

GIT-ICS-82/02  
A SIMPLE AND EFFICIENT  
BYZANTINE GENERALS ALGORITHM †

Nancy A. Lynch\*  
Michael J. Fischer\*\*  
Robert Fowler\*\*\*

February, 1982

\*Computer Science Department  
Massachusetts Institute of Technology  
Cambridge, MA 02139

\*\*Department of Computer Science  
Yale University  
New Haven, CT 06520

\*\*\*Computer Science Department  
University of Washington  
Seattle, WA 98195

† This work was supported in part by the Office of Naval Research under Contract N00014-80-C-0221 through a subcontract from the University of Washington, by the Office of Army Research under Contract DAAG29-79-C-0155, and by the National Science Foundation under Grants MCS79-24370 and MCS81-16678.

# A SIMPLE AND EFFICIENT BYZANTINE GENERALS ALGORITHM

Nancy A. Lynch  
Massachusetts Institute of Technology<sup>1</sup>  
Cambridge, Massachusetts

Michael J. Fischer  
Yale University  
New Haven, Connecticut

and

Robert Fowler  
University of Washington  
Seattle, Washington

February, 1982

\*This work was supported in part by the Office of Naval Research under Contract N00014-80-C-0221 through a subcontract from the University of Washington, by the Office of Army Research under Contract DAAG29-79-C-0155, and by the National Science Foundation under Grants MCS-79-24370 and MCS81-16678.

---

<sup>1</sup>On leave from Georgia Institute of Technology

## ABSTRACT

The Byzantine Generals problem involves a system of  $N$  processes,  $t$  of which may be unreliable. The problem is for the reliable processes to agree on a binary value sent by a "general", which may itself be one of the  $N$  processes. If the general sends the same value to each process, then all reliable processes must agree on that value, but in any case, they must agree on the same value. We give an explicit solution for  $N = 3t + 1$  processes, using  $2t + 4$  rounds and  $O(t^3 \log t)$  message bits, where  $t$  bounds the number of faulty processes. This solution is easily extended to the general case of  $N \geq 3t + 1$  to give a solution using  $2t + 5$  rounds and  $O(tN + T^3 \log t)$  message bits.

### 1. Introduction

The Byzantine Generals problem (or, the problem of "assuring interactive consistency") is defined in [PSL]. It is assumed that there are  $N$  isolated processes, of which at most  $t$  are faulty. The processes can communicate by means of two-party messages, using a medium which is reliable and of negligible delay. The sender of a message is always identifiable by the receiver. The problem is for the nonfaulty processes to agree on a binary value sent by a "general", which may itself be one of the  $N$  processes. If the general sends the same value to each process, then all reliable processes must agree on that value. If the general sends different values to different processes (i.e. the general is "faulty"), then all reliable processes must agree on some value.

Algorithms for solving this problem are surprisingly difficult to devise. The difficulty is that faulty processes can provide conflicting information to different parts of the system. This fact causes simple solutions based on majority voting to fail, since a faulty process could cause two nonfaulty processes to decide that the majority voted in opposite ways.

An efficient solution to the Byzantine Generals problem would be a valuable tool for the construction of reliable computer systems. Such systems should be able to handle malfunctioning components which provide conflicting information.

The algorithms in the earliest papers on this problem [PSL, LSP] seem to be quite expensive, both in terms of number of message bits (exponential in  $t$ , the number of faults) and time ( $t + 1$  rounds of synchronous message exchange). This is true even in the presence of certain authentication capabilities. It is shown in [FL], in the simplest case of non-authenticated communication, that  $t + 1$  rounds are optimal, for worst-case algorithm behavior. This lower bound result is extended in [DS, DLM] to the case in which arbitrary authentication capabilities are allowed. Thus, there is no way to improve on the number of rounds in the earlier algorithms.

The more serious drawback of the earlier algorithms is the large amount of message traffic which is sent among the processes. There is essentially no structure to the information which is exchanged in those algorithms; processes repeatedly broadcast everything they know, and then apply certain decision functions to the final results. It is obviously desirable to discover ways of summarizing the information, only sending what is relevant.

The first solution which requires an amount of communication polynomial in the number of faults appears in [DS]. The authors summarize the information in clever ways, and obtain a solution which uses  $4t + 4$  rounds and  $O(n^4 \log n)$  message bits. (Their solution can easily be modified, using the same trick we use in Section 3., to use  $4t + 5$  rounds and  $O(tN + t^4 \log t)$  message bits.)

In the present paper, we use many of the ideas of [DS], plus several new ones, to devise another solution with polynomial communication. Our solution uses only  $2t + 5$  rounds, and  $O(tN + t^3 \log t)$  message bits, thus giving important savings both in time and amount of communication. In addition, we think that the new algorithm is considerably simpler than the algorithm of [DS].

We do not know if our algorithm is optimal; in particular, we have so far been unsuccessful at removing the factor of 2 which separates the number of rounds used by our algorithm from the known minimum.

## 2. The Model

Let  $[N]$  denote  $\{1, \dots, N\}$ .

We model a Byzantine Generals algorithm as a synchronous system of automata. Such a system  $S$  is described by the following:

$N$  -- the number of processes;

$Q = (Q_1, \dots, Q_N)$  -- the state sets of each of the  $N$  processes;

$q_0 = (q_{0_1}, \dots, q_{0_N})$  -- initial states for each process indicating the general's value is "0",

$q_1 = (q_{1_1}, \dots, q_{1_N})$  -- initial states for each process indicating the general's value is "1",

$F = (F_1, \dots, F_N)$ , where each  $F_i \subseteq Q_i$  -- accepting states for each process,

$M = (M_1, \dots, M_N)$  -- the sets of possible messages which each process might send,

$\mu_{i,j}: Q_i \rightarrow M_j, i, j \in [N]$  -- the message generation functions,

(where  $\mu_{i,j}$  describes messages sent from process  $i$  to process  $j$ )

and

$\zeta_j: Q_j \times M_1 \times \dots \times M_N \rightarrow Q_j, j \in [N]$ --the state transition functions.

Let  $T \subseteq [N]$ , and let  $v \in \{0,1,?\}$ . ( $T$  is the set of *reliable* processes or "truthtellers", and  $v$  is the general's value. A value of '?' indicates that the general himself is unreliable.) A sequence of state vectors  $q(0), q(1), \dots, q(R)$  is an  $R$ -round  $(T,v)$ -computation if there exist messages  $m_{i,j}(r) \in M_{i,j}, i,j \in [N], 0 \leq r < R$ , such that

1. INITIALIZATION:

If  $v = 0$  then  $q(0) = q_0$ .

If  $v = 1$  then  $q(0) = q_1$ .

If  $v = ?$  then  $q_i(0) \in \{q_0, q_1\}$ , for all  $i \in [N]$ .

2. CORRECT MESSAGES:

For each  $r, 0 \leq r < R$  and each  $i \in T, j \in [N], m_{i,j}(r) = \mu_{i,j}(q_i(r))$ .

3. CORRECT TRANSITIONS:

For each  $r, 0 \leq r < R$ , and each  $j \in T, q_j(r+1) = \zeta_j(q_j(r), m_{1,j}(r), \dots, m_{N,j}(r))$ .

We say that  $S$  solves the Byzantine Generals problem in  $R$  rounds if for every  $T \subseteq [N]$  with  $|T| \geq N-t$ , every  $v \in \{0,1,?\}$ , and every  $R$ -round  $(T,v)$ -computation  $q(0), \dots, q(R)$ , the final state vector  $q(R)$  satisfies the following:

1. AGREEMENT: If  $i,j \in T$ , then  $q_i(R) \in F_i$  iff  $q_j(R) \in F_j$ .

2. VALIDITY: If  $v \neq '?'$ , then for all  $i \in T, q_i(R) \in F_i$  iff  $v = 1$ .

Intuitively, a step or round of the computation takes place in two phases. First, every process sends a message to every other. Secondly, each process changes state based on its old state and the messages it receives. Unreliable processes can send arbitrary messages, so there are in general many possible computations, all of which must satisfy the agreement and validity conditions above.

We assume about the general only that it is a possibly-unreliable data source that communicates a (binary) value to each of the  $N$  processes in the system before the algorithm begins. Thus, the general might be one of the  $N$  processes, or it might be a sensor or I/O device that all processes can read. In our formalization, the general's value is encoded by each process's start state. In other treatments of this problem, the general is identified with one of the  $N$  processes which carry out the algorithm, and each other process starts in the same state regardless of the general's value. Our version is slightly stronger, for a solution to our problem solves the other version by simply adding an

initial round in which the general sends his value to each other process. The converse, however, is not in general true, for an algorithm might make use of the fact that at most  $t-1$  unreliable processes remain when the general has been determined to be unreliable and is a known one of the processes.

### 3. A Simplification

We give an explicit construction for the case  $N = 3t + 1$ . To handle the case of  $N > 3t + 1$ , just run the given algorithm on any subset  $A$  with  $|A| = 3t + 1$ . After the last round, a designated subset  $B \subseteq A$ ,  $|B| = 2t + 1$ , broadcasts its answers to all  $N$  processes. Since all the  $(t + 1$  or more) reliable processes in  $B$  agree, a simple majority vote gives all the other reliable processes consistent answers. This takes only one additional round and  $O(tN)$  additional message bits above and beyond the basic algorithm.

### 4. Basic Solution

Now assume  $N = 3t + 1$ . Let  $LOW = t + 1$  and  $HIGH = 2t + 1$ . We describe a system  $S$ .

The only pieces of information sent in messages are process indices and one special value '\*'. Formally, let  $I$  (the set of message items) =  $\{ '*' \} \cup [N]$ . Messages are sets of message items; thus, each  $M_i = 2^I$ .

A process state consists of a number (representing the current round) together with a set of "data entities". A data entity is either the single value 0 or 1 (representing a value of 0 or 1 received from the general) or else a pair consisting of a message item and a process from which that message is received. Each process remembers the initial value and all the messages it has ever received from any process. Formally, a data entity is an element of  $D = \{0,1\} \cup (I \times [N])$ . A process state  $q$  is a pair  $(data(q), round(q))$ , where  $data(q) \subseteq D$  and  $round(q) \in \mathbb{N}$ . That is, each  $Q_i = 2^D \times \mathbb{N}$ . The initial states are  $q_{0_i} = (\{0\}, 0)$  and  $q_{1_i} = (\{1\}, 0)$ . The transition function simply records all new messages received, together with their senders, and increments the round number. That is,

$$\zeta_i(q, m_1, \dots, m_N) = (data(q) \cup \{(x,j) \in D \mid x \in m_j\}, round(q) + 1).$$

Thus, the data component of the process state behaves "monotonically"-new data entities can get added during the course of an execution, but nothing is ever deleted.

We require some notation for characterizing process states. Let  $q$  be any process state and let  $x \in I$ . We define

$$W_x(q) = \{j \in [N] \mid (x,j) \in data(q)\},$$

the witnesses to  $x$ , and we let  $w_x(q) = |W_x(q)|$ . We define

$$C(q) = \{k \in [N] \mid w_k(q) \geq HIGH\},$$

the confirmed processes, and we let  $c(q) = |C(q)|$ . Process  $i$  initiates in  $q$  if either

- I1.  $1 \in \text{data}(q)$ ,
- I2.  $c(q) \geq \text{LOW} + \lceil \text{round}(q)/2 \rceil - 1$ , or
- I3.  $i \in W_*(q)$ .

Process  $i$  commits in  $q$  if  $c(q) \geq \text{HIGH}$ .

The heart of the algorithm is the message generation function. The function is defined to be monotonic in the data component of the state - more data entities can only cause more messages to be sent. Since the data component of the state behaves monotonically, this definition implies that any message, once sent, will be sent on all subsequent rounds. This is an obvious inefficiency which is removed by a trivial optimization. (See Section 6.) It is useful to describe the algorithm in this way, however, since the monotonic algorithm is easier to reason about than its optimized version.

We define  $\mu_{i,j}(q)$  to be the smallest set satisfying the following rules:

- M1. (Initiation) If  $i$  initiates in  $q$ , then  $'*' \in \mu_{i,j}(q)$ .
- M2. (Direct witness)  $W_*(q) \subseteq \mu_{i,j}(q)$ ;
- M3. (Indirect witness) If  $w_k(q) \geq \text{LOW}$ , then  $k \in \mu_{i,j}(q)$  for each  $k \in [N]$ .

Finally,  $F_i = \{q \in Q \mid i \text{ commits in } q\}$ .

**Theorem 1:** Let  $R = 2t + 4$ . Then  $S$  solves the Byzantine Generals problem in  $R$  rounds.

The correctness of this algorithm is somewhat subtle and is proved in the next section. However, the following intuition should help the reader's understanding.

During the course of execution, processes initiate from time to time. This means that they know that the general has sent a "1" to some reliable process and that they are proposing to accept. A process announces initiation by sending a '\*' to the other processes.

A process receiving a '\*' becomes a witness to the sending process's initiation. A process can become an "indirect" witness by hearing about it from at least  $\text{LOW}$  other processes, since then at least one of them must be reliable. In either case, it broadcasts that fact to all processes, including itself. (The sending process will thus record itself as a witness at the same time as all other processes do.)

A process receiving a message item  $k \in [N]$  from process  $j$  records the fact that  $j$  claims to be a witness  $k$ . When at least  $HIGH$  distinct  $j$ 's claim to be witnesses to  $k$ , then  $k$  is confirmed. The confirming process then knows one of two things must be true: Either  $k$  is reliable and indeed has initiated, or  $k$  is unreliable but nevertheless has told at least  $LOW$  reliable processes that it had initiated.

A process initiates on the first round if it receives a "1" from the general. Thereafter, it can only initiate if it has confirmed sufficiently many initiations by other processes. This threshold number for initiation starts out at  $LOW$  and increases by one every two rounds until it reaches  $HIGH$ . By that time, either at least  $LOW$  reliable processes will have initiated or it is no longer possible for a reliable process to initiate. In the former case, after three more rounds every reliable process will commit. In the latter case, no reliable process can commit. The delicate part of the algorithm concerns these last two facts; namely, initiating and committing are easy enough so that as soon as  $LOW$  reliable processes initiate, then an avalanche begins which results in all reliable processes initiating and committing a small number of rounds later. On the other hand, committing is hard enough so that no process commits in the last three rounds except as a result of an avalanche started earlier.

## 5. Proof of Correctness

The following lemmas prove Theorem 1 and establish the correctness of the algorithm. All refer to a fixed  $(T,v)$ -computation  $q(0), \dots, q(R)$ ,  $R = 2t + 4$ , with associated messages  $m_{i,j}(r)$ ,  $i, j \in [N]$ ,  $0 \leq r < R$ .

Lemma 2 formalizes the monotonicity properties of process states.

**Lemma 2:** Let  $0 \leq r' \leq r \leq R$ ,  $i \in T$ . Then  $W_x(q_i(r')) \subseteq W_x(q_i(r))$  for all  $x \in I$ , and  $C(q_i(r')) \subseteq C(q_i(r))$ . Moreover, if  $i$  initiates (commits) in  $q_i(r')$ , then  $i$  initiates (commits) in  $q_i(r)$ .

**Proof:** If  $r' = r$ , then there is nothing to prove. So assume  $r' < r$ . Monotonicity of  $W$  and  $C$  are obvious; hence, if  $i$  commits in  $q_i(r')$ , then it commits in  $q_i(r)$ . Suppose  $i$  initiates in  $q_i(r')$ . Then  $'*' \in m_{i,i}(r')$ , so  $i \in W_*(q_i(r' + 1))$ , and by monotonicity of  $W$ ,  $i \in W_*(q_i(r))$ . Thus,  $i$  initiates in  $q_i(r)$  by Rule I3. □

The next lemma says that whenever a truth teller initiates, it is confirmed at all truth tellers two rounds later.

**Lemma 3:** Let  $i, j \in T$ . If  $i$  initiates in  $q_i(r)$ ,  $0 \leq r \leq R-2$ , then  $i \in C(q_j(r+2))$ .

**Proof:** Let  $k \in T$ . Then  $i \in W_*(q_k(r+1))$  by Rule M1. Similarly,  $k \in W_i(q_j(r+2))$  by Rule



M2. Hence,  $W_i(q_j(r+2)) \supseteq T$ . The lemma follows since  $|T| \geq \text{HIGH}$ .

□

Next, we show that whenever all truthtellers initiate, they all commit two rounds later.

**Lemma 4:** Let  $0 \leq r \leq R-2$ . If all  $i \in T$  initiate in  $q_i(r)$ , then all  $i \in T$  commit in  $q_i(r+2)$ .

**Proof:** By Lemma 3,  $i \in C(q_j(r+2))$  for all  $j \in T$ ; hence,  $c(q_i(r+2)) \geq \text{HIGH}$ .

□

The next lemma describes some information that the views of different truthtellers at the same round must have in common.

**Lemma 5:** Let  $i, j, k \in T, x \in I$ . Then  $k \in W_x(q_i(r))$  iff  $k \in W_x(q_j(r))$ .

**Proof:** Follows from an easy induction on  $r$  using the fact that reliable processes always broadcast their messages to every process.

□

Next, we show the important fact that any process which gets confirmed at one truthteller, will be confirmed at all truthtellers one round later.

**Lemma 6:** Let  $0 \leq r \leq R-1, j, k \in T$ . If  $i \in C(q_k(r))$  then  $i \in C(q_j(r+1))$ .

**Proof:** Since  $i \in C(q_k(r))$ , there is a set  $A \subseteq T \cap W_i(q_k(r))$  with  $|A| = \text{LOW}$ . Let  $j' \in T$ . Then by Lemma 5,  $A \subseteq W_i(q_{j'}(r))$ . Thus,  $i \in m_{j',j}(r)$ , by Rule M3. Hence,  $j' \in W_i(q_j(r+1))$ . Thus,  $i \in C(q_j(r+1))$ .

□

**Lemma 7:** Let  $0 \leq r < R, i, j \in T$ . If  $i$  commits in  $q_i(r)$ , then  $j$  commits in  $q_j(r+1)$ .

**Proof:** by Lemma 6.

□

The next lemma says that if there are sufficiently many witnesses for a truthteller, then that truthteller has actually initiated.

**Lemma 8:** Let  $i, j \in T$ . If  $w_i(q_j(r)) \geq \text{LOW}$ , then  $r \geq 2$  and  $i$  initiates in  $q_i(r-2)$ .

**Proof:** We proceed by induction on  $r$ . Suppose the lemma is true for all  $r' < r$ , for  $r \geq 0$ , and suppose  $w_i(q_j(r)) \geq \text{LOW}$ . Then there is some  $k \in T \cap W_i(q_j(r))$ . But then  $r \geq 1$  and  $i \in m_{k,j}(r-1)$ , and this is either because of M2 or M3. If it is because of M2, then  $i \in W_*(q_k(r-1))$ , so that  $r \geq 2$  and  $i \in m_{i,k}(r-2)$  and hence  $i$  initiates in  $q_i(r-2)$ . If it is because of M3, then  $w_i(q_k(r-1)) \geq \text{LOW}$ . Then by induction,  $r-1 \geq 2$  and  $i$  initiates in  $q_i(r-3)$ . Application of Lemma 2 shows that  $i$  initiates in  $q_i(r-2)$ .

□

The following lemma follows easily from Lemma 8.

**Lemma 9:** Let  $i \in T$ , and suppose  $i$  commits in  $q_i(r)$ . Then  $r \geq 2$  and there is a set  $A \subseteq T$  with  $|A| = \text{LOW}$  such that every  $j \in A$  initiates in  $q_j(r-2)$ .

**Proof:**  $c(q_i(r)) \geq \text{HIGH}$ , so there is a set  $A \subseteq T \cap C(q_i(r))$  with  $|A| = \text{LOW}$ . Each  $j \in A$  has  $w_j(q_i(r)) \geq \text{HIGH}$ ; hence, by Lemma 8,  $r \geq 2$  and  $j$  initiates in  $q_j(r-2)$ . □

The following key lemma says that whenever LOW truth-tellers initiate, then all truth-tellers commit four rounds later. This is the "avalanche" described in the intuitive discussion of the algorithm.

**Lemma 10:** Let  $0 \leq r \leq R-4$ . If there is a set  $A \subseteq T$ ,  $|A| = \text{LOW}$ , such that all  $i \in A$  initiate in  $q_i(r)$ , then all  $j \in T$  commit in  $q_j(r+4)$ .

**Proof:** Let  $r'$  be the least number such that all  $i \in A$  initiate in  $q_i(r')$ . By Lemma 3,  $A \subseteq C(q_j(r'+2))$  for all  $j \in T$ . We now argue that  $j$  initiates in  $q_j(r'+2)$ . It will then follow by Lemma 4 that  $j$  commits in  $q_j(r'+4)$ , and hence also in  $q_j(r+4)$  by Lemma 2.

If  $r' = 0$ , then  $c(q_j(r'+2)) \geq |A| = \text{LOW} + \Gamma(r'+2)/2\Gamma - 1$ . Thus,  $j$  initiates in  $q_j(r'+2)$  by Rule I2. If  $r' > 0$ , then there is some  $k \in A$  such that  $k$  initiates in  $q_k(r')$  and  $k$  does not initiate in  $q_k(r'-1)$ . Then  $k$  initiates in  $q_k(r')$  using Rule I2, so  $c(q_k(r')) \geq \text{LOW} + \Gamma r'/2\Gamma - 1$ . If  $k \in C(q_k(r'))$ , then Lemma 8 implies that  $k$  initiates in  $q_k(r'-2)$ , a contradiction (using Lemma 2). Thus,  $k \notin C(q_k(r'))$ . By Lemmas 2 and 6,  $C(q_j(r'+2)) \supseteq C(q_k(r'))$  for all  $j \in T$ . By Lemma 3,  $k \in C(q_j(r'+2))$ . Hence,  $c(q_j(r'+2)) \geq \text{LOW} + \Gamma r'/2\Gamma = \text{LOW} + \Gamma(r'+2)/2\Gamma - 1$ . Thus,  $j$  initiates in  $q_j(r'+2)$  by Rule I2 as desired. □

We are now ready to prove the properties required for Theorem 1 - agreement and validity.

**Lemma 11:** If any  $i \in T$  commits in  $q_i(R)$ , then all  $j \in T$  commit in  $q_j(R)$ .

**Proof:** Assume  $i \in T$  commits in  $q_i(R)$ . By Lemma 9, there is a set  $A \subseteq T$  with  $|A| = \text{LOW}$  such that every  $j \in A$  initiates in  $q_j(R-2)$ .

We consider two cases. First, assume all  $j \in A$  initiate in  $q_j(R-4)$ . In this case, Lemma 10 implies the result. Second, assume that some  $j \in A$  initiates in  $q_j(r)$  but not in  $q_j(r-1)$ , for some  $r \in \{R-3, R-2\}$ . Then  $j$  initiates by I2. Then  $c(q_j(r)) \geq \text{LOW} + \Gamma r/2\Gamma - 1 \geq \text{LOW} + t = \text{HIGH}$ , so  $j$  commits in  $q_j(r)$ . Then Lemmas 7 and 2 imply the result. □

**Lemma 12:** Let  $i \in T$ .

(a) If  $v = 0$ , then  $q_i(R) \notin F_i$ .

(b) If  $v = 1$ , then  $q_i(R) \in F_i$ .

**Proof:** (a)  $v = 0$ . Suppose  $i$  commits in  $q_i(R)$ . Then by Lemma 9, there is an element  $j \in T$  that initiates in  $q_j(R-2)$ . Consider the least  $r$  for which some  $j \in T$  initiates in  $q_j(r)$ . Clearly  $r > 0$  by the initial conditions. Hence,  $j$  initiates by Rule I2, so  $c(q_j(r)) \geq \text{LOW}$ . Thus, there is a  $k \in T \cap C(q_j(r))$ , so  $w_k(q_j(r)) \geq \text{HIGH}$ . But then it follows from Lemma NANCY7 that  $k$  initiates in  $q_k(r-2)$ , contradicting the choice of  $r$ . We conclude that  $q_i(R) \notin F_i$ .

(b)  $v = 1$ . Each  $i \in T$  initiates in  $q_i(0)$  by Rule I1. By Lemma 4, each  $i \in T$  commits in  $q_i(2)$ . Thus,  $q_i(R) \in F_i$ .

□

## 6. Complexity Analysis

Since  $||i|| = N+1$ , each message item can be encoded by  $O(\log N)$  bits, and a message  $m$  consisting of  $k$  message items can be encoded in length  $O(k \log N)$ . The algorithm of the previous section sends  $N^2$  messages on each round, and each message potentially contains  $N+1$  message items; hence an upper bound on the number of message bits sent is  $O(N^2 R (N+1) \log N) = O(t^4 \log t)$ . (The log factor can be eliminated by a bitwise encoding of the entire message.)

A minor modification of the algorithm however results in a saving of the factor of  $R$ . The algorithm is monotone in the sense that data entities are never deleted from the data part of the state, and incoming messages have no effect except to be added into the state. Thus, the algorithm would operate exactly the same if each message item were sent from  $i$  to  $j$  only once. The only change to the algorithm would be that each process would have to remember in its state which messages had previously been sent out and to whom, and to omit sending a previously-sent message. The result is that each process  $i$  would send a maximum of  $||i||$  message items to each process  $j$  during the entire course of the algorithm. The total number of message bits then would be  $O(N^2 (N+1) \log N) = O(t^3 \log t)$ .

Combining the ideas of the previous paragraph with those of Section 3, we obtain:

**Theorem 13:** There is an algorithm which solves the Byzantine Generals problem for  $t$  unreliable processes out of a total of  $N \geq 3t+1$ , uses  $2t+5$  rounds of information exchange, and sends  $O(t^3 \log t + tN)$  message bits.

**Acknowledgements:**

The authors would like to thank Paris Kanellakis for help in understanding the key ideas in [DS], and Eugene Stark for help in improving the presentation.

**References:**

[DLM] DeMillo, R., Lynch, N. and Merritt, M. Cryptographic Protocols. SIGACT 1982

[DS] Dolev, D. and Strong, S. An Efficient Byzantine Agreement Without Authentication.  
SIGACT 1982

[FL] Fischer, M. and Lynch, N. A Lower Bound on the Time to Reach Interactive Consistency.  
To appear in IPL.

[LSP] Lamport, L., Shostak, R. and Pease, M. The Byzantine Generals Problem, Manuscript.

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