# BYZANTINE AGREEMENT

by H. R. Strong and D. Dolev*

IBM Research Laboratory, K55/281
San Jose, CA 95193

## ABSTRACT

Byzantine Agreement is a paradigm for problems of reliable consistency and synchronization in distributed systems. This paper is a survey of recent work on algorithms and requirements for reaching Byzantine Agreement. The name Byzantine is applied to this work because no assumption is made about the behavior of faulty components except to quantify the maximum number of independent failures handled. Thus algorithms for Byzantine Agreement set a new standard for reliability and availability of distributed systems in an environment of potentially unreliable components. While there are many open questions and much implementation work still remains, the materials surveyed here have established that this standard can be met in an efficient and economical way and that such standards should be sought in all areas of distributed systems.

## INTRODUCTION

"Reaching agreement in the presence of faults" is the title of the paper by Pease, Shostak, and Lamport that introduced a new standard into the literature on reliable distributed systems in the guise of a military application: the problem of the Byzantine generals.[19] Rather than speak of generals and messengers here, we will abstract the essentials of the problem and consider a network of **n processors** capable of exchanging **messages** over bidirectional **links. The problem is for all of the** processors to **agree** on the contents (**value**) of a message being sent by one of them. It becomes an interesting problem in an environment containing potentially faulty processors or links. The term **Byzantine** (coined by Leslie Lamport) is applied to the problem when we make **no assumption about the behavior of faulty components.** Thus an algorithm for reaching Byzantine agreement must cope with processors or links that could fail to relay a message as intended or even lie about its contents.

*Current address:
D. Dolev
Hebrew University, Givat Ram
91904 Jerusalem, Israel

Byzantine Agreement results when, in the presence of undetected faulty processors, all **correct** (non-faulty) processors are able to agree either on a value or on the conclusion that the originator of the value is faulty. More explicitly, **Byzantine Agreement is achieved when**
**(I) all correct processors agree on the same value, and**
**(II) if the sender is correct, then all correct processors agree on its value.**

Implicit in (I) and (II) is the idea that there must be some time by which each of the processors has completed the execution of its algorithm for reaching agreement, and that this time must be known by all processors.

A typical algorithm for reaching Byzantine Agreement is parameterized by the number **t** of faulty components that it can handle. It will guarantee that processors following it will reach Byzantine Agreement provided the number of faulty components in the network does not exceed t. This parameter is a convenient measure of the **reliability** of the algorithm, so we will refer to such an algorithm as having **reliability t.**

The original military application (Byzantine Generals Problem)[17] is an example of an application of Byzantine Agreement to distributed synchronization: in the presence of suspected traitors, the generals must exchange messages in order to agree on a time to attack. The commanding general is to send the time to all; but the commander may be a traitor and send different times to different generals. The survival of the loyal generals depends on their ability to reach agreement on some time to attack (not necessarily that supplied by the commander) or on withdrawal if the commander is found to be a traitor. The application generalizes to any set of processes that must agree on a time to perform some synchronous action. One of the processes is coordinator and will decide the time; but it may be faulty, so a default time is prearranged for this case.

Other applications such as preserving the consistency of replicated data have no particular synchrony constraint. Suppose we are working in a distributed environment without distributed transaction processing. We may require a Byzantine Agreement in order to guarantee that some replicated catalog will reach eventual consistency at all sites in spite of an unreliable communication medium.

A combination of consistency and sychronization is provided by the application of Byzantine Agreement algorithms to produce an algorithm for a nonblocking distributed transaction commit.[9] In a typical two phase distributed commit, a process must remain in a **blocking** state, holding resources locked between the time that it tells the transaction coordinator that it is prepared to commit and the time that it receives a commit or abort message from the coordinator, even if the coordinator crashes. Moreover, database consistency may not be preserved if the coordinator becomes confused and sends a commit message to some and an abort message to others. In "Distributed commit with bounded waiting," the authors show how use Byzantine Agreement to overcome such coordinator confusion and to commit or abort within a fixed time after entering the prepared state. Database consistency will be preserved provided the number of faults does not exceed the reliability of the agreement algorithm.

Algorithms for Byzantine Agreement are currently being implemented in the prototype for the Highly Available Systems project at IBM San Jose Research Laboratory.

In the following sections we will discuss a simple model for Byzantine Agreement that facilitates its study, known resource requirements for reaching Byzantine Agreement including lower bounds on time and upper bounds on reliability, algorithms that give the best known performance in a variety of contexts, known tradeoffs between time and messages, and generalizations.

## A PHASE MODEL FOR BYZANTINE AGREEMENT

We assume
**(1) perfect communication:** a completely connected network with absolutely reliable links,
**(2) exact synchronization:** one clock shared by all processors, partitioning time into discrete **phases**, each longer than the longest delay due to message transmission and processing, and
**(3) authentication:** a message protocol involving unforgeable signatures that prevents one processor from lying about the contents of a message it received from another.[8]

In a typical authentication protocol [19], the transmitter appends a signature to the message to be sent. This signature contains a sample portion of the message encoded in such a way that any receiver can verify that the message is authentic and that it was sent by the sender, but no processor can forge the signature of another. Thus no processor can change the content of a message undetectably.

In subsequent sections we will discuss the weakening of each of these assumptions, but making them simplifies the isolation of the Byzantine Agreement problem from problems of connectivity and clock synchronization.

In describing the message behavior of our network of processors, we will assume that the phases are numbered and that the sender begins sending its value according to its algorithm during phase 1. We assume
**(4) known initial conditions:** each processor knows the identity of the sender and of the other participants and the beginning time of phase 1.

## REQUIREMENTS FOR BYZANTINE AGREEMENT

In their original paper, Pease, Shostak, and Lamport showed that, **without authentication, n processor Byzantine Agreement could only be achieved with reliability less than one-third n.**[19] For reliability t with $n>3t$, they provided an algorithm that achieved Byzantine Agreement after $t+1$ phases and required $O(n^{t+1})$ messages. With authentication, they provided an algorithm with the same performance ($t+1$ phases, $O(n^{t+1})$ messages) for any reliability. Subsequently, Fischer and Lynch showed that without authentication, **$t+1$ was a lower bound on the number of phases required in the worst case.**[12] A general proof (with or without authentication) of the $t+1$ worst case lower bound was found by Dolev and Strong[7,8] and independently by DeMillo, Lynch, and Merritt.[1] Moreover, Fischer and Lamport have shown that this result holds even in a restricted model in which the only way a processor can fail is to stop sending messages.[11] **Thus t faulty processors can force n processors to take $t+1$ phases to reach agreement simply by ceasing to function at inopportune times.**

A natural question that arises here concerns the case of an algorithm with reliability t handling fewer than t faults: How soon can the processors reach agreement in case there are only **f** actual faults? In order to answer this question, it is necessary to make explicit what it means to finish or **stop** the algorithm. We say that a processor has **stopped** when it has decided upon a value (for agreement) and will do no further processing or relaying of messages pertaining to this agreement. When a processor has stopped with respect to a particular agreement, it could cut all its communication links without affecting the outcome of the agreement for other processors or for itself. If all correct processors reach agreement and stop during the same phase, we say the agreement is **immediate**. Otherwise, the agreement is **eventual**. The original algorithms provided by Pease, Shostak, and Lamport were algorithms for reaching Immediate Byzantine Agreement: all processors stopped after $t+1$ phases and no processor could stop any earlier. Note that since the agreement reached by these algorithms is always immediate, each processor is provided with the additional knowledge of when each other correct processor will stop. Otherwise, a processor might know only that eventually the other correct processors would agree with its choice of value, but it might not know when the agreement would be universal.

In the context of Immediate Byzantine Agreement algorithms, the worst case lower bound turns out to be general. Dolev, Reischuk, and Strong show that, for any f and t, **any Byzantine Agreement algorithm with reliability t that reaches immediate agreement whenever there are f actual faults requires at least t+1 phases.**[6] However, **for Eventual Byzantine Agreement, the lower bound is only min(f+2,t+1).**[10]

Dolev and Reischuk have shown that there are tradeoffs between the number of phases required and the number of messages required by a Byzantine Agreement algorithm.[5] A quick summary of their lower bound results shows that any n processor Byzantine Agreement algorithm with reliability t must in the worst case require the exchange of $(n-t)(t+1)/4$ messages without authentication or $(n-t)(t+1)/4$ signatures with authentication. Thus any Byzantine Agreement requires $\Theta(nt\log(n))$ bits of information exchange in the worst case. This bound is a far cry from the upper bound $O(n^{t+1})$ provided by the original algorithms. In the next section we will discuss some of the best algorithms known and narrow the distance considerably between upper and lower bounds on the number of messages.

## OPTIMAL ALGORITHMS FOR BYZANTINE AGREEMENT

An important, though often overlooked, measure of goodness of an algorithm is simplicity. The simplest good algorithm for Byzantine Agreement with reliability t is optimal with respect to time for immediate agreement (t+1 phases). Though its $O(n^2)$ message requirement is not optimal, we discuss it here because it establishes the feasibility of actually implementing algorithms for Byzantine Agreement with reliability greater than 1. The algorithm appears in "Authenticated algorithms for Byzantine Agreement."[8] In most applications (those not concerned with deliberate sabotage) its required authentication algorithm may be replaced by error detection.

**In phase 1 of this algorithm, the sender signs and sends its value to all participants.**

**Every processor is then to wait for receipt of messages. If during phase k, a processor receives a message containing value v and signed by k distinct processors (beginning with the sender), then the receiver is to insert v into an ordered set (no duplicates) of** committed **values, and if $k<t+1$ and v is a new value and one of the first two, then the receiver is to sign and send this message to all participants during phase k+1.**

**After phase t+1, if a processor has exactly one value committed, then that is the value of the agreement; otherwise, it agrees on the default value representing sender fault.**

It is fairly easy to show that, provided there are no more than t faulty processors, the first two values committed by any correct processor (if they exist) are committed by every correct processor. Thus, if any correct processor agrees on other than the default value, then all correct processors do. However, if the sender is correct then each correct processor commits its value during phase 1 and no other value will ever be committed because no other value will ever have the sender's unforgeable signature.

Since each processor sends at most 2n messages, the worst case number of messages required by this algorithm is at most $2n^2$.

The best published algorithm for Immediate Byzantine Agreement using authentication and time optimal (t+1 phases) requires $O(nt)$ messages in the worst case, so upper and lower bounds are tight to within a constant multiple.[8] Most important is the fact that there exist algorithms requiring a number of messages that is a small polynomial in n and t. This establishes the feasibility of implementing Byzantine Agreement algorithms with high reliability. Using authentication, the reliability can be as high as the number of participants (though of course if all processors are faulty the agreement is vacuous).

Without authentication, the results are more varied and the algorithms and proofs of correctness are not nearly as simple. For very large n (compared to t) the lower and upper bounds on the number of phases required (keeping the number of messages polynomial in both n and t) coincide (t+1 for immediate agreement, min(f+2,t+1) for eventual agreement).[6] For n close to the lower bound of 3t+1, the best known upper bound on the number of phases required with a polynomial number of messages is 2t+3 for immediate or min(2f+5,2t+3) for eventual agreement.[4,6] Thus there is a gap between upper and lower bounds for small n. Reischuk has closed the gap for $n>20t$ but with a rather large polynomial number of messages.[20]

## OTHER MODELS AND GENERALIZATIONS

Other related notions of agreement that either weaken the kind of agreement required (e.g. Dolev[2]) or strengthen the assumptions made about the behavior of faulty processors (e.g. Lamport[14]) have been studied. Here we will restrict attention to studies involving weakening the fairly restrictive assumptions of our phase model.

(1) Perfect communication

Dolev has shown that in order to achieve Byzantine Agreement with reliability t, the connectivity of the network must be at least t+1 with authentication or 2t+1 without authentication.[3] Note that this connectivity is measured in terms of potentially unreliable links, while reliability is measured in terms of failed components including either links or processors. Lamport, Shostak, and Pease showed that t+1 connectivity was sufficient using an exponential number of messages.[17] Dolev and Strong provide an algorithm for Byzantine Agreement with reliability t on a t+1 connected network with e edges that requires O(e) messages and t+d phases, where d is the maximum (over all pairs of processors) of the minimum (over all sets of t+1 node disjoint paths between the pair) of the maximum length in links of the set.[8]

To allow links as well as processors to fail, and to provide the hope of eventually detecting faulty processors and correcting any damage they cause locally by failing to agree with others, some change must be made in the definition of a correct processor (to account for the otherwise correct processor that is isolated by incorrect links). Such a change was suggested by Dolev and Strong when they proposed their application of Byzantine Agreement to distributed transaction commit.[9] There they suggest a conservative algorithm by means of which a processor can decide when it has become too isolated to participate in Byzantine Agreements. They also suggest a method of counting component failures to reach an equivalent number of processor failures in order to apply the theory developed for the model with perfect communication. Most algorithms designed for the perfect communication model will have the same reliability measured in component failures tolerated on a sufficiently connected network with potentially unreliable links.

(2) Exact synchronization

Assuming a bound on the rate of drift of clocks, Dolev and Strong provide an asynchronous algorithm for Byzantine Agreement based on work of Lamport.[9,15] One can imagine using periodic Byzantine Agreements to synchronize clocks to within the tolerance required. Alternatively, Lamport and Melliar-Smith provide algorithms directly suited to such synchronization.[16] The problem seems not to require as strong an agreement as the Byzantine Agreement and further work in the area is likely. A significant byproduct of Dolev and Strong's contruction is that phases are no longer discrete but rather nested and message activity need not wait ~~to~~ march in lock step with worst case times.

(3) Authentication

We have discussed alternatives to authentication in the previous sections. One interesting open question concerns whether authentication actually helps in any way other than allowing the reliability to exceed one-third of the number of processors.

(4) Known initial conditions

At the cost of carrying additional information along with the value, initial conditions need not be known: The initial message can carry an agreement identifier, a time of generation, and even a list of participants, as well as the value and signature. If these are authenticated, then no other information is needed as long as the default value has a meaning in the application that requires no action. If the default requires an action (e.g. abort transaction) then the starting time for the agreement must be bounded independent of receipt of any message of the agreement algorithm.

## SUMMARY

We summarize here the key results on algorithms for Byzantine Agreement. These results were obtained within the phase model but apply, when suitably interpreted, to the more realistic model with weakened restrictions discussed in the previous section. Recall that in the model, n is the number of processors, t is the reliability of the algorithm, and f is the number of actual faults. We use the abbreviations IBA for Immediate Byzantine Agreement and EBA for Eventual Byzantine Agreement.

Without authentication, BA can only be reached when $n>3t$.[19]

IBA requires at least t+1 phases.[6]

EBA requires at least min(f+2,t+1) phases.[10]

BA requires the exchange of $\Theta(nt\log(n))$ bits of information.[5]

Using authentication, IBA can be achieved in t+1 phases with the exchange of $O(nt\log(n))$ bits.[8]

For $n>>t$ IBA can be achieved in t+1 phases with the exchange of $O(nt^3\log(n))$ bits.[6]

For $n>3t$ IBA can be achieved in 2t+3 phases with the exchange of $O(nt^3\log(n))$ bits.[4]

For $n>>t$ EBA can be achieved in min(f+2,t+1) phases with the exchange of $O(nt^3\log(n))$ bits.[6]

**For n>3t EBA can be achieved in min(2f+5,2t+3) phases with the exchange of $O(nt^3\log(n))$ bits.[6]**

We call an agreement algorithm **feasible** if it requires an amount of information exchange measured in bits that is a small polynomial in both n and t. Tradeoffs have been established between the number of phases and the amount of information exchange required[5] but there is still a large gap between known upper and lower bounds on the number of phases required by feasible algorithms for BA with high reliability. Moreover, it has not been established whether authentication helps in any way other than to allow reliability to exceed one-third n.

What has been established is that there do exist feasible algorithms for Byzantine Agreement. The standard that they set for reliability offers a valuable model for reliable distributed systems.

## REFERENCES

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

[2] D. Dolev, "The Byzantine Generals Strike Again," Journal of Algorithms, vol. 3, no. 1, pp. 14-30, 1982.

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

[4] D. Dolev, M. Fischer, R. Fowler, N. Lynch, and R. Strong, "Efficient Byzantine Agreement Without Authentication," Information and Control, to appear. See also IBM Research Report RJ3428 (1982).

[5] D. Dolev and R. Reischuk, "Bounds on Information Exchange for Byzantine Agreement," Proceedings, ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, Ottawa, Aug. 1982. See also IBM Research Report RJ3587 (Sep. 1982).

[6] D. Dolev, R. Reischuk, and H. R. Strong, "'Eventual' Is Earlier then 'Immediate'," 23rd Annual Symposium on Foundations of Computer Science, 1982. See also IBM Research Report RJ3632 (Oct. 1982).

[7] D. Dolev and H. R. Strong, "Polynomial algorithms for multiple processor agreement," proceedings, the 14th ACM SIGACT Symposium on Theory of Computing, May 1982. See also IBM Research Report RJ3342 (1981).

[8] D. Dolev and H. R. Strong, "Authenticated Algorithms for Byzantine Agreement," Siam Journal on Computing, to appear. See also IBM Research Report RJ3416 (1982).

[9] 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. See also IBM Research Report RJ3417 (1982).

[10] 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. See also IBM Research Report RJ3418 (1982).

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

[12] M. Fischer and N. Lynch, "A Lower Bound for the Time to Assure Interactive Consistency," Information Processing Letters, 14(4), pp. 183-186, 1982.

[13] M. Fischer, N. Lynch, and M. Paterson, Impossibility of Distributed Consensus with One Faulty Process, unpublished manuscript, Aug., 1982.

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

[15] L. Lamport, "Using Time Instead of Timeout for Fault-Tolerant Distributed Systems," Technical Report, Computer Science Laboratory, SRI International, June 1981.

[16] L. Lamport, and P. M. Melliar-Smith, "Synchronizing Clocks in the Presence of Faults," Technical Report, Computer Science Laboratory, SRI International, March 1982.

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

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

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

[20] R. Reischuk,"A New Solution for the Byzantine Generals Problem", in preparation.