INFORMATION AND COMPUTATION 97, 61–85 (1992)

Modular Construction of a Byzantine Agreement Protocol with Optimal Message Bit Complexity*

BRIAN A. COAN

Bellcore, Morristown, New Jersey 07960

AND

JENNIFER L. WELCH

University of North Carolina, Chapel Hill, North Carolina 27514

This paper presents a new Byzantine agreement protocol that tolerates t processor faults using $3t+1$ processors, $t+o(t)$ rounds, $O(t^2)$ total message bits, and $O(t^\varepsilon)$ maximum message size, for any $\varepsilon > 0$. The protocol is optimal or near optimal in all cost measures: the number of processors is optimal, the message bit complexity is optimal, the number of rounds exceeds the lower bound by $o(t)$, and the maximum message size exceeds the lower bound by $O(t^\varepsilon)$. The round complexity is uniformly better than $2 \cdot (t+1)$ and thus is reasonable even for small t. This is the first Byzantine agreement protocol to have optimal message bit complexity. The new protocol is constructed by recursively applying a simple, yet general, transformation that changes the number of rounds, total message bits, and maximum message size required by a Byzantine agreement protocol, but preserves correctness, number of processor faults tolerated, and total number of processors. Each application of this new transformation reduces the number of message bits sent—at the expense of adding rounds of communication. Surprisingly, the base case of the recursive construction is the agreement protocol of Lamport, Shostak, and Pease, which has a number of message bits exponential in t.

1. INTRODUCTION

In the field of distributed computing, Byzantine agreement is a fundamental problem which was first introduced by Pease, Shostak, and Lamport (1980). Many variants of this problem have been studied. In this

* This paper combines results that were presented in preliminary form at the 8th ACM Symposium on Principles of Distributed Computing (Coan and Welch, 1989a) and the 27th Allerton Conference on Communication, Control, and Computing (Coan and Welch, 1989b).

0890-5401/92 $3.00

paper we consider a system of deterministic processors that communicate by message passing in a sequence of synchronous rounds. Some processors are correct, following their protocol exactly. The rest experience Byzantine faults, deviating from their protocols in arbitrary ways. Each processor begins with an input bit and eventually each nonfaulty processor must irrevocably decide on an output bit satisfying two conditions: (1) all the output bits are the same, and (2) if all the input bits are the same, then all the output bits must equal the input bits. For this version of the problem, there is currently a gap between the known lower bounds on resources and the most efficient known protocols. The results in this paper help narrow that gap.

There are five cost measures that interest us: number of processors, rounds of communication, total number of message bits, size of the largest message, and amount of local computation. If t is an upper bound on the number of processor faults, then the following lower bounds are known: $3t+1$ processors (Lamport, Shostak, and Pease, 1982; Fischer, Lynch, and Merritt, 1986), $t+1$ rounds (Fischer and Lynch, 1982), $\Omega(t^2)$ message bits (Dolev and Reischuk, 1985), and maximum message size of 1. The lower bounds on processors, message size, and rounds were known to be achievable. For example, the original protocol of Lamport, Shostak, and Pease (1982) achieves the bounds on processors and rounds (but requires a number of message bits that is exponential in t). Protocols of Bar-Noy and Dolev (1991) and Coan (1988) achieve the lower bound on message size and rounds (but exceed the lower bounds on processors and total message bits). The best previous upper bound on message bits is achieved by the protocol of Dolev *et al.* (1982), requiring $O(t^3 \log t)$ total message bits. Our new protocol achieves a message bit complexity of $O(t^2)$, which is optimal. Most protocols, including ours, achieve polynomial local computation; we will not discuss this measure further.

Our technique consists of the recursive application of a general two-step transformation that reduces the number of message bits and the size of the largest message sent at the expense of a small increase in the number of rounds. The transformation leaves unchanged the number of processors and the number of faults tolerated. The first step transforms any Byzantine agreement protocol that has a known upper bound on rounds—essentially all existing protocols have such a bound—into a protocol for the Byzantine broadcast problem. The Byzantine broadcast problem is a new problem first defined in this paper. In a Byzantine broadcast protocol, a subset of the processors, called the "committee," attempts to reach agreement and to communicate the decision to the rest of the processors. The resulting Byzantine broadcast protocol operates correctly as long as there are enough nonfaulty processors in its committee. The second step of our transformation produces a Byzantine agreement protocol from a collection

of Byzantine broadcast protocols that have disjoint committees. For the combined protocol to work correctly, it is sufficient that the fault-tolerance assumption of any one of its constituent Byzantine broadcast protocols be satisfied—it does not impair the correct operation of the combined protocol if the committees of the remaining Byzantine broadcast protocols have "too many" faulty processors. Thus, we can combine B Byzantine broadcast protocols, tolerating $t_1, ..., t_B$ faults, respectively, to produce a Byzantine agreement protocol tolerant of $t_1 + \cdots + t_B + B - 1 = t$ faults overall. The resulting protocol satisfies the condition referred to above, and thus our transformation can be applied recursively.

One application of our transformation takes B, the number of committees, as a parameter. In applying the transformation recursively, we introduce two additional parameters, l and d. The parameter l is a lower bound on the number of faults to be tolerated by protocol instances at the base of the recursion. (There is a trade-off between the message size complexity and the round complexity in the choice of l: a smaller l means a smaller message size but more rounds.) Finally, we let the depth of the recursion be a function f of t, the number of faults to be tolerated, and d, our third parameter. (In sufficiently large systems, the parameter d determines a small constant increment in the depth of the recursion, thereby reducing the message size; it is chosen to be $\max\{4, \lceil 1/\varepsilon \rceil\}$, where $O(t^\varepsilon)$ is the desired upper bound on message size.) By using the protocol of Lamport, Shostak, and Pease (1982) at the base of the recursion and by defining f appropriately, we obtain a Byzantine agreement protocol that uses $3t+1$ processors, $t + o(t)$ rounds, $O(t^2)$ message bits, and $O(t^\varepsilon)$ message size, for any $\varepsilon > 0$. Judicious choices for B and l, discussed in Section 6, cause the round complexity to be uniformly better than $2 \cdot (t+1)$, even for small values of t.

A different attempt at narrowing the gap in resource requirements for Byzantine agreement is the protocol of Moses and Waarts (1992), which uses $6t+1$ processors, $O(t^8)$ message bits, and exactly $t+1$ rounds. Their protocol is an ingenious synthesis of several complicated techniques; our approach is simpler. The protocol of Berman, Garay, and Perry (1989a) has further reduced the number of processors required to $4t+1$ while leaving the other cost measures essentially unchanged. Compared to these two protocols, our protocol uses fewer processors and message bits, at the expense of using $o(t)$ more rounds and of scarificing early stopping (Dolev, Reischuk, and Strong, 1990).

Our technique resembles the "shifting gears" work of Bar-Noy *et al.* (1992) in that it allows us to combine several Byzantine agreement protocols into one that is more efficient than any of the originals. However, our technique treats the Byzantine agreement protocols completely abstractly and works for any collection of Byzantine agreement protocols

(that terminate in a known bounded number of rounds). The technique of Bar-Noy *et al.* (1992) only works for protocols of a specific form, a form that excludes some known protocols. There is also a similarity between the structure of our new protocol and the structure of the randomized agreement protocols of Ben-Or (1983), Rabin (1983), and Chor and Coan (1985): the skeleton is the same, while the distributed coin flip in the randomized protocols is replaced by Byzantine broadcast in our (deterministic) protocol.

In a preliminary version of this paper (Coan and Welch, 1989a), we gave a modular construction of a class of protocols with $3t+1$ processors, $t+o(t)$ rounds, $O(t)$ message size, and $O(t^{2+\varepsilon})$ total message bits, for all $\varepsilon > 0$. Simultaneously and independently Berman and Garay (1989) used a single application of a transformation similar to ours to obtain a class of protocols with $4t+1$ processors, $(1+\varepsilon)(t+1)$ rounds, $t^{O(1/\varepsilon)}$ message size, and $t^{O(1/\varepsilon)}$ total message bits, for all ε, $0 < \varepsilon \leqslant 1$. Subsequently, it was observed independently by Berman, Garay, and Perry (1989b) and by Coan and Welch (1989b) that a slight variant of the recursive construction in Coan and Welch (1989a) yields a single protocol with optimal message bit complexity—the protocol presented in this paper.

By a result of Lamport, Shostak, and Pease (1982), no Byzantine agreement protocol for t faults can exist unless there are at least $3t+1$ processors. In this paper we only consider the case where there are exactly $3t+1$ processors. Since processors are an expensive resource, it is reasonable to minimize the total number of processors to achieve a certain level of fault tolerance, as represented by t. The easiest way to generalize our results for more than $3t+1$ processors is to utilize the simple and elegant transformation given by Dolev *et al.* (1982). Details are left to the reader.

In Section 2 we review the model. In Section 3 we define the Byzantine agreement and Byzantine broadcast problems. In Section 4 we describe the two components of our general transformation and we combine these to give the general transformation. In Section 5 we construct our new protocol by recursively applying our general transformation. In Section 6 we bound the constant in the round complexity of our protocol. In Section 7 we discuss other applications of our transformation.

2. Model

We consider a computer system consisting of a collection P of processors that interact by sending messages over links. Each link provides a bidirectional communication path between a pair of processors. The collection of links constitutes the communication network. It is known that no agree-

ment protocol is possible unless the connectivity of the communication network is at least $2t + 1$ (Dolev, 1982; Fischer, Lynch, and Merritt, 1986). Throughout this paper we assume that the communication network is fully connected and reliable.

We assume a *synchronous* timing model, meaning that all processors take steps in unison and all messages are received in the next step after they are sent. An execution of a protocol consists of a series of *rounds*. In each round each processor sends messages, whose contents depend on its internal state, receives all the messages just sent to it, and acts on the messages received by altering its internal state. The internal state of a processor is simply the values of its variables and program counter. A *protocol for* P, where P is a set of processors, prescribes what the processors' internal states are, what the initial internal states are, what messages should be sent, and how internal states should be altered. Each processor in P is assumed to have a unique name known to all the processors, and each processor can reliably determine the name of the sender of each message that it receives.

Each processor's state set contains a subset of *initial* states, partitioned into v-initial states, and a subset of *final* states, partitioned into v-final states, with the v-initial state of each processor being unique, $v \in \{0, 1\}$. In an execution, if processor p starts in a v-initial state, then v is p's *input*; if p enters a v-final state, then v is p's *output*. At the beginning of an execution, all nonfaulty processors must be in initial states. A protocol can only map v-final states to v-final states, to model the irrevocability of deciding on an output.

In an execution of a protocol, some processors are nonfaulty, and the rest are faulty. A *nonfaulty* processor sends messages and changes state according to the protocol. A *faulty* processor can deviate from the protocol. We are interested in protocols that tolerate *Byzantine* processor faults. In the Byzantine fault model there are no constraints on the contents of messages sent by a faulty processor; however, the recipient of any message knows the identity of the sender. Although a faulty processor may send different messages to each of the other processors, it may not send multiple messages to any single recipient in a round.

Our protocols will call other protocols as subroutines, using the following syntax and semantics. Let $\mathscr{P}$ and $\mathscr{P}'$ be protocols for P and P', respectively, where P' is a subset of P. Processor p in P', during an execution e of protocol $\mathscr{P}$, may execute in some (constant, i.e., fixed before the start of execution and therefore not computed by p) round, say r,

$$\text{call } \mathscr{P}'(\text{rounds: } R, \text{ input: } X, \text{ output: } Y).$$

We require that every processor in P' call $\mathscr{P}'$ in round r with the same (constant, i.e., fixed before the start of execution and therefore not

computed by that processor) value of R. (This is a requirement on the code, not a requirement on the behavior of faulty processors.) The effect of the call is that p gets a new set of variables, as specified by the protocol for p in $\mathscr{P}'$, initialized to a v-initial state, where v is the value of variable X at the beginning of round r; p computes according to $\mathscr{P}'$ for the R rounds indexed from r to $r+R-1$; and p returns value w in variable Y, where p ends in a w-final state after the R rounds ($Y=$ UNINITIALIZED if p is not in a final state). The value of Y is available to the compute phase of round $r+R-1$.

The *processor complexity* of a protocol for P is $|P|$. The *round complexity* of a protocol is the maximum, over all executions e, of the number of rounds of e required for all nonfaulty processors to be in final states. The *message bit complexity* of a protocol is the maximum, over all executions e, of the total number of bits sent in messages by nonfaulty processors in e. The *message size complexity* of a protocol is the maximum, over all executions e, of the number of bits sent in any single message by any nonfaulty processor in e.

3. Problem Statements

Our Byzantine agreement protocol uses as subroutines solutions to a closely related problem, which we call the Byzantine broadcast problem. We now define these two problems.

3.1. *Byzantine Agreement*

A protocol for P is a *Byzantine agreement protocol for P and t*, $t<|P|$, if there exists an integer $R\geqslant 0$ such that every execution at least R rounds long satisfies the following three conditions.

- *Termination.* At the end of R rounds every nonfaulty processor in P is in a final state.
- *Agreement.* If at most t processors in P are faulty, then there is a v such that any nonfaulty processor in P that enters a final state enters a v-final state.
- *Validity.* If at most t processors in P are faulty and there is a v such that all nonfaulty processors in P begin in a v-initial state, then any nonfaulty processor in P that enters a final state enters a v-final state.

This definition imposes the nonstandard requirement that a nonfaulty processor must eventually enter a final state, even if more than t processors are faulty. The more standard termination requirement, used elsewhere in

the literature, is that a nonfaulty processor must eventually enter a final state in any execution in which t or fewer processors are faulty. Most, but not all, protocols that satisfy the standard termination requirement can trivially be transformed into protocols that satisfy our termination requirement. In particular, this transformation is possible for any protocol for which there is a known upper bound, possibly a function of $|P|$ and t, on the number of rounds needed for termination.

We have defined a Byzantine agreement protocol for a fixed P and t. There is a natural sense in which we would like to consider two Byzantine agreement protocols, for different sets of processors and different t's, to be instances of the same "protocol." We thus assume an infinite set of processor names U and define a Byzantine agreement *protocol family* to be an infinite set $\{\mathrm{BA}(n, t)\colon n \geq 1 \text{ and } t \geq 0\}$ satisfying the following condition. For all n and t, either $\mathrm{BA}(n, t)$ is the empty set, or $\mathrm{BA}(n, t)$ is an infinite set of Byzantine agreement protocols, one protocol $\mathrm{BA}(P, t)$ for each n-element subset P of U. Each $\mathrm{BA}(P, t)$ must be a Byzantine agreement protocol for P and t. We require that for all n-element sets P and P', $\mathrm{BA}(P, t)$ and $\mathrm{BA}(P', t)$ are copies of each other (i.e., they are the same protocol except for renaming of processors).

We now define complexity measures for Byzantine agreement protocol families. We do not need analogous definitions for Byzantine broadcast.

The *processor complexity*, $P(t)$ of protocol family BA is the minimum n such that $\mathrm{BA}(n, t) \neq \varnothing$, i.e., the minimum number of processors needed to tolerate t faults. The rest of the complexity measures are defined assuming the minimum number of processors. The *round complexity*, $R(t)$, of protocol family BA is the function describing the round complexity of protocols in the set $\mathrm{BA}(n, t)$, where $n = P(t)$. The *message bit complexity*, $M(t)$, of protocol family BA is the function describing the message bit complexity of protocols in the set $\mathrm{BA}(n, t)$, where $n = P(t)$. The *message size complexity*, $S(t)$, of protocol family BA is the function describing the message size complexity of protocols in the set $\mathrm{BA}(n, t)$, where $n = P(t)$.

3.2. *Byzantine Broadcast*

The Byzantine broadcast problem is essentially the problem of having a known, fixed subset of the processors, called the committee, perform Byzantine agreement and communicate the result to the rest of the processors. Thus the committee causes each of the nonfaulty processors to choose an element of $\{0, 1\}$ in such a way that if enough members of the committee are nonfaulty then (1) all the nonfaulty processors choose the same element, and (2) if there is a v in $\{0, 1\}$ such that all the members of the committee begin with the input v, then all the nonfaulty processors choose v. The identity of each member of the committee is known to all processors.

A protocol for P is a *Byzantine broadcast protocol for P, C, and t'*, $C \subseteq P$, $t' < |C|$, if there exists an integer $R \geq 0$ such that every execution at least R rounds long satisfies the following three conditions. (C is the committee.)

- *Termination.* At the end of R rounds every nonfaulty processor in P is in a final state.
- *Agreement.* If at most t' processors in C are faulty, then there is a v such that any nonfaulty processor in P that enters a final state enters a v-final state.
- *Validity.* If at most t' processors in C are faulty, and if there is a v such that all nonfaulty processors in C begin in a v-initial state, then any nonfaulty processor in P that enters a final state enters a v-final state.

4. Transformations

Our general transformation is the composition of two transformations, the first from a Byzantine agreement protocol to a Byzantine broadcast protocol, and the second from a collection of Byzantine broadcast protocols to a Byzantine agreement protocol. In this section we define the component transformations and the general transformation.

4.1. *Constructing a Byzantine Broadcast Protocol from a Byzantine Agreement Protocol*

This subsection describes a simple way to transform any Byzantine agreement protocol (with a known, fixed upper bound on rounds) into a Byzantine broadcast protocol. The processors in the committee perform the Byzantine agreement protocol and then send the result to the rest of the processors, each of which decides on a value that it receives "often enough."

The transformation is presented formally as Transformation 1, where P is a set of processors, $C \subseteq P$ is the committee, t' is a nonnegative integer with $|C| \geq 2t' + 1$, and BA is a Byzantine agreement protocol for C and t' with round complexity R, message bit complexity M, and message size complexity S. The requirement that $|C| \geq 2t' + 1$ is sufficient for the correctness of our transformation; although, in our model, the existence of a Byzantine agreement protocol to be transformed requires $|C| \geq 3t' + 1$.

Transformation 1: *Changing Byzantine Agreement to Byzantine Broadcast*

Code for processor p in P:

Initialization code:

bb-in(p) is initialized to the input of processor p

bb-out(p) is initialized to UNINITIALIZED; it will hold p's output

Code for round 1:
  if p is in C
    then call BA(rounds; R, input: *bb-in*(p), output: *ba-out*(p))
    else do nothing
Code for round r, $2 \leqslant r \leqslant R$:
  comment: processors in C continue the subprotocol
    processors in $P - C$ continue to do nothing
Code for round $R+1$:
  send: **if** p is in C **then** send *ba-out*(p) to each processor in $P - C$
  receive: **if** p is in $P - C$ **then** receive messages from processors in C
  compute: *decisions*(p) := the multiset of the messages just received,
    discarding any message that is not an element of $\{0, 1\}$
    if p is in C
      then *bb-out*(p) := *ba-out*(p)
      else *bb-out*(p) := the most common element in *decisions*(p)
        (break ties by picking 0 as a default choice)

Let BB be the result of applying Transformation 1 to BA.

THEOREM 1. *BB is a Byzantine broadcast protocol for* P, C, *and* t', *with complexities* $|P|$ *processors,* $R+1$ *rounds,* $M + (|P| - |C|) \cdot |C|$ *message bits, and* $\max\{1, S\}$ *message size.*

Proof. Choose any execution of BB at least $R+1$ rounds long.

Termination. We show that after $R+1$ rounds, each nonfaulty processor p in P has some output value for *bb-out*(p). If p is in C, then since BA terminates after R rounds, *ba-out*(p) has some output value at the end of round R, and hence *bb-out*(p) is assigned an output value in round $R+1$. If p is in $P - C$, then the code clearly assigns an output value to *bb-out*(p) in round $R+1$.

Agreement. Suppose at most t' processors in C are faulty. Then by agreement of BA, after R rounds there is some v in $\{0, 1\}$ such that *ba-out*(p) $= v$, for all nonfaulty p in C. Then in round $R+1$, each nonfaulty p in C sets *bb-out*(p) to be v, and each nonfaulty p in $P - C$ receives at least $|C| - t'$ messages for v and at most t' messages for any $w \neq v$. Since $|C| \geqslant 2t' + 1$, a majority of the messages are for v, and thus each nonfaulty p in $P - C$ sets *bb-out*(p) to be v in round $R+1$.

Validity. Suppose at most t' processors in C are faulty, and for some v in $\{0, 1\}$ and for all nonfaulty p in C, *bb-in*(p) $= v$. By validity of BA, after R rounds *ba-out*(p) $= v$, for all nonfaulty p in C. By the same argument as for agreement, *bb-out*(p) $= v$ in round $R+1$, for all nonfaulty p in P.

Complexities. Obviously, the protocol uses $|P|$ processors. One extra round is added by Transformation 1, giving a total of $R+1$ rounds. The message bit complexity is equal to the message bit complexity M of the BA subroutine, plus the maximum number of message bits sent by nonfaulty processors in the extra round, in any execution. The number of additional message bits is maximized in any execution with no faults: in the extra round, each of the $|C|$ processors in C sends one bit to each of the $|P|-|C|$ processors not in C, for a total of $(|P|-|C|)\cdot|C|$. The size of the largest message sent is the maximum of the size S of the largest message sent in the BA subroutine, and the size of the largest message sent in the extra round, which is one. ▮

4.2. *Constructing a Byzantine Agreement Protocol from Several Byzantine Broadcast Protocols*

This subsection describes a simple way to transform a collection of Byzantine broadcast protocols into a Byzantine agreement protocol for P and t.

During the agreement protocol, each correct processor establishes and maintains a *favored value* that it currently prefers as the decision. This favored value can be adjusted as the protocol proceeds. At any time the favored value can be 0 indicating a preference for 0, 1 indicating a preference for 1, or ? indicating no preference. The goal is to bring the favored values of the various processors into agreement on either 0 or 1 while not violating the validity condition.

A collection of disjoint subsets of processors is chosen (when the protocol is written, not executed). For each subset in the collection, all the processors in P execute the following block of rounds. First, two rounds of preliminary message exchanges are performed, during which each processor in P selects an element of $\{0, 1\}$ to use as input to a Byzantine broadcast protocol and computes its current favored value. These preliminary rounds ensure that no two nonfaulty processors favor different elements of $\{0, 1\}$, and if any nonfaulty processor favors a v in $\{0, 1\}$, then v is the common input of all nonfaulty processors in the Byzantine broadcast protocol. Then the processors execute a Byzantine broadcast protocol using the inputs just calculated and using the current subset as the committee. The output of the Byzantine broadcast protocol replaces the favored value for any nonfaulty processor that lacked a preference for an element of $\{0, 1\}$. The number and size of the subsets is chosen in such a way that at least one of the subsets includes sufficiently many nonfaulty processors. After the block of rounds using such a subset, all the processors have the same favored value, an element of $\{0, 1\}$; this favored value persists until the end of the protocol, at which time it is chosen as the output for the Byzantine agreement.

Fix the following for the rest of Section 4:

- an integer t (the number of faults to be tolerated overall);
- a set of processors P, $|P| = n = 3t + 1$ (the collection of processors to reach agreement);
- an integer B, $2 \leqslant B \leqslant t + 1$ (the number of committees);
- integers t_b, where $t_b = \lfloor (t + 1 - b)/B \rfloor$, for $1 \leqslant b \leqslant B$ (the number of faults to be tolerated by the bth committee);
- B disjoint subsets of P, C_b, $1 \leqslant b \leqslant B$ (the set of processors in the bth committee), chosen arbitrarily such that $|C_b| = 3t_b + 1$; and
- Byzantine broadcast protocols BB_b for P, C_b, and t_b, with round complexity R_b, message bit complexity M_b, and message size complexity S_b, $1 \leqslant b \leqslant B$.

A few comments are in order. The t_b's are approximately equal integers whose sum is $t + 1 - B$, as is shown in Lemma 1. Thus, P is large enough to choose the disjoint C_b's (i.e., $\sum_{b=1}^{B} (3t_b + 1) \leqslant 3t + 1$, since $B \geqslant 2$) and, as shown in Lemma 2, as long as there are at most t faulty processors in P, there will be at least one BB_b that operates correctly (i.e., enough processors in C_b are nonfaulty).

We now construct a Byzantine agreement protocol for P and t, using the BB_b's. Rounds are numbered using ordered pairs (b, i), where $1 \leqslant b \leqslant B$ and $1 \leqslant i \leqslant R_b + 2$; the right-hand element of the pair increases faster, giving a lexicographic ordering of the pairs. Let *block* b be rounds $(b, 1)$ through $(b, R_b + 2)$ inclusive.

Transformation 2: *Changing Byzantine Broadcasts to Byzantine Agreement*

Code for processor p in P:

Initialization code:

favor(p) is initialized to the input to processor p

ba-out(p) is initialized to UNINITIALIZED; it will hold p's output

Code for rounds $(b, 1)$ *and* $(b, 2)$, $1 \leqslant b \leqslant B$:

send: **if** *favor*(p) $\in \{0, 1\}$ **then** send *favor*(p) to each processor in P **else** do nothing

receive: receive messages from processors in P

compute: *values*(p) := the multiset of the messages just received, discarding any message that is not an element of $\{0, 1\}$

common(p) := the most common element in *values*(p) (break ties by picking 0 as a default choice)

number(p) := the cardinality of *common*(p) in *values*(p)

if *number*(p) $\geqslant n - t$ **then** *favor*(p) := *common*(p) **else** *favor*(p) := ?

Code for round $(b, 3)$, $1 \leqslant b \leqslant B$:

call BB_b (rounds: R_b, input: *common*(p), output: *bb-out*(p))

Code for rounds $(b, 4)$ *through* $(b, R_b + 2)$, $1 \leqslant b \leqslant B$:

comment: all processors continue the subprotocol

Code added to the end of the compute phase of round $(b, R_b + 2)$, $1 \leqslant b \leqslant B$:

if *favor*$(p) = ?$ **then** *favor*$(p) :=$ *bb-out*(p)

if $b = B$ **then** *ba-out*$(p) :=$ *favor*(p)

Let BA be the protocol obtained by applying Transformation 2 to BB_b, $1 \leqslant b \leqslant B$.

THEOREM 2. *BA is a Byzantine agreement protocol for P and t, with complexities*

- *n processors,*
- $\sum_{b=1}^{B} (R_b + 2)$ *rounds,*
- $\sum_{b=1}^{B} (M_b + 2n^2)$ *message bits, and*
- $\max\{1, S_1, \ldots, S_B\}$ *message size.*

Five lemmas are needed for the proof of Theorem 2. Lemma 1 establishes a relationship between the total number of faults and the fault tolerance of the committees. This relationship is used in the proof of Lemma 2, which states that there is at least one committee whose fault tolerance is not exceeded. Lemma 3 is a technical result used in the proof of Lemma 4. Lemma 4 states that there is a block such that all the nonfaulty processors will favor the same element of $\{0, 1\}$ at the end of that block. Lemma 5 states that once all the nonfaulty processors favor the same v in $\{0, 1\}$, then they will continue to favor v.

LEMMA 1. $\sum_{b=1}^{B} t_b = t + 1 - B$.

Proof. Recall that $t_b = \lfloor (t + 1 - b)/B \rfloor$ for all b, $1 \leqslant b \leqslant B$. Choose non-negative integers x and y such that $t + 1 = xB + y$ and $y < B$. Then

$$t_b = \begin{cases} x & \text{if } 1 \leqslant b \leqslant y; \\ x - 1 & \text{otherwise.} \end{cases}$$

Thus

$$\sum_{b=1}^{B} t_b = y \cdot x + (B - y)(x - 1).$$

Substituting $(t + 1 - y)/B$ for x and simplifying yields the result. ∎

LEMMA 2. *Consider any execution of BA at least B blocks long, in which at most t processors in P are faulty. Then there is some b,* $1 \leqslant b \leqslant B$, *such that at most* t_b *processors in* C_b *are faulty.*

Proof. Suppose not. Then since the C_b are disjoint, there are at least $\sum_{b=1}^{B}(t_b+1)$ faulty processors in P. We then calculate that

$$\begin{aligned}\sum_{b=1}^{B}(t_b+1) &= B+\sum_{b=1}^{B} t_b \\ &= B+t+1-B \qquad \text{by Lemma 1} \\ &= t+1.\end{aligned}$$

Thus we have contradicted the choice of t as the upper bound on the number of faulty processors in P. ▮

We use the following notation (assuming a particular execution of BA). Let $X(p)[b]$ be the value of variable $X(p)$ at the end of block b (i.e., at the end of round (b, R_b+2)). Let $X(p)[0]$ be the value of variable $X(p)$ initially. Let $X(p)[b, r]$ be the value of variable $X(p)$ at the end of round (b, r).

LEMMA 3. *Consider any execution of BA at least B blocks long, in which at most t processors in P are faulty. For all nonfaulty p and q in P, and all b, $1 \leqslant b \leqslant B$, if p sends a message with value $v \in \{0, 1\}$ in round $(b, 2)$ and q sends a message with value $w \in \{0, 1\}$ in round $(b, 2)$, then $v = w$.*

Proof. Suppose p sends v in round $(b, 2)$ and q sends w in round $(b, 2)$. Assume in contradiction $v \neq w$. Then $favor(p)[b, 1] = v$ and $favor(q)[b, 1] = w$. Thus p receives at least $n-t$ messages for v in round $(b, 1)$, and q receives at least $n-t$ messages for w in round $(b, 1)$. Thus at least $n-2t$ nonfaulty processors send messages for v in round $(b, 1)$, and q receives at least $n-2t$ messages for v in round $(b, 1)$. Then q recevies at most $2t$ messages for w in round $(b, 1)$. Since $n > 3t$, it follows that $n-t > 2t$ and q receives less than $n-t$ messages for w in round $(b, 1)$, a contradiction. ▮

LEMMA 4. *Consider any execution of BA at least B blocks long, in which at most t processors in P are faulty. Then there exists a b, $1 \leqslant b \leqslant B$, and a value v in $\{0, 1\}$ such that $favor(p)[b] = v$ for all nonfaulty p in P.*

Proof. By Lemma 2, there is some b such that at most t_b processors in C_b are faulty. Fix the smallest such b. There are three cases.

Case 1. All nonfaulty p in P set $favor(p)$ to be $bb\text{-}out(p)$ at the end of block b. By agreement of BB_b, we are done.

Case 2. All nonfaulty p in P set $favor(p)$ to be $common(p)$ at the end of round $(b, 2)$. Suppose that $common(p)[b, 2] = v$ and $common(q)[b, 2] = w$, for nonfaulty p and q in P. In round $(b, 2)$ processor

p receives at least $n-t$ messages for v, at least $n-2t>t$ of which are from nonfaulty processors. Similarly, in round $(b, 2)$ processor q receives more than t messages for w from nonfaulty processors. By Lemma 3, $v=w$.

Case 3. Some nonfaulty p in P sets $favor(p)$ to be $bb\text{-}out(p)$ at the end of block b and some nonfaulty q in P sets $favor(q)$ to be $common(q)$ at the end of round $(b, 2)$. Suppose $favor(p)[b]=bb\text{-}out(p)[b]=v$, and $favor(q)[b]=common(q, 2)[b]=w$. We show $v=w$. In round $(b, 2)$, q receives at least $n-t$ messages for w, at least $n-2t>t$ of which are from nonfaulty processors. By Lemma 3, no nonfaulty processor sends a message for $w' \neq w$ in round $(b, 2)$. Thus every nonfaulty r receives at least $n-2t$ messages for w in round $(b, 2)$, and at most t messages for $w' \neq w$, causing it to set $common(r)$ to w before calling BB_b; thus, $common(r)[b, 2]=w$. So all nonfaulty processors r in C_b, of which there is at least one, have $common(r)[b, 2]=w$. By validity of BB_b, $bb\text{-}out(r)[b]=w$ for all nonfaulty r, including p. ∎

LEMMA 5. *Consider any execution of BA at least B blocks long, in which at most t processors in P are faulty. Then for all b, $1 \leqslant b \leqslant B$, if for some v in $\{0, 1\}$ and for all nonfaulty p in P, $favor(p)[b-1]=v$, then for all nonfaulty p in P, $favor(p)[b]=v$.*

Proof. All nonfaulty p in P send messages for v in round $(b, 1)$. All nonfaulty p in P receive at least $n-t$ messages for v in round $(b, 1)$, and set $favor(p)$ to v at the end of round $(b, 1)$. Thus in round $(b, 2)$, all nonfaulty p in P send messages for v, and all nonfaulty p in P receive at least $n-t$ messages for v. Since $n>3t$, it follows that $n-t>n/2$, implying v is the most common element of $\{0, 1\}$ received. Thus for all nonfaulty p in P, $number(p)[b, 2] \geqslant n-t$, and p uses $common(p)$ to set $favor(p)$ to v. ∎

Proof of Theorem 2. Choose any execution of BA at least B blocks long.

Termination. By the code and termination of BB_b, every nonfaulty p in P has an output value for $ba\text{-}out(p)$ after B blocks.

Agreement. Suppose at most t processors in P are faulty. By Lemma 4, there is a block b and a v in $\{0, 1\}$ such that $favor(p)[b]=v$ for all nonfaulty p in P. By Lemma 5, $favor(p)[b']=v$ for all nonfaulty p in P, and all $b' \geqslant b$. Thus $favor(p)[B]=v$ for all nonfaulty p in P, and thus $ba\text{-}out(p)[B]=v$ for all nonfaulty p in P.

Validity. Suppose at most t processors in P are faulty. Suppose for some v in $\{0, 1\}$ and for all nonfaulty p in P, the input to p is v. Then $favor(p)[0]=v$ for all nonfaulty p in P, by the code. As in the argument for agreement, $ba\text{-}out(p)[B]=v$ for all nonfaulty p in P.

Complexities. By the definition of the protocol, $|P| = n$. Each block b, $1 \leqslant b \leqslant B$, consists of two overhead rounds and R_b Byzantine broadcast rounds. Each block b, $1 \leqslant b \leqslant B$, contributes n^2 message bits per overhead round (each processor sends one bit to each processor) and M_b message bits from the Byzantine broadcast subroutine. The largest message sent in an overhead round is one bit, and the largest message sent in the bth Byzantine broadcast subroutine has size S_b, $1 \leqslant b \leqslant B$. ▮

4.3. *The General Transformation*

In this subsection we give our general transformation. It is the composition of Transformations 1 and 2.

Recall that B is the number of committees in Transformation 2. Let BA′ be a Byzantine agreement protocol family with processor complexity $3t+1$, round complexity $R(t)$, message bit complexity $M(t)$, and message size complexity $S(t)$. Given B and BA′, we define another Byzantine agreement protocol family BA as follows. (This is Transformation 3.)

We define BA(P, t) for any P and t with $|P| = 3t+1$. Let $t_b = \lfloor (t+1-b)/B \rfloor$, $1 \leqslant b \leqslant B$. Choose B disjoint subsets of P, C_b, $1 \leqslant b \leqslant B$, such that $|C_b| = 3t_b + 1$. Apply Transformation 1 to P and BA′(C_b, t_b) to obtain Byzantine broadcast protocol BB$_b$ for P, C_b, and t_b, for all b, $1 \leqslant b \leqslant B$. Apply Transformation 2 to P, t, and BB$_b$, $1 \leqslant b \leqslant B$, to obtain BA$(P, t)$. Thus Transformation 3 is simply a composition of Transformations 1 and 2.

THEOREM 3. *BA is a Byzantine agreement protocol family with complexities*

- $3t+1$ *processors,*
- $\sum_{b=1}^{B} (R(t_b)+3)$ *rounds,*
- $\sum_{b=1}^{B} (M(t_b)+3(t-t_b)(3t_b+1)+2(3t+1)^2)$ *message bits, and*
- $\max\{1, S(t_1), ..., S(t_B)\}$ *message size.*

Proof. Choose any P and t with $|P| = 3t+1$.

Correctness. By Theorem 1, each BB$_b$ is a Byzantine broadcast protocol for P, C_b, and t_b, $1 \leqslant b \leqslant B$. Then by Theorem 2, BA$(P, t)$ is a Byzantine agreement protocol for P and t.

Complexities. By definition, BA(P, t) uses $3t+1$ processors. The stated number of rounds is correct, since by Theorem 2 the number of rounds is $\sum_{b=1}^{B} (R_b+2)$, and by Theorem 1 each R_b is $R(t_b)+1$. The stated number of message bits is correct, since by Theorem 2 the number of message bits is $\sum_{b-1}^{B} (M_b+2n^2)$, and by Theorem 1 each M_b is $M(t_b)+(|P|-|C_b|)\cdot|C_b|$, where $|P| = n = 3t+1$ and $|C_b| = 3t_b+1$ for all b. By Theorem 2 the maximum message size is $\max\{1, S_1, ..., S_B\}$, and by Theorem 1 each S_b is $\max\{1, S(t_b)\}$, giving the stated bound. ▮

5. The New Byzantine Agreement Protocol

In this section we recursively apply our general transformation to construct our efficient new Byzantine agreement protocol.

Begin by fixing integers $B \geq 2$ and $l \geq 0$. The integer B is the number of committees for Transformation 3. The integer l is a lower bound on the number of faults to be tolerated by each instance of the base protocol. Now we give an inductive definition of a class of protocols BA_i. Each successive member of the class is obtained by an additional application of Transformation 3 (using B as a parameter).

First we define the base protocol family, BA_0. It is a slight variant of the protocol family presented by Lamport, Shostak, and Pease (1982). In the problem solved there, a single processor has an input bit on which the rest must agree; we consider the obvious modification of their protocol to the version of Byzantine agreement that we are studying—a copy of Lamport, Shostak, and Pease's protocol is run for each processor and its input, and processors decide on the majority value in the resulting vector. We call this modified Byzantine agreement protocol family LSP. (Since it terminates in $t+1$ rounds no matter how many faults there are, it satisfies our non-standard termination condition.) Choose any P and t with $|P| = 3t+1$ and $t \geq 0$. Let $\mathrm{BA}_0(P, t)$ be $\mathrm{LSP}(P, t)$.

Assume protocol family BA_{i-1} has been defined, where $i-1 \geq 0$. We now define protocol family BA_i. Choose any P and t with $|P| = 3t+1$. Let $\mathrm{BA}_i(P, t)$ be Transformation 3 applied to B and BA_{i-1} if $t \geq (l+1) \cdot B^i - 1$; otherwise, let $\mathrm{BA}_i(P, t)$ be $\mathrm{BA}_{i-1}(P, t)$. Thus, when t is big enough for i applications of Transformation 3 to be well defined and for each instance of the base protocol to tolerate at least l faults, $\mathrm{BA}_i(P, t)$ is defined to be the BA_0 protocol transformed i times; otherwise, it is defined to be the BA_0 protocol transformed as many times as is possible and still have the base protocols tolerate at least l faults. Lemma 6 below and inspection of the transformations show that BA_{i-1} has the necessary processor complexity, namely $3t+1$, for Transformation 3 to be applicable.

We are now ready to define the protocol family that is the main result of this paper. We define it as one of the BA_i protocols, chosen based on t, the number of faults to be tolerated. First, fix $\varepsilon > 0$; ε is to be the exponent in the message size complexity. Then let $d = \max\{4, \lceil 1/\varepsilon \rceil\}$ and let

$$f(t) = \begin{cases} \left\lceil \log_B \dfrac{d \cdot t \cdot \log\log t}{\log t} \right\rceil & \text{if } t \geq 4; \\ 0 & \text{otherwise.} \end{cases}$$

(The default base for logarithms in this paper is 2.) Now we define the

Byzantine agreement protocol family BA as follows. Given P and t, with $|P| = 3t+1$, let $\mathrm{BA}(P, t)$ be $\mathrm{BA}_i(P, t)$, where $i = f(t)$. The motivation for this definition is that it yields committees at the lowest level of recursion that are of size $O(\log t/\log\log t)$. Given that an exponential message bit algorithm is used in these committees, this is the largest committee size that will still yield an overall message bit complexity of $O(t^2)$. We want committees as large as possible because increasing the size of committees decreases the number of rounds, but we do not want them to be too big because increasing the size of committees increases the size of messages. The role played by d is to reduce the committee size by a factor of d, thereby saving a factor of d in the exponent of the message size complexity. We now state our main result.

THEOREM 4. BA *is a Byzantine agreement protocol family with*

- *processor complexity* $P(t) = 3t+1$,
- *round complexity* $R(t) = t + o(t)$,
- *message bit complexity* $M(t) = O(t^2)$, *and*
- *message size complexity* $S(t) = O(t^e)$.

Five lemmas are used in the proof of Theorem 4. Lemma 6 gives the complexity of the base protocol BA_0. Lemma 7 gives the round complexity of BA_i, for any i. Lemma 8 gives an upper bound on the message size complexity for BA_i, for any i. Lemma 9 gives an upper bound on the message bit complexity of BA_i, for any i. Lemmas 7, 8, and 9 hold only when $t \geqslant (l+1)\cdot B^i - 1$, i.e., when t is large enough that the protocol consists of i transformations of BA_0. However, this will be sufficient for our asymptotic analysis. Lemma 10 is a technical result used to bound the portion of the message bit complexity of BA caused by execution of the LSP protocol at the base level of the recursive construction.

Let BA_i have processor complexity $P_i(t)$, round complexity $R_i(t)$, message bit complexity $M_i(t)$, and message size complexity $S_i(t)$, for all $i \geqslant 0$.

LEMMA 6. BA_0 *is a Byzantine agreement protocol family, and*

- $P_0(t) = 3t+1$,
- $R_0(t) = t+1$,
- $M_0(t) \leqslant (3t+1)^{t+3}$, *and*
- $S_0(t) \leqslant (3t)^t$ *if* $t \geqslant 1$, *and* $S_0(0) = 1$.

Proof. By Lamport, Shostak, and Peasc (1982). ∎

M_0 and S_0 are only defined for nonnegative integers. It will be helpful in the analysis to define nondecreasing functions of nonnegative real

arguments that upper-bound them. Let $M_*(t) = (3t+1)^{t+3}$ be defined for all real $t \geq 0$; let $S_*(t) = \max\{1, (3t)^t\}$ be defined for all real $t > 0$ and let $S_*(0) = 1$. Note that M_* is nondecreasing for all arguments and $M_0(t) \leq M_*(t)$ for all nonnegative integers t, and similarly for S_*.

The next lemma gives an exact expression for the round complexity of BA_i. The expression is equal to the number of rounds in instances of BA_0, which is $t+1$, plus the number of overhead rounds added by our transformation, which is the last term.

LEMMA 7. *$R_i(t) = t + 1 + 3 \cdot \sum_{j=1}^{i} B^j$ for all $i \geq 0$, and all $t \geq (l+1) \cdot B^i - 1$.*

Proof. We proceed by induction on i. The basis, $i = 0$, is true by Lemma 6. Suppose the result is true for $i - 1 \geq 0$. We show it for i. Pick any $t \geq (l+1) \cdot B^i - 1$. By Theorem 3,

$$R_i(t) = \sum_{b=1}^{B} (R_{i-1}(t_b) + 3).$$

Note that for all b

$$\begin{aligned}
t_b &= \lfloor (t+1-b)/B \rfloor && \text{by the definition of } t_b \\
&\geq \lfloor ((l+1) \cdot B^i - b)/B \rfloor && \text{because } t \geq (l+1) \cdot B^i - 1 \\
&\geq \lfloor ((l+1) \cdot B^i - B)/B \rfloor && \text{because } b \leq B \\
&= (l+1) \cdot B^{i-1} - 1.
\end{aligned}$$

Since $t_b \geq (l+1) \cdot B^{i-1} - 1$, we can apply the inductive hypothesis to the above expression for $R_i(t)$ to get

$$\begin{aligned}
R_i(t) &= \sum_{b=1}^{B} \left(t_b + 1 + 3 \cdot \sum_{j=1}^{i-1} B^j + 3 \right) \\
&= \sum_{b=1}^{B} t_b + B + 3B \cdot \sum_{j=1}^{i-1} B^j + 3B \\
&= t + 1 - B + B + 3B \cdot \sum_{j=1}^{i-1} B^j + 3B && \text{by Lemma 1} \\
&= t + 1 + 3 \cdot \sum_{j=1}^{i} B^j. \quad \blacksquare
\end{aligned}$$

The next lemma gives an upper bound on the message size complexity of BA_i. The bound is the message size complexity for the instances of BA_0, each of which tolerates approximately t/B^i faults.

LEMMA 8. *$S_i(t) \leqslant S_*(t/B^i)$ for all $i \geqslant 0$, and all $t \geqslant (l+1) \cdot B^i - 1$.*

Proof. We proceed by induction on i. The basis, $i=0$, follows from Lemma 6 and the definition of S_*. Suppose the lemma is true for $i-1 \geqslant 0$. Choose $t \geqslant (l+1) \cdot B^i - 1$. As shown in the proof of Lemma 7, $t_b \geqslant (l+1) \cdot B^{i-1} - 1$ for all b, allowing use of the inductive hypothesis.

By Theorem 3,

$$S_i(t) = \max\{1, S_{i-1}(t_1), ..., S_{i-1}(t_B)\}.$$

By induction, since each $t_b \geqslant (l+1) \cdot B^{i-1} - 1$,

$$\leqslant \max\{1, S_*(t_1/B^{i-1}), ..., S_*(t_B/B^{i-1})\}.$$

Since S_* is nondecreasing and each $t_b \leqslant t/B$,

$$\leqslant \max\{1, S_*(t/B^i)\}$$
$$= S_*(t/B^i). \quad \blacksquare$$

The next lemma gives an upper bound on the message bit complexity of BA_i. The expression is the sum of two terms. The first term is the message bit complexity of the BA_0 instances: there are B^i instances and each is of a size to tolerate approximately t/B^i faults. The second term is the cost of the overhead rounds. There are B sets of overhead rounds involving all the processors, i.e., $O(Bt^2)$ bits. There are B^2 sets involving a $1/B$ fraction of the processors, i.e., $O(t^2)$ bits, and so on, up to B^i sets involving a $1/B^{i-1}$ fraction of the processors, i.e., $O(t^2/B^{i-2})$ bits. The sum of all these expressions is $O(B \cdot t^2)$, which is $O(t^2)$ since B is fixed.

There exists a constant c such that $3(t-t_b)(3t_b+1) + 2(3t+1)^2 \leqslant ct^2$ for all b. In fact, letting $c=44$ is sufficient. Use of this constant shortens the statement of the bound on message bit complexity given in Theorem 3 and thus simplifies the statement of the next lemma.

LEMMA 9. *$M_i(t) \leqslant B^i \cdot M_*(t/B^i) + c \cdot t^2 \cdot \sum_{k=1}^{i} 1/B^{k-2}$ for all $i \geqslant 0$, and all $t \geqslant (l+1) \cdot B^i - 1$.*

Proof. We proceed by induction on i. The basis, $i=0$, follows from Lemma 6 and the definition of M_*. Suppose the lemma is true for $i-1 \geqslant 0$. Pick any $t \geqslant (l+1) \cdot B^i - 1$. As shown in the proof of Lemma 7, $t_b \leqslant (l+1) \cdot B^{i-1} - 1$ for all b, allowing use of the inductive hypothesis.

By Theorem 3 and the definition of c,

$$M_i(t) \leqslant \sum_{b=1}^{B} (M_{i-1}(t_b) + c \cdot t^2).$$

By induction since each $t_b \geqslant (l+1) \cdot B^{i-1} - 1$,

$$\leqslant \sum_{b-1}^{B} \left(B^{i-1} \cdot M_*(t_b/B^{i-1}) + c \cdot t_b^2 \cdot \sum_{k=1}^{i-1} \frac{1}{B^{k-2}} + c \cdot t^2 \right).$$

Since M_* is nondecreasing and each $t_b \leqslant t/B$,

$$\leqslant \sum_{b=1}^{B} \left(B^{i-1} \cdot M_*(t/B^i) + c \cdot (t/B)^2 \cdot \sum_{k=1}^{i-1} \frac{1}{B^{k-2}} + c \cdot t^2 \right)$$

$$= B^i \cdot M_*(t/B^i) + c \cdot t^2 \cdot \sum_{k=1}^{i} \frac{1}{B^{k-2}}. \quad \blacksquare$$

We need one more lemma before we can prove the main result.

LEMMA 10. *If* $t \geqslant 4$ *then* $((\log t)/\log\log t)^{(\log t)/\log\log t} \leqslant t$.

Proof. Choose $t \geqslant 4$. Using the identity that $x^x = 2^{x \cdot \log x}$, with $x = (\log t)/\log\log t$, we get

$$\begin{aligned} ((\log t)/\log\log t)^{(\log t)/\log\log t} &= 2^{((\log t)/\log\log t) \cdot \log((\log t)/\log\log t)} \\ &\leqslant 2^{((\log t)/\log\log t) \cdot \log\log t} \\ &= 2^{\log t} \\ &= t. \quad \blacksquare \end{aligned}$$

Proof of Theorem 4. For any P and t, $\mathrm{BA}(P, t)$ is defined to be $\mathrm{BA}_i(P, t)$, where $i = f(t)$. Correctness of BA follows from the correctness of the constituent transformations.

We now calculate the four complexity measures for BA. Recall that B, l, and d are fixed constants.

The processor complexity $P(t)$ is $3t+1$. If $f(t) = 0$ this follows from Lemma 6; otherwise, it follows from Theorem 3.

The round, message bit, and message size complexities are calculated for all $t \geqslant t_0$, where t_0 is a constant such that $t_0 \geqslant \max\{4, (l+1) \cdot B^{f(t_0)}\}$. (Such a constant exists since $t_0 \geqslant \max\{4, (l+1) \cdot B^{f(t_0)}\}$ implies $\log t_0 \geqslant (l+1) \cdot d \cdot \log\log t_0$ and log grows faster than log log.) Obviously for all $t \geqslant t_0$, $t \geqslant \max\{4, (l+1) \cdot B^{f(t)}\}$.

For the rest of this proof we assume $t \geqslant t_0$. Thus Lemmas 7, 8, and 9 hold. For clarity of presentation, let $i = f(t)$. We have the following facts.

- *Fact* 1. $i = \lceil \log_B(d \cdot t \cdot (\log\log t)/\log t) \rceil$, since $t \geq 4$.
- *Fact* 2. $\log_B(d \cdot t \cdot (\log\log t)/\log t) \leq i < \log_B(d \cdot t \cdot (\log\log t)/\log t) + 1$, by Fact 1 and the definition of ceiling.
- *Fact* 3. $d \cdot t \cdot (\log\log t)/\log t \leq B^i < B \cdot d \cdot t \cdot (\log\log t)/\log t$, by Fact 2.
- *Fact* 4. $1 \leq t/B^i \leq \log t/(d \cdot \log\log t)$, since $t \geq (l+1) \cdot B^i$ and by Fact 3.

The round complexity is calculated as

$$
\begin{aligned}
R(t) &= R_i(t) \\
&= t + 1 + 3 \cdot \sum_{j=1}^{i} B^j && \text{by Lemma 7} \\
&< t + 1 + 3B^{i+1} \\
&< t + 1 + 3B \cdot B \cdot d \cdot t \cdot (\log\log t)/\log t && \text{by Fact 3} \\
&= t + o(t) && \text{since } B \text{ and } d \text{ are fixed.}
\end{aligned}
$$

The message bit complexity is calculated as

$$
\begin{aligned}
M(t) &= M_i(t) \\
&\leq B^i \cdot M_*(t/B^i) + c \cdot t^2 \cdot \sum_{k=1}^{i} \frac{1}{B^{k-2}} && \text{by Lemma 9.}
\end{aligned}
$$

We now show that each of the two terms is $O(t^2)$. The second term obviously is. The first term is $B^i \cdot M_*(t/B^i)$. By Fact 3, $B^i < B \cdot d \cdot t \cdot (\log\log t)/\log t$. By the definition of M_*,

$$
\begin{aligned}
M_*(t/B^i) &= (3t/B^i + 1)^{t/B^i + 3} \\
&\leq (4t/B^i)^{4t/B^i} && \text{since } t/B^i \geq 1 \text{ (Fact 4)} \\
&\leq \left(\frac{4 \cdot \log t}{d \cdot \log\log t}\right)^{4 \cdot \log t/(d \cdot \log\log t)} && \text{by Fact 4} \\
&\leq \left(\frac{\log t}{\log\log t}\right)^{\log t/\log\log t} && \text{since } d \geq 4. \\
&\leq t && \text{by Lemma 10.}
\end{aligned}
$$

Thus the first term is $O(t^2 \log\log t/\log t) = o(t^2)$.

The message size complexity is calculated as

$$
\begin{aligned}
S(t) &= S_i(t) \\
&\leqslant S_*(t/B^i) && \text{by Lemma 8} \\
&= (3t/B^i)^{t/B^i} && \text{since } t/B^i \geqslant 1 \text{ (Fact 4)} \\
&\leqslant \left(\frac{3 \cdot \log t}{d \cdot \log\log t}\right)^{\log t/(d \cdot \log\log t)} && \text{by Fact 4} \\
&\leqslant \left(\frac{\log t}{\log\log t}\right)^{(\log t/\log\log t)\cdot(1/d)} && \text{since } d \geqslant 4 \\
&\leqslant \sqrt[d]{t} && \text{by Lemma 10} \\
&\leqslant t^{\varepsilon} && \text{because } d \geqslant 1/\varepsilon.
\end{aligned}
$$

This is $O(t^{\varepsilon})$ as desired. ∎

6. Bounding the Round Complexity

In Theorem 4 of Section 5 we gave a bound of $t + o(t)$ on the asymptotic round complexity of our new protocol. In this section we show that the appropriate choice of the parameters B and l to Transformation 3 yields a protocol with a round complexity that is better than $2 \cdot (t+1)$ for all $t \geqslant 0$. In other words, the $o(t)$ term is not hiding a large constant. This result follows from Theorem 5 below, when $B = 4$ and $l = 3$.

Theorem 5. *For all* $i \geqslant 0$, $t \geqslant 0$, $l \geqslant 0$, *and* $B \geqslant 2$,

$$R_i(t) < \left(\frac{3B}{(l+1)(B-1)} + 1\right) \cdot (t+1).$$

Proof. Fix B, l, i, P, and t. Let $k = \max\{0\} \cup \{j\colon 1 \leqslant j \leqslant i \text{ and } t \geqslant (l+1) \cdot B^j - 1\}$. By inspection of the recursive definition of $\mathrm{BA}_i(P, t)$ in Section 5 we see that $\mathrm{BA}_i(P, t) = \mathrm{BA}_k(P, t)$. Thus $R_i(t) = R_k(t)$ and it is sufficient to bound $R_k(t)$. There are two cases.

Case 1. Suppose $k = 0$. In this case $R_k(t) = t + 1$ by Lemma 6. The claimed bound follows immediately since $l \geqslant 0$ and $B \geqslant 2$.

Case 2. Suppose $k \geqslant 1$. From the definition of k it follows that $t \geqslant (l+1) \cdot B^k - 1$. Thus Lemma 7 implies that

$$R_k(t) = t + 1 + 3 \cdot \sum_{j=1}^{k} B^j.$$

Because $B \geq 2$ the sum of the geometric series is strictly less than $B^{k+1}/(B-1)$. After substituting and rearranging we have that

$$R_k(t) < t + 1 + B^k \cdot \frac{3B}{B-1}.$$

Since $t \geq (l+1) \cdot B^k - 1$ if follows that $(t+1)/(l+1) \geq B^k$. Substituting this bound for B^k in the above expression for $R_k(t)$ and rearranging produces the desired result. ∎

7. Other Applications of the Transformation

In this section we use our general transformation to develop a few additional (small) results. Specifically, we consider the effect of recursively applying our transformation to other base protocol families besides LSP.

Suppose we choose a Byzantine agreement protocol family for BA_0 such that $P_0(t) = at + 1$ for some $a \geq 3$, $R_0(t) = t + 1$, $M_0(t) \leq M_*(t)$ for some nondecreasing function M_* with nonnegative real arguments, and $S_0(t) \leq S_*(t)$ for some nondecreasing function S_* with nonnegative real arguments.

Then our analysis can be trivially modified to show that for all $i \geq 0$, and for all sufficiently large t, $P_i(t) = at + 1$, $R_i(t) \leq t + 1 + 3B^{i+1}$, $M_i(t) \leq B^i \cdot M_*(t/B^i) + O(t^2)$, and $S_i(t) \leq S_*(t/B^i)$.

Once one is given a particular choice of the base protocol family, it remains to choose $f(t)$, the function that gives the depth of the recursion when tolerating t faults. The complexities of the resulting protocol family, call them P, R, M, and S, can be obtained by substituting $f(t)$ for i in the above formulas.

Observation 1. If we let $f(t) = \max\{0, \lfloor \log_B(t/(3B)) \rfloor\}$, then, regardless of the base protocol, we get $P(t) = at + 1$, $R(t) \leq 2t + 1$, $M(t) = O(t^2)$, and $S(t) = O(1)$. This choice of $f(t)$ causes the number of faults to be tolerated in each base protocol instance, which is $t/B^{f(t)}$, to be of constant size, $3B$. Since each base protocol instance is of constant size, its contribution to the complexities is constant. Thus if we use a base protocol family with $a = 3$, the resulting protocol family has simultaneously optimal processors, message bits, and message size, at the cost of doubling the number of rounds.

Observation 2. Now suppose that BA_0 is a protocol such as that of Moses and Waarts (1992) or Berman, Garay, and Perry (1989a), so that $a \geq 4$, $M_*(t) = t^x$ for some constant x, and $S_*(t) = t^y$ for some constant $y < x$. Then if we let $f(t) = \lfloor \log_B(t^{(x-2)/(x-1)}) \rfloor$, we get $P(t) = at + 1$, $R(t) \leq$

$t + O(t^{(x-2)/(x-1)})$, $M(t) = O(t^2)$, and $S(t) \leqslant t^{y/(x-1)}$. Compared to using LSP as the base protocol, we have somewhat better round complexity (although still $t + o(t)$) and worse processor complexity and message size complexity. The message bit complexity remains optimal.

Observation 3. Now suppose that BA_0 is the protocol of Berman and Garay (1990) with $a = 3$, $R_0 = t + 1$, $M_* = O(t^3 \cdot 1.5^t)$, and $S_* = O(1.5^t)$. We let $f(t) = 0$ if $t \leqslant 1$, otherwise we let $f(t) = \lceil \log_B(d \cdot t/\log t) \rceil$, where d is defined as in Section 5 to be $\max\{4, \lceil 1/\varepsilon \rceil\}$. The result is that $P(t) = 3t + 1$, $R(t) = t + O(t/\log t)$, $M(t) = O(t^2)$, and $S(t) = O(t^\varepsilon)$ for any ε. Compared to using LSP as the base, we obtain improved round complexity (the $o(t)$ term is $O(t/\log t)$ instead of $O(t \cdot \log\log t/\log t)$), while the other complexities are unchanged.

ACKNOWLEDGMENTS

We thank Jamal Golestani, Will Leland, and the referees for carefully reading earlier versions of this paper, Linda Ness, Abel Weinrib, and George Welch for helpful discussions, and Yoram Moses for encouraging us to improve the constants in an earlier version.

RECEIVED January 2, 1990; FINAL MANUSCRIPT RECEIVED August 1, 1990

REFERENCES

BAR-NOY, A., AND DOLEV, D. (1991), Consensus algorithms with one-bit messages, *Distrib. Comput.* **4**, 105–110.

BAR-NOY, A., DOLEV, D., DWORK, C., AND STRONG, H. R. (1992), Shifting gears: Changing algorithms on the fly to expedite Byzantine agreement, *Inform. and Comput.*, to appear.

BEN-OR, M. (1983), Another advantage of free choice: Completely asynchronous agreement protocols, *in* "Proceedings, 2nd ACM Symposium on Principles of Distributed Computing," pp. 27–30.

BERMAN, P., AND GARAY, J. A. (1989), Asymptotically optimal distributed consensus, *in* "Proceedings, 16th EATCS Colloquium on Automata, Languages, and Programming," pp. 80–94.

BERMAN, P., AND GARAY, J. A. (1990), "Better Masking for Better Consensus," Technical Report CS-90-24, Department of Computer Science, Pennsylvania State University.

BERMAN, P., GARAY, J. A., AND PERRY, K. J. (1989a), Towards optimal distributed consensus, *in* "Proceedings, 30th IEEE Symposium on Foundations of Computer Science," pp. 410–415.

BERMAN, P., GARAY, J. A., AND PERRY, K. J. (1989b), "Recursive Phase King Protocols for Distributed Consensus," Technical Report CS-89-24, Department of Computer Science, Pennsylvania State University.

CHOR, B., AND COAN, B. A. (1985), A simple and efficient randomized Byzantine agreement algorithm, *IEEE Trans. Software Engrg.* **SE-11**, 531–539.

COAN, B. A. (1988), Efficient agreement using fault diagnosis, *in* "Proceedings, 26th Allerton Conference on Communication, Control, and Computing," pp. 663–672.

COAN, B. A., AND WELCH, J. L. (1989a), Modular construction of nearly optimal Byzantine agreement protocols, *in* "Proceedings, 8th ACM Symposium on Principles of Distributed Computing," pp. 295–306.

COAN, B. A., AND WELCH, J. L. (1989b), A Byzantine agreement protocol with optimal message bit complexity, *in* "Proceedings, 27th Allerton Conference on Communication, Control, and Computing," pp. 1062–1071.

DOLEV, D. (1982), The Byzantine generals strike again, *J. Algorithms* **3**, 14–30.

DOLEV, D., FISCHER, M. J., FOWLER, R. J., LYNCH, N. A., AND STRONG, H. R. (1982), An efficient algorithm for Byzantine agreement without authentication, *Inform. and Control* **52**, 257–274.

DOLEV, D., AND REISCHUK, R. (1985), Bounds on information exchange for Byzantine agreement, *J. Assoc. Comput. Mach.* **32**, 191–204.

DOLEV, D., REISCHUK, R., AND STRONG, H. R. (1990), Early stopping in Byzantine agreement, *J. Assoc. Comput. Mach.* **37**, 720–741.

FISCHER, M. J., AND LYNCH, N. A. (1982), A lower bound for the time to assure interacrtive consistency, *Inform. Process. Lett.* **14**, 183–186.

FISCHER, M. J., LYNCH, N. A., AND MERRITT, M. (1986), Easy impossibility proofs for distributed consensus problems, *Distrib. Comput.* **1**, 26–39.

LAMPORT, L., SHOSTAK, R. E., AND PEASE, M. (1982), The Byzantine generals problem, *ACM Trans. Program. Lang. Syst.* **4**, 382–401.

MOSES, Y., AND WAARTS, O. (1992), Coordinated traversal: $(t+1)$-round Byzantine agreement in polynomial time, *J. Algorithms*, to appear.

PEASE, M., SHOSTAK, R. E., AND LAMPORT, L. (1980), Reaching agreement in the presence of faults, *J. Assoc. Comput. Mach.* **27**, 228–234.

RABIN, M. (1983), Randomized Byzantine generals, *in* "Proceedings, 24th IEEE Symposium on Foundations of Computer Science," pp. 403–409.