ for some natural number k; thus, the size of each subset is bounded by the upper bound on the number of faults. Arrange all the processors in each subset of B into a binary tree of depth k. We assume that this construction is known to each of the n processors.

We will now describe how the processors in B can be informed about a value v on which the active processors in A have agreed upon using the previous algorithm. A message is called valid if it consists of a possible value the transmitter might have sent with at least t+1 signatures of active processors appended to it. Thus, no processor can ever send a valid message containing a value different from the value agreed upon.

To make the analysis easier the actions of the algorithm will be grouped together in certain blocks. Each action can take place only at a certain phase in a certain block of phases. This rigidity which means that the algorithm cannot terminate earlier is not essential. It can be re-

laxed such that in most cases only a few additional phases are necessary.

Algorithm 3:

For each $p \in A$ define $B(p,k)=B$ and $C(p,k)$ as the set consisting of the r binary trees defined above.

For $x=k,\ldots,0$ do
(start of the phases in block x)

$\ell(x) := 2^x-1$ denotes the number of processors in a binary tree of depth x.

Phase 1: Each $p \in A$ sends a valid message to the root of each binary tree C of depth x in $C(p,x)$.
If a processor $q \in B$ that is a root of a binary tree C of depth x has just received a valid message from at least t+1 processors in A it defines this message to be message m(1) and becomes active for the next $2\ell(x)-1$ phases. Let $q_1,\ldots,q_{\ell(x)-1}$ be an arbitrary ordering of the processors in the subtree, excluding the root q itself.

For each active root q and for every $1 \leq y \leq \ell(x)-1$:

Phase 2y: q sends m(y) to q_y.

Phase 2y+1: If at phase 2y a processor $q_y \in C$ has just received a valid message from q, where q is its unique ancestor at height x, with some or no signatures of processors in C appended to it, it signs this message and returns it to q .

If q receives m(y) back from q_y with the signature of q_y appended to it, it defines this message to be m(y+1), else m(y+1):=m(y) .

Phase $2\ell(x)$: q sends $m(\ell(x))$ to every processor in A .

Each processor $p \in A$ defines the set $B(p,x-1)$ to be the set of all the processors in $B(p,x)$ whose signature did not appear in any valid message p got back from roots of subtrees of depth x. The set $B(p,x-1)$ does not include those roots themselves.
$C(p,x-1)$ consists of all binary subtrees with roots in $B(p,x-1)$.

end (end of block x)

end of Algorithm 3

Correctness: Any valid message must contain the value the processors in A have agreed upon using Algorithm 2. To prove the correctness of Algorithm 3 all we have to do is to make sure that each processor in B gets a valid message at least once. This follows easily from the following:

Claim 1: If a processor q in B has not got a valid message by the end of block x ($x=k,\ldots,1$), then for all correct processors $p \in A$ processor q is still in $B(p,x-1)$.

Observe that Claim 1 holds regardless how many processors in B are faulty: it may be more than t . In this case as we will see only the number of messages will increase.

In the last block each subtree is the processor itself; therefore, all processors in B that have not seen a valid message yet will get it directly from the correct processors in A at that phase.

Number of phases: Since block x consists of $2(2^x-1)$ phases for $x>0$ and 1 for $x=0$ the total number of phases of Algorithm 3 equals $4s-1-2\log(s+1)$.

Number of messages:

Claim 2: Let C be one of the binary trees (of size s and depth k). If C contains b(C) faulty processors, then the sum of the sizes of all the subtrees of C with a faulty processor as root is bounded by slog(b(C)+1).

Observe that during the algorithm a correct processor in C receives at most one value from a correct root. Therefore, the total number of messages sent by correct processors within C is bounded by 2(slog(b(C)+1) + s).

Claim 3: The number of subtrees in C the roots of which receive a valid value from at least one processor in A is bounded by 2b(C)+1.

Therefore, the total number of messages involving correct processors of a given binary tree is bounded by

$$(2t+1)(2b(C)+1) + 2(s\log(b(C)+1) + s).$$

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(b(C)+1) over all the binary trees is also bounded by t. Hence the total number of messages sent by correct processors in Algorithm 3 is bounded by

$$2t(2t+1) + (2t+1)r + 2st + 2sr \leq 6t^2 + (2t+1)m/s + 2m + O(t).$$

Combining Algorithms 2 and 3 we get a k-phase Byzantine Algorithm, where $k \leq 3t+4s$, that sends at most $2n + (2t+1)m/s + 11t^2 + O(t)$ messages.

If $t^3=o(n)$ Algorithm 1 instead of 2 can be used and Algorithm 3 can be simplified. In this modified version a processor considers a message valid even if it carries fewer than t+1 signatures. A root of a tree of size s forwards a value if it receives it from at least t+1 processors in A. Other processors only sign the first valid message they get from their root. All we have to do is to run the phases of block k and one additional phase, in which each processor p in A directly informs each processor q in B whose signature appended to the correct value p has not been received from the corresponding root for q . The number of messages of this type is bounded by $(2t+1)t(s-1) = o(n)$, since there are at most t cycles each missing at most s-1 signatures. Now the processors in B use the following decision function:

If in the last phase a processor q receives a value v from at least t+1 processors in A , it agrees on v , otherwise it agrees on the first value it has received from its root.

It can easily be seen that each correct processor in B agrees on the same value as the correct processors in A . Algorithms 1 and 3 give a total of t+2s+3 phases and $2n+(2t+1)n/s+o(n)$ messages. Thus we have

Theorem 5: *For any $1 \leq s \leq t < n$ there is an authenticated algorithm to reach Byzantine Agreement among n processors with at most t faults in at most (3t+4s) phases sending no more than $2n+(2t+1)n/s+11t^2+o(n)$ messages.*
If $t^3=o(n)$ the number of phases can be reduced to t+2s+3. □

The above theorem presents a tradeoff between phases and messages, by fixing s to be t one minimizes the number of messages, which then becomes $4n+11t^2+o(n)$.

6. REFERENCES

[DH] W. Diffie and M. Hellman, "New direction in cryptography", IEEE Trans. on Inform. IT-22,6(1976), 644-654.

[DLM] R. A. DeMillo, N. A. Lynch, and M. Merritt, "Cryptographic Protocols", proceedings, the 14th ACM SIGACT Symposium on Theory of Computing, May, 1982.

[Da] D. Dolev, "The Byzantine Generals Strike Again", Journal of Algorithms, vol. 3, no. 1, 1982.

[Db] D. Dolev, "Unanimity in an Unknown and Unreliable Environment", 22nd Annual Symposium on Foundations of Computer Science, pp. 159-168, 1981.

[DFFLS] D. Dolev, M. Fischer, R. Fowler, N. Lynch, and R. Strong, "Efficient Byzantine Agreement Without Authentication", IBM Research Report RJ3428 (1982).

[DSa] D. Dolev and H. R. Strong, "Polynomial algorithms for multiple processor agreement", proceedings, the 14th ACM SIGACT Symposium on Theory of Computing, May 1982.

[DSb] D. Dolev and H. R. Strong, "Authenticated Algorithms for Byzantine Agreement", IBM Research Report RJ3416 (1982).

[DSc] D. Dolev and H. R. Strong, "Distributed Commit with Bounded Waiting", Proceedings, Second Symposium on Reliability in Distributed Software and Database Systems, Pittsburgh, July 1982.

[DSd] D. Dolev and H. R. Strong, "Requirements for Agreement in a Distributed System", Proceedings, the Second International Symposium on Distributed Data Bases, Berlin, Sep. 1982.

[FFL] M. Fischer, R. Fowler, and N. Lynch, "A Simple and Efficient Byzantine Generals Algorithm", Proceedings, Second Symposium on Reliability in Distributed Software and Database Systems, Pittsburgh, July 1982.

[FL] M. Fischer, and L. Lamport, private communication of paper in preparation, April, 1982.

[La] L. Lamport, "The Weak Byzantine Generals Problem", JACM, to appear.

[LSP] L. Lamport, R. Shostak, and M. Pease, "The Byzantine Generals Problem", ACM Trans. on Programing Languages and Systems, to appear.

[LF] N. Lynch, and M. Fischer, "A Lower Bound for the Time to Assure Interactive Consistency", Information Processing Letters, to appear.

[PSL] M. Pease, R. Shostak, and L. Lamport, "Reaching Agreement in the Presence of Faults", JACM, vol. 27, no. 2, pp. 228-234, 1980.

[RSA] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems", Comm. ACM 21(1978), 120-126.