THE FAILURE DISCOVERY PROBLEM

Vassos Hadzilacos
Computer Systems Research Institute
University of Toronto
6 King's College Road
Toronto, Ontario M5S 1A1
Canada
e-mail: vassos@csri.toronto.edu

Joseph Y. Halpern
IBM Almaden Research Center
Department K53/802
650 Harry Road
San Jose, California 95120–6099
U.S.A
e-mail: halpern@ibm.com

Abstract: We define a simple variant of the Byzantine agreement (BA) problem, called the Failure Discovery (FD) problem that, roughly speaking, amounts to reaching Byzantine Agreement provided that no failures are discovered. We show how a protocol for FD can be extended to one for BA, with no message overhead in the failure-free runs. We also show that, for so-called *benign* failures, if the FD protocol satisfies an additional property, the message-preserving extension to a BA protocol can be accomplished with minimal time overhead in the failure-free runs. Our results show that FD is a useful building block for BA; indeed, it it has been used in this way in a companion paper (Hadzilacos and Halpern [1991]).

This paper is essentially the same as one that appears in *Mathematical Systems Theory* **26**, 1993, pp. 41–102.

October 2, 1996

1. Motivation and overview

Practical fault-tolerant protocols are sometimes composed of two subprotocols: One that ensures correct operation in the absence of failures, and another invoked only when failures do, in fact, occur. The main motivation for such a decomposition is that the first subprotocol is much simpler and efficient than the second. It should be emphasized that the protocol that "works if there are no failures" does have some obligations to fulfill in case there are failures: It must be capable of detecting that failures have occurred and somehow ensuring that the failure-handling subprotocol is invoked. Arguably the best known protocols that are structured in this manner are atomic commitment protocols (cf. Skeen [1982], Bernstein *et al.* [1987]). Another example is the missing writes protocol of Eager and Sevcik [1983] for managing replicated copies of data. Also, similar in spirit are the so-called blast protocols for data transfer (cf. Zwaenepoel [1985]). In all cases, the common theme is cautious optimism: The protocols assume failure-free operation, but provide the mechanisms that can handle failures when these occur.

In our work on the complexity of the Byzantine Agreement (BA) problem in the absence of failures (cf. Hadzilacos and Halpern [1991]), we also found it convenient to decompose the BA protocols in this fashion. This has led us to define the Failure Discovery (FD) problem, which is the formal statement of the problem that the first subprotocol must solve in this case. In the next section of this paper we give the precise specification of the FD problem, some simple protocols that solve it, and compare it to BA. We then show how a FD protocol can be extended to a full-fledged BA protocol without any message overhead and, for benign failures, with only a small time overhead. Furthermore, we show that, under certain additional assumptions, this small time overhead is necessary if we wish to maintain zero message overhead. These results are formulated and discussed in Section 3, while their proofs can be found in Section 4.

Our results suggest that FD is, indeed, the right way to abstract the problem of "solving BA when there are no failures". They imply that if we wish to design BA protocols that are message-optimal in the absence of failures, it is enough to design FD protocols that are message-optimal — a conceptually simpler task. If the BA protocols must also be as time-efficient as possible (subject to being message-optimal) then our results also show that, at least for benign failures, it is enough to design a message-optimal FD protocol that is as time-efficient as possible. We use the results of this paper in precisely this manner in a companion paper where we investigate the complexity of BA in the absence of failures (Hadzilacos and Halpern [1991]), a problem which, in fact, motivated this work.

We caution the reader that our Failure *Discovery* problem is quite different from failure *detection*, a term that has been used with two related, but distinct, meanings in recent literature. As an algorithmic technique, failure detection refers to the identification and "isolation" of faulty processes in a distributed computation. This technique has been used in the design of efficient protocols for BA (for example, Bar-Noy *et al.* [1987], Moses and Waarts [1988]), and other problems (for example, Dwork [1990]). Also, Chandra and Toueg [1991], and Ricciardi and Birman [1991] use the term "failure detector" to refer to a distributed oracle that provides the processes of an asynchronous system with hints about which processes are suspected to be faulty.

2. The Failure Discovery problem

The Byzantine Agreement (BA) problem (Pease *et al.* [1980], Lamport *et al.* [1982], Fischer [1983]) concerns a network of n processes consisting of a distinguished process, the *sender*, and $n-1$ *receivers.* The sender has an *initial value* which it wishes to broadcast to the receivers. The complication is that some of the processes (possibly including the sender) may be faulty, i.e., may not exhibit the behaviour specified by the algorithm they are supposed to execute. The exact number or identity of the faulty processes is not known *a priori.* The BA problem is to design a protocol, i.e., an algorithm for each process, which will ensure the following three conditions in the presence of up to t faulty processes, where t is a fixed parameter between 1 and $n-2$:

Termination: Every correct process eventually chooses a decision value.

Agreement: No two correct processes choose different decision values.

Validity: If the sender is correct then no correct process chooses a value different from the sender's initial value.

Regarding the restriction that $t \leq n-2$, we note that a BA protocol that tolerates $n-2$ failures, trivially tolerates $n-1$ and n failures: The only additional runs we get if more than $n-2$ processes fail are those in which at most one process is correct, in which case it can decide arbitrarily (or its initial value, if it is the sender) without endangering the satisfaction of the BA properties. In this sense, the problem is nontrivial for $t \leq n-2$, and because many results require a special formulation if t is $n-1$ or n, we eschew these uninteresting cases and assume throughout that $n \geq t+2$.

Because it is not known which processes are faulty, designing a correct BA protocol is subtle. Most of the difficulty is due to having to worry about the possibility of failures. Since, in many applications, failures are quite rare a recent trend has been to investigate the difficulty of BA and some of its variants in the special case when no failures occur (cf. Amdur *et al.* [1992], Attiya *et al.* [1990], Hadzilacos and Halpern [1991]). To facilitate such investigations, we introduce here a simpler variant of BA that we call *Failure Discovery* (FD). Roughly speaking, a protocol for FD is one that solves BA provided that no failures are discovered.

To proceed more formally we need to describe a model of computation and to establish some terminology. We use what has become the standard model of computation for the BA problem: The interconnection network is fully connected (so any two processes can exchange messages directly), a process knows the identity of the sender of each message it receives, and system computations proceed in successive synchronous *rounds.* In each round every process can send messages to other processes and receives all messages sent to it by other processes in *that* round. A *run* of a protocol is simply a sequence of rounds. The *view of a process p in (round i of) run r* consists of the sequence of the sets of messages that p receives in each round of r (up to and including round i). In the case of the sender, the view also includes the initial value. We say that *p cannot distinguish runs r and r' (in round i)*, if p has the same view in the two runs (in round i). Since the view of a process encapsulates all the information available to it, the process' actions — the messages it sends, whether to decide and whether to halt — depend exclusively on its view. Thus, if p cannot distinguish two runs in round i, it will send the same messages in round $i+1$ in both runs. Also, if it decides (or halts) in round i in one run, it will decide the same value (or halt) in round i of the other run. A run is called *failure-free* if no process is faulty in it. We say that a process p *discovers a failure* in (round i of) run r if p can distinguish r from all failure-free runs (in round i).[1]

More formally then, the FD problem is to devise a protocol that will ensure the following properties in the presence of up to t faulty processes:

Weak Termination: Each correct process eventually either chooses a decision value or discovers a failure.

Weak Agreement: If no correct process discovers a failure then Agreement holds.

Weak Validity: If no correct process discovers a failure then Validity holds.

The difficulty of solving the BA problem can be quite sensitive to the types of failures that can occur. Not surprisingly, the FD problem is similarly sensitive to the types of failures that can occur. We focus on four failure types here:

a. *Crash failures:* A faulty process stops prematurely; once it has stopped, it sends no more messages. The premature stopping can occur at any point during a round. In particular, a process may stop after having sent some but not all messages it is supposed to send in a round.

b. *Sending omission failures:* A process may fail to send one or more messages prescribed by its algorithm.

c. *General omission failures:* A process may fail to send one or more messages prescribed by its algorithm and/or may fail to receive one or more messages sent to it.

d. *Arbitrary failures:* Faulty processes can act arbitrarily, without any restriction to their possible behaviour.

It is easy to see that crash failures can be viewed as a special case of sending omission failures (where, from a certain point on, a faulty process omits to send all messages), sending omissions failures are a special case of general omission failures, and general omission failures are a special case of arbitrary failures. We refer to crash, sending and general omission failures collectively as *benign* failures.

An appreciation for how simple the FD problem is can be gained by considering the following simple protocol, which achieves FD in the case of general omission failure (and, *a fortiori*, in the case of crash failures and sending omission failures): The sender sends its value to each receiver. In round 1, each receiver decides on the sender's value if it gets a message from the sender, otherwise it discovers a failure. The sender decides on its value. We leave it to the reader to check that this protocol solves the FD problem in the case of general omission failures. For future reference, we call this protocol $\mathbf{D}_0$.

This protocol does not solve the FD problem in the case of arbitrary failures, since in this case, a faulty sender may send different values to different receivers. However, the following two-round protocol does the trick: In round 1, the sender sends its value to

[1] It is interesting to note that by this definition, a process discovers a failure iff it *knows* that some process is faulty, where we use the phrase "a process knows a fact" in the precise sense of knowledge theory in distributed systems (cf. Halpern and Moses [1990]).

each receiver. In round 2, each receiver tells the other receivers what value it got from the sender. At the end of round 2, if a receiver got a value v from the sender, and was informed by each of the other receivers that they also got v from the sender, it decides on v; otherwise it discovers a failure. The sender decides on its value. We leave it to the reader to check that this protocol solves the FD problem in the case of arbitrary failures. For future reference, we call this protocol $\mathbf{D}_1$.

Protocols $\mathbf{D}_0$ and $\mathbf{D}_1$ already show that the FD problem is qualitatively easier than BA. For example, it is well known that BA cannot be solved in the case of arbitrary failures unless $n > 3t$, and that it requires $t + 1$ rounds even in the case of crash failures (see the survey by Fischer [1983] for details and further references). As shown by protocol $\mathbf{D}_1$, FD can be solved for arbitrary values of n and t even if we allow arbitrary failures. Protocol $\mathbf{D}_0$ shows that FD in the case of general omission failures (and hence also crash failures) can be solved in one round; protocol $\mathbf{D}_1$ shows that it can be solved in the case of arbitrary failures in two rounds. (It is easy to see that it cannot be solved in one round in the case of arbitrary failures.)

Another difference regarding the round complexity of FD and BA problems is revealed by the following

Proposition 1: *Let* $\mathbf{D}$ *be a FD protocol so that all processes* decide *in* M *rounds in the failure-free runs. Then we can effectively transform* $\mathbf{D}$ *to a FD protocol* $\mathbf{D}'$ *that tolerates the same type of failures as* $\mathbf{D}$ *such that all correct processes* halt *in* M *rounds in all runs of* $\mathbf{D}'$.

Proof: Consider the situation at the end of round M in a run r of $\mathbf{D}$. If a correct process cannot distinguish r from a failure-free run of $\mathbf{D}$ where it decides v, then it must decide v by the end of round M in r (since all correct processes terminate by the end of round M in failure-free runs of $\mathbf{D}$). If a correct process p can distinguish r from all failure-free runs of $\mathbf{D}$, then p knows that there has been a failure in r and, by definition, p discovers a failure in r. Thus, it follows that protocol $\mathbf{D}'$, where processes proceed just as $\mathbf{D}$ up to the end of round M, and then terminate, achieves FD. ∎

A result like this does not hold for BA protocols. For example, the early-stopping protocol for crash failures of Lamport and Fischer [1982] (cf. also Fischer [1983]) has the property that in the failure-free runs, all processes decide in the first round and halt in the second. Yet, as mentioned before, in any BA protocol for crash failures there must be at least one run that requires $t + 1$ rounds.

Our main motivation for considering FD is to use message-optimal FD protocols to help us to construct BA protocols that are message-optimal in the failure-free runs. The protocols $\mathbf{D}_0$ and $\mathbf{D}_1$ described above are not message-optimal. We now present two message-optimal FD protocols that tolerate general omission failures designed for the special case in where there are only two possible initial values, 0 and 1. These protocols, whose correctness and message optimality is proved in Hadzilacos and Halpern [1991], will be useful in our later discussion.

In the first protocol, GOF1, we partition the $n-1$ receivers into three sets: B, R_0 and R_1, where B has size t. We arrange the processes in R_v, $v = 0, 1$, in a linear order, which

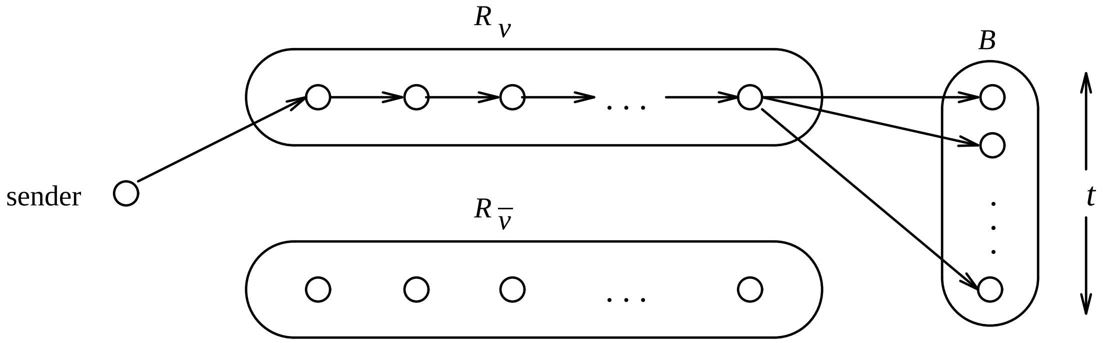

Figure 1: Protocol GOF1 — failure-free run with initial value $v \in \{0, 1\}$

we refer to as a *chain*. If the sender has initial value v, in round 1 it sends the message v only to the first process in R_v in the chain. That message is passed along the chain of processes in R_v. The last process in that chain broadcasts v to all the processes in B. At the end of round $\max(|R_0|, |R_1|) + 1$, a process in R_v decides v if it receives a message, and decides $\overline{v}$ if it receives no message; a process in B decides v if it receives a message containing v and discovers a failure if it receives no message. The sender decides its initial value. The protocol is illustrated in Figure 1.

In the second protocol, GOF2$_v$, we partition the set of receivers into two sets: B and R_v, where $|B| = t - 1$.[2] The processes in each of these sets are arranged in a linear chain. The protocol is illustrated in Figure 2. If the sender has initial value v, it sends v through the B-chain; the message is then passed along the R_v-chain and the last process in R_v broadcasts the message to all the processes in B for a second time, as well as to the sender. If the sender has initial value $\overline{v}$, no message is sent by any process! At the end of round $|B| + |R_v| + 1$, processes decide as follows: A process in R_v decides v if it has received a message and decides $\overline{v}$ if it received no message. A process in B decides v if it received two messages containing v, decides $\overline{v}$ if it received no message, and discovers a failure if it received only one message containing v. As for the sender, if its initial value is $\overline{v}$ then it simply decides $\overline{v}$. If its initial value is v then it decides v if it received a message in round $|B| + |R_v| + 1$; if it received no such message it discovers a failure.

We conclude this section by comparing the FD problem with the *Crusader Agreement* problem of Dolev [1982], which in some respects is similar to FD. Rather than requiring that the conditions of BA hold provided that no failures are discovered, as in FD, in Crusader Agreement the conditions of BA must hold provided that the failure of the *sender* is not discovered. More precisely, a protocol for Crusader Agreement must satisfy Termination, Validity, and the following weakening of Agreement

Agreement′: No two correct processes decide different values unless at least one of them discovers that the sender is faulty — i.e., can distinguish the run from all runs in which the sender is correct.

It is immediate from the definition, that a Crusader Agreement protocol is, *a fortiori*, a FD

[2] This is actually a special case of the protocol GOF2$_v$ as defined in Hadzilacos and Halpern [1991], but it is this particular instance of the protocol that is the most interesting.

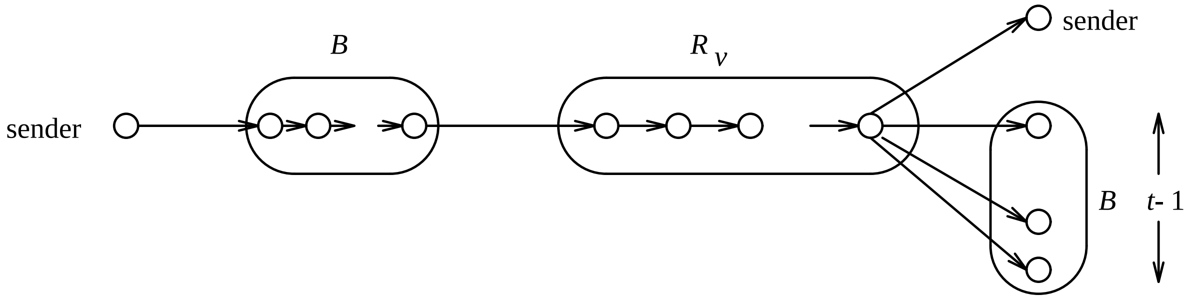

Figure 2: Protocol GOF2$_v$ — failure-free run with initial value v

protocol. The converse is not always true. GOF1 and GOF2 are examples of FD protocols that do not solve the Crusader Agreement problem. On the other hand, $\mathbf{D}_0$ solves the Crusader problem for crash failures, since a process that discovers a failure in fact discovers the sender's failure![3] Also, a modification of $\mathbf{D}_1$ achieves Crusader Agreement for arbitrary failures, if $n > 3t$. (We leave this as an exercise.) However, for arbitrary failures, the requirement $n > 3t$ is necessary for *any* protocol that solves Crusader Agreement — as it is for any protocol that solves BA. In contrast, as $\mathbf{D}_1$ shows, FD can be solved for any value of n.

3. Extending a FD protocol to a BA protocol

We say that a protocol $\mathbf{A}$ is an *extension* of a protocol $\mathbf{B}$, or that $\mathbf{A}$ *extends* $\mathbf{B}$, if every run of $\mathbf{A}$ has a prefix in which precisely the same messages are sent and received as in a run of $\mathbf{B}$, and for every run r of $\mathbf{B}$, there is a run of $\mathbf{A}$ that has a prefix in which precisely the same messages are sent and received as in r. We are interested in constructing BA protocols that extend FD protocols without sending any extra messages in the failure-free runs. This captures the intuition that to solve BA we start by running a FD protocol, and then use the BA protocol only in case a failure is discovered.

A FD protocol is called *pure* if it has the following property: A process decides under the protocol *only* in runs which it cannot distinguish from some failure-free run. In other words, if a process knows that a failure has occurred, it simply discovers a failure and does not decide. All FD protocols that we have described are pure. Given a FD protocol $\mathbf{D}$ that is not pure, we can construct a pure FD protocol $\mathbf{D}'$ that tolerates the same type of failures as $\mathbf{D}$ without adding any messages or rounds — merely by preventing a process from deciding in a run which it can distinguish from all failure-free runs. Since our results concern the extension of FD protocols to BA protocols with minimal message and round overhead, we can assume, without loss of generality, that we are dealing with pure FD protocols. This will simplify some technical definitions.

Our first theorem states that a FD protocol can be extended to a BA protocol with no message overhead in the failure-free runs, regardless of the type of failures. We call a

[3] Actually, to be perfectly accurate, to solve Crusader Agreement we must slightly modify $\mathbf{D}_0$ since we are now required to satisfy the Termination property. Thus, a process that discovers the sender's failure chooses an arbitrary decision value.

BA protocol **A** *special* if, in all runs of **A** where the sender sends no messages and all other processes are correct, no messages are sent at all. Special BA protocols are known for all types of failures (see Srikanth and Toueg [1987] for arbitrary failures, and Hadzilacos [1984] for benign failures). We say that a protocol *halts in M rounds* if all correct processes halt by the end of round M in all its runs. (In case of FD protocols, by Proposition 1, we may assume that this is the same as the number of rounds for processes to decide in the failure-free runs.)

Theorem 1: *Let* **D** *be a FD protocol that tolerates some failure type in whose failure-free runs processes halt within M rounds, let* **A** *be a special BA protocol that tolerates the same failure type in whose runs (not only the failure-free ones!) correct processes halt within N rounds. Then there is an effective way of combining* **D** *and* **A** *to construct a BA protocol* **B** *that extends* **D** *and tolerates the same failure type. Moreover, in each failure-free run of* **B** *there are no messages sent other than those sent by* **D**, *and all processes halt by round $M + N$.*

We defer the proof (and the proofs of all other theorems in this section) to the next section. The significance of this theorem lies in that it reduces the task of devising a BA protocol which is message efficient in the failure-free runs to the (simpler) task of devising a FD protocol which is efficient in the same way.

The transformation from FD to BA protocol in Theorem 1, although message-preserving in the failure-free runs, is far from being round-preserving. Since, as we mentioned above, there is no BA protocol that terminates in less than $t+1$ rounds even in the case of crash failures, this transformation will add at least $t+1$ rounds to the FD protocol, even in the failure-free runs.

Obviously, this round overhead is not always necessary when we extend a FD protocol to a BA protocol. For example, if the FD protocol **D** already happens to be a BA protocol it can be "extended" to a BA protocol with zero message and round overhead! Less trivial examples also exist. For instance, the early-stopping BA protocol of Lamport and Fischer [1982] (cf. also Fischer [1983]) extends $\mathbf{D}_0$ so that in failure-free runs processes decide in round 1 and halt in round 2. However such efficient extensions are *ad hoc*. What we would like is a general way of transforming a FD protocol to a BA protocol which preserves the number of messages but introduces only minimal round overhead in the failure-free runs. It turns out that in the case of benign failures we can do this, provided our FD protocol satisfies an extra property. In order to understand this property, we must consider the *Uniform* Byzantine Agreement (UBA) problem, as defined by Neiger and Toueg [1990]. A protocol for UBA must satisfy Termination and Validity, and the following strengthening of Agreement:

Uniform Agreement: No two processes (*whether correct or faulty*) choose different decision values.

Uniform Agreement is a property of considerable interest. It has been singled out as a key difference between BA and the Atomic Commitment problem (Hadzilacos [1986]). Neiger and Toueg [1990] provide UBA protocols for all types of benign failures and show that, in the case of *general* omission failures, UBA can be achieved only if $n > 2t$. In the case of arbitrary failures, the behaviour of faulty processes is completely unconstrained and thus we cannot hope to satisfy Uniform Agreement.

Here we consider *Uniform FD* (UFD), which bears the same relationship to UBA as FD does to BA. Thus, a protocol for UFD satisfies Weak Termination, Weak Validity, and

Weak Uniform Agreement: If no correct process discovers a failure then no two processes (*whether correct or faulty*) choose different decision values.

Weak Uniform Agreement cannot be achieved if we allow arbitrary failures (with or without authentication). This is so because in this case a process can behave correctly as far as the messages it sends are concerned, but then can choose an arbitrary decision value, quite inconsistent with the messages it sent! Thus, we shall only study UFD for benign failures. In this case, UFD can be achieved. Indeed, all the message-optimal FD protocols for benign failures given in Hadzilacos and Halpern [1991], including GOF1 and GOF2$_v$, satisfy this property, as does $\mathbf{D}_0$. This shows that, unlike UBA, we do not require that $n > 2t$ in order to achieve UFD in the case of general omission failures.

As we now show, UFD protocols can be extended to BA protocols with zero message overhead and an overhead of at most two rounds in the failure-free runs. The exact amount of round overhead for processes to decide and halt depends on the failure type we consider, and whether the protocol satisfies one of two more technical properties.

We say that a UFD protocol $\mathbf{D}$ is *safe* if there is no run of $\mathbf{D}$ in which Validity or Uniform Agreement is violated. (This does not mean that the protocol is a UBA protocol, since Termination may still be violated!) For example, protocol $\mathbf{D}_0$ is safe, as is GOF1 in the special case $n = t+2$. On the other hand, GOF1 is not safe if $n > t+2$, and GOF2$_v$ is not safe for all values of n. We say that the *sender cannot discover a failure in* $\mathbf{D}$ if there is no run of $\mathbf{D}$ in which the sender discovers a failure. For example, the sender cannot discover a failure in $\mathbf{D}_0$ or GOF1, but it can discover a failure in GOF2$_v$.

Because the distinction between deciding and halting features prominently in Theorem 2, it is important to clarify this issue. When we measure the round complexity of a BA protocol, we may be interested in the number of rounds needed for correct processes to decide or to halt. The Termination property implies that a correct process cannot halt unless it has decided, since by definition, a process cannot do anything after it has halted! However, a process cannot necessarily halt as soon as it decides, for its subsequent participation may be required to ensure that other correct processes reach the same decision.

Theorem 2 is somewhat awkward to state. To faciliate its comprehension we have summarised it in Figure 3.

Theorem 2: *Let* $\mathbf{D}$ *be a UFD protocol that tolerates some type of benign failure and halts in* M *rounds. Then there is an effective way to construct protocols* $\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{B}_3$, *and* $\mathbf{B}_4$, *each of which is an extension of* $\mathbf{D}$, *such that in each failure-free run of* $\mathbf{B}_1$, $\mathbf{B}_2$, $\mathbf{B}_3$, *and* $\mathbf{B}_4$, *there are no messages other than those sent by* $\mathbf{D}$. *In addition*

(a) *in the failure-free runs of* $\mathbf{B}_1$, *all processes decide in* M *rounds and halt in* $M+1$ *rounds; if* $\mathbf{D}$ *is safe then* $\mathbf{B}_1$ *is a BA protocol that tolerates the same type of failures as* $\mathbf{D}$.

(b) *in the failure-free runs of* $\mathbf{B}_2$*, all processes decide and halt in* $M+1$ *rounds. Moreover, if* $\mathbf{D}$ *tolerates crash failures then* $\mathbf{B}_2$ *is a BA protocol that tolerates crash failures; if* $\mathbf{D}$ *tolerates sending (resp. general) omission failures and the sender cannot discover a failure in* $\mathbf{D}$ *then* $\mathbf{B}_2$ *is a BA protocol tolerating sending (resp. general) omission failures.*

(c) *in the failure-free runs of* $\mathbf{B}_3$*, all processes decide in* $M+1$ *rounds and halt in* $M+2$ *rounds. If* $\mathbf{D}$ *tolerates sending omission failures then* $\mathbf{B}_3$ *is a BA protocol tolerating sending omission failures (even if the sender can discover a failure in* $\mathbf{D}$*).*

(d) *in the failure-free runs of* $\mathbf{B}_4$*, all processes decide and halt in* $M+2$ *rounds. If* $n > 2t$ *and* $\mathbf{D}$ *tolerates general omission failures then* $\mathbf{B}_4$ *is a BA protocol tolerating general omission failures (even if the sender can discover a failure in* $\mathbf{D}$*).*

BA protocol constructed	Type of failure tolerated	Conditions on UFD protocol D	Round of decision (in f.f. runs)	Round of termination (in f.f. runs)
$\mathbf{B}_1$	any benign	safe	M	$M+1$
$\mathbf{B}_2$	crash any omission	none sender cannot discover failure	$M+1$	$M+1$
$\mathbf{B}_3$	sending omission	none	$M+1$	$M+2$
$\mathbf{B}_4$	general omission	$n > 2t$	$M+2$	$M+2$

Figure 3: The protocols of Theorem 2

It is natural to ask whether the conditions we require of $\mathbf{D}$ are necessary for achieving the round overhead in each part of Theorem 2. For example, in part (a), if the given UFD protocol is not safe, must processes delay their decision until round $M+1$ in any extension of the the UFD protocol that solves BA, or could they still decide in round M? Similar questions arise regarding the other parts. In Theorem 3 we give a partial affirmative answer to these questions. For example, there is a subclass of nonsafe FD protocols, called "nondecisive", for which we can show that in any BA protocol that extends a nondecisive protocol, processes cannot decide before round $M+1$ in the failure-free runs. Analogous results show that the round overhead in the other parts of Theorem 2 is necessary, at least when the FD protocols we wish to extend to BA protocols satisfy certain properties, to which we now turn our attention.

In protocol $\mathbf{B}_2$, processes defer their decision for one round after the termination of $\mathbf{D}$. We now define a subclass of nonsafe FD protocols for which this delay is provably necessary. Recall that a FD protocol is not safe if Uniform Agreement or Validity is violated in some run. A FD protocol is *nondecisive* if Agreement or Validity is violated in some run. It is immediate that a nondecisive protocol is not safe.

For example, GOF1 is nondecisive (and therefore not safe) if $n \geq t+3$. For, in that case there are at least two processes, say p and q, in $R_0 \cup R_1$. Let $v \in \{0,1\}$ be such that $p \in R_v$. If q is also in R_v suppose, without loss of generality, that p precedes q in the R_v-chain. Consider the run in which the sender has initial value v and the only faulty process is p, which omits to pass the message containing v on to the next process in the chain. Thus in this run q will decide $\overline{v}$, in violation of Validity. If $q \in R_{\overline{v}}$, consider the run in which the sender has initial value v and the predecessor p' of p is the only faulty process and omits to pass the message v on to p. Since neither p nor q receive a message in this run, they will decide $\overline{v}$ and v, respectively, so we have a violation of Agreement.

Next, we define three related and increasingly restricted subclasses of FD protocols in which the sender can discover a failure, for which we can show that the round overhead for deciding and halting exhibited by $\mathbf{B}_3$ and $\mathbf{B}_4$, as well as the requirement $n > 2t$ for the latter, are necessary. We say that a FD protocol $\mathbf{D}$ is *weakly sender-dependent* if there exist two runs r and r' of $\mathbf{D}$ with the following properties, where v is any decision value:

SD1. In r the sender is correct, has initial value v and discovers a failure; no other correct process discovers a failure; and Validity is violated (i.e., some correct process decides $v' \neq v$).

SD2. In r' the sender is faulty, has initial value v and discovers a failure; all other processes are correct and do not discover a failure.

SD3. No process except possibly those that are faulty in r can distinguish r and r'.

A FD protocol is *sender-dependent* if it satisfies SD1, SD2, and a stronger version of SD3, where all processes (including those that are faulty in r) cannot distinguish r and r'; i.e. it satisfies

SD3′. No process can distinguish r and r'.

Finally, a FD protocol is *strongly sender-dependent* if it is sender-dependent and, in addition, SD1 is strengthened to

SD1′. SD1 holds and the number of faulty processes in r is at most $t-1$.

For example, GOF2$_v$ is weakly sender-dependent. To see this, let p be the process to which the sender sends its initial value when that value is v. Let r be the run in which the sender has initial value v, and the only faulty process is p, which does not pass on the sender's message, and let r' be the run in which the sender has initial value v but omits to send the message to p. It is straightforward to verify that these two runs satisfy SD1–SD3. Furthermore, in the case of general omission failures, GOF2$_v$ is actually sender-dependent (not just weakly so). To see this, slightly modify the run r described previously, so that p omits to receive the message sent to it by the sender (rather than omitting to send a message to the next process in the chain). Note that now not even p can tell the difference between r and r' (whereas in the scenario described before it could, as in r it receives a message from the sender while in r' it does not). In fact, if $t > 1$ then the scenario we just described actually shows that GOF2$_v$ is *strongly* sender-dependent, since r has only one faulty process, p. If $t = 1$ then no protocol can be strongly sender-dependent as, in that case, the definition would require the existence of a run r with at most $t - 1 = 0$ faulty processes in which the sender discovers a failure, which is impossible!

Theorem 3: *Suppose* $\mathbf{D}$ *is a FD protocol that tolerates some type of benign failure and halts in* M *rounds, and* $\mathbf{B}$ *is an extension of* $\mathbf{D}$ *that achieves BA and tolerates the same type of failures, such that the only messages sent in the failure-free runs of* $\mathbf{B}$ *are those sent by* $\mathbf{D}$*. Then* *(a) if* **D** *is nondecisive then some process does not decide before round $M+1$ in some failure-free run of* **B**.

(b) if **D** *is weakly sender-dependent, and* **D** *and* **B** *tolerate sending omission failures then some process does not halt before round $M+2$ in some failure-free run of* **B**,

(c) if **D** *is sender-dependent, and* **D** *and* **B** *tolerate general omission failures then some process does not decide before round $M+2$ in some failure-free run of* **B**.

(d) if **D** *is strongly sender-dependent,* **D** *and* **B** *tolerate general omission failures, all processes decide by round $M+2$ in all failure-free runs of* **B**, *and $n \geq 4$, then we must have $n > 2t$.*

Notice that in part (d) of the theorem, the assumption that the correct processes decide in $M+2$ is necessary. If we do not put constraints on when the correct processes decide, then it follows from Theorem 1 that we do not need to assume $n > 2t$.

Theorem 3 is useful, despite the seemingly esoteric nature of the properties for which we were able to show that the round overhead of the constructions in Theorem 2 is necessary. In Hadzilacos and Halpern [1991] we present lower bounds on the number of rounds required by message-optimal FD protocols. Our proofs yield enough information about the structure of minimal-round message-optimal protocols that we can actually show that such protocols must have the properties mentioned in Theorem 3. Thus, we can use this theorem to establish that the constructions of Theorem 2, if applied to our minimal-round message-optimal FD protocols, yield BA protocols that are message-optimal and have minimal round complexity in the failure-free runs.

We conclude this section with an interesting aside, concerning the extension of UFD protocols to protocols that solve *Weak Byzantine Agreement (WBA)*, a variation of BA defined by Lamport [1983]. A protocol that solves the WBA problem must satisfy Termination, Agreement and the following weakening of Validity

Weak Validity′: If there are no failures then Validity holds.

It turns out that all the intricacies of extending a UFD protocol to a BA protocol exhibited in Theorem 2 are really due to the need to ensure the Validity property. If instead of BA we are interested in WBA we have

Theorem 4: *Let* **D** *be a UFD protocol that tolerates any type of benign failure and halts in M rounds. Then there is an effective way to extend* **D** *to a WBA protocol* **B** *that tolerates the same type of failures so that in each failure-free run of* **B** *there are no messages other than those sent by* **D**, *and all processes decide in round M and halt in round $M+1$.*

Note that there are no restrictions on **D** beyond the fact that it is a UFD protocol: It need not be safe, and the sender may discover failures.

4. Proofs of theorems

The following simple lemma will be useful in the proofs of Theorems 1 and 2.

Lemma 1: *Let* **D** *be a FD (resp. UFD) protocol that tolerates at least crash failures, and r be a run of* **D** *in which at most t processes are faulty or discover a failure. Then Agreement (resp., Uniform Agreement) is not violated in r. Furthermore, if the sender does not discover a failure, Validity is not violated (resp. no process decides a value other than the sender's initial value) in r.*

Proof: Suppose, by way of contradiction, that Agreement (resp. Uniform Agreement) is violated in r. Consider the run r' which is just like r except that in the last round all processes that discover a failure in r crash after sending their messages (if any). Because **D** is a (pure) FD (or UFD) protocol, any process that is correct and decides in r is also correct and decides (indeed, on the same value) in r'. Thus, r' is a run of **D** with at most t faulty processes, in which no correct process discovers a failure, yet Agreement (resp. Uniform Agreement) is violated, contradicting that **D** is a FD (resp. UFD) protocol. Now suppose that the sender does not discover a failure. This means that the sender decides its initial value, say v. Suppose, by way of contradiction, that Validity is violated. This means that the sender is correct but some correct receiver p decides $v' \neq v$, which is a violation of Agreement, which we just proved is impossible. (In case of Uniformity, suppose, by way of contradiction, that some process decides $v' \neq v$. This violates Uniform Agreement, which we just proved is impossible.) ▮

Proof of Theorem 1: The basic idea of this construction is very simple: The FD protocol **D** will be used first and the BA protocol **A** will be used as a backup mechanism only if necessary — i.e., if some process discovers a failure. For this to work properly however, there must agreement among the (correct) processes whether **A** should be used as a backup mechanism. We can achieve such agreement by using **A**! This, of course, seems circular: We use **A** to determine if **A** should be used. As we shall see shortly, special protocols allow us to avoid this circularity.

We now explain how to construct **B** using **D** and **A**. Without loss of generality, let 0 be the decision value of correct processes in the run of **A** in which the sender sends no messages and all other receivers are correct. Note that because **A** is special, no messages are sent in this run. In **B** there is a designated sender with an initial value it wishes to broadcast to $n-1$ receivers. Each process p works as follows in **B**:

In rounds 1 through M, p proceeds just as it does in **D**, playing the role of the sender with initial value v if and only if it is the sender in **B** with initial value v.

During rounds $M+1$ through $M+N$, p is involved in n simultaneous invocations of **A**, acting as the sender in exactly one of these invocations as follows:

> If, by the end of protocol **D** (i.e., by round M), p has discovered a failure, then in round $M+1$ it invokes **A** to broadcast this fact. It does this by using 1 as its initial value. On the other hand, if p has reached a decision at the end of **D**, then it does nothing in the invocation of **A** in which it is the sender. By Termination and Agreement of **A**, by round $M+N$ all correct processes will agree on whether p discovered a failure or not. Furthermore, by Termination and Validity of **A**, if p is correct and discovered a failure, then all correct processes will agree that it did.

Thus, by the end of round $M+N$, process p will find itself with n decisions for the equally many invocations of **A** in which it participated. There are two cases.

1) If all of these decisions are 0 then no correct process discovered a failure at the end of protocol **D**. In this case, p will simply adopt its decision at the end of **D** as its final decision for **B**. (Note that in this case, by Weak Termination of **D**, and Termination and Validity of **A**, p must have reached a decision at the end of **D**.)

2) If, on the other hand, any one of the decisions is not 0, then p will participate in yet another invocation of **A**. However, in this case p will play the role of the sender iff it is the sender in **B** and, in that case, it will use as its initial value in this invocation of **A** the initial value that it wishes to broadcast in **B** (and which it used in **D**). By the end of round $M+2N$, p will adopt its decision in this invocation of **A** as its final decision for **B**.

To show that the constructed protocol **B** is a correct BA protocol we consider two cases.

Case 1: Some process p that is correct in the entire execution of **B** decides a value other than 0 in at least one of the n invocations of **A** in which it participated in rounds $M+1$ through $M+N$. By Agreement for **A**, all correct processes will decide a value other than 0 in that invocation of **A**. Thus, by (2), all correct processes will participate in the final invocation of **A** (in rounds $M+N+1$ through $M+2N$) and because **A** is a correct BA protocol they will choose the same decision value. If the sender is correct and has initial value v, they will all decide v. So **B** satisfies the BA properties in this case.

Case 2: Every process p that is correct in the entire execution of **B** decides 0 in all n invocations of **A** in which it participated in rounds $M+1$ through $M+N$. Thus, at the end of round $M+N$ all correct processes adopt their decision under **D** as their final decision under **B** and halt. It suffices to show that the decisions reached under **D** at the end of round M satisfy Agreement and, if the sender is correct throughout **B** then they also satisfy Validity. At most t processes are faulty or discover a failure in the execution of **D** in the first M rounds. (Otherwise, there would be some process that discovers a failure in **D** which remains correct throughout **B**. Such a process would invoke **A** with initial value 1 in round $M+1$. By Termination and Validity of **A**, all correct processes would then decide 1 in at least that invocation of **A**, contradicting the hypothesis of Case 2.) By Lemma 1, the decisions reached under **D** satisfy Agreement. Now suppose that the sender is correct throughout **B**. By Lemma 1 again, either the decisions reached under **D** satisfy Validity — in which case we are done — or the sender discovers a failure in **D**. In the latter case, the sender invokes **A** with initial value 1 in round $M+1$ and, because we assume that it is correct throughout **B**, by Termination and Validity of **A** all correct processes will decide 1 in that invocation, contrary to the hypothesis of Case 2.

Finally, we show that in each failure-free run of **B**, there are no messages sent other than those sent by **D**, and all processes halt by round $M+N$. Consider any failure-free run of **B**. Since the run is failure-free, no process can discover a failure. By the Weak Termination property of **D**, at the end of round M, all processes will have reached a decision. Thus, none of them will send any messages in the invocation of **A** in rounds $M+1$ through $M+N$ in which it acts as the sender. Since, by assumption, **A** is special, these invocations will therefore generate no messages and will all result in decision 0. Hence, in any failure-free run of **B**, all processes will reach their final decision in round $M + N$ without any messages other than those used by protocol **D**. ▮

Proof of Theorem 2: The basic idea in all the constructions is that processes which discovered a failure at the end of **D** broadcast that fact to inform others that Agreement and Validity may be violated. A process that receives a "failure discovered" message and did reach a decision under **D** must then disseminate its decision to all processes.

(a) In protocol $\mathbf{B}_1$, each process p executes the following algorithm:

In rounds 1 **through** M**:** p executes protocol **D**; if it reaches a decision v in protocol **D**, it adopts v as its decision under protocol $\mathbf{B}_1$.

In round $M + 1$**:** If p discovered a failure under **D**, it broadcasts the message "failure discovered".
If p does not receive a "failure discovered" message in round $M + 1$, it halts.

Any process that halts in round $M + 1$ does not, of course, participate in subsequent rounds. The remaining part of the protocol will be referred to as the *relay protocol.* It runs for $t + 1$ rounds, starting in round $M + 2$:

In round $M + 2$**:** If p is the sender and has initial value v, it broadcasts $\langle S, v\rangle$.
If p is a receiver and reached a decision v in round M under **D**, it broadcasts $\langle R, v\rangle$.

In rounds $M + 3$ **through** $M + t + 2$**:** If p received $\langle tag, v\rangle$ (where tag is either S or R) in the previous round and had not received this message before, it broadcasts $\langle tag, v\rangle$.

At the end of round $M + t + 2$**:** Let X_p be the set of triples received by p during the relay protocol. We say that X_p *contains a value* v if it contains $\langle tag, v\rangle$, for some tag. Process p chooses its decision value depending on the set X_p according to the following rule:

Decision rule for $\mathbf{B}_1$**:** If X_p contains a single value v, then p decides v; if X_p contains $\langle R, v\rangle$ and does not contain $\langle R, v'\rangle$ for $v' \neq v$, then p decides v (even if X_p contains $\langle S, v'\rangle$); otherwise p chooses a default value as its decision.

It may seem somewhat counterintuitive (and, indeed, may seem to contradict Validity) that the sender's value is discarded when there is a conflict with a receiver's value, but as we shall see, this can only happen if the sender is faulty.

It is clear from the construction that in the failure-free runs of $\mathbf{B}_1$, precisely the same number of messages are sent as in the corresponding run of **D**, and correct processes decide in round M and halt in round $M+1$. It remains to show that $\mathbf{B}_1$ is a BA protocol provided that **D** is safe. In order to prove this, we first make some observations on the properties of the relay protocol, which will also apply in our later protocols. In the claims below, we talk about round ℓ of the relay protocol, which is actually round $M + 1 + \ell$ of protocol $\mathbf{B}_1$. A straightforward induction on ℓ shows that

Claim 1: *If a process p that participates in the relay protocol receives a message $\langle tag, v\rangle$ in some round ℓ of the relay protocol, $1 \leq \ell \leq t+1$, then there is a sequence $p_1, p_2, \ldots, p_{\ell+1} = p$ of distinct processes such that $\langle tag, v\rangle$ was sent by p_i and received by p_{i+1} in round i of* *the relay protocol, for all $1 \leq i \leq \ell$. Furthermore, v is either the decision reached by p_1 under* **D***, or the sender's initial value (in which case p_1 is the sender).*

Next we show that the relay protocol ensures that, for any message $\langle tag, v\rangle$, either all or none of the correct processes that participate receive that message. To see this, consider the earliest round ℓ, $1 \leq \ell \leq t+1$, in which a correct process, say p, receives $\langle tag, v\rangle$. The choice of ℓ, the fact that in Claim 1 the processes involved in the chain that propagates $\langle tag, v\rangle$ are distinct, and the fact that there are at most t faulty processes, imply that $\ell < t+1$, so round $\ell+1$ exists. Since p is correct it will broadcast $\langle tag, v\rangle$ in round $\ell+1$ and therefore all correct processes will also receive that message. Hence we have that

Claim 2: *The set of correct processes that participate in the relay protocol receive exactly the same set of $\langle tag, v\rangle$ pairs.*

We are now ready to prove that $\mathbf{B}_1$ is a BA protocol if **D** is a safe FD protocol. Consider a run r of $\mathbf{B}_1$. Obviously Termination holds since all processes decide by round $M+t+2$. Now we prove that Agreement and Validity hold as well. Since **D** is safe, the decisions reached by correct processes under **D** do not violate (Uniform) Agreement or Validity. Thus, all correct processes that decide in round M will choose the same decision value, say v, and if the sender is correct then v is the sender's initial value. Also, by Claim 2, all correct processes that decide in round $M+t+2$ will choose the same decision value. It remains to show that if

(i) the sender is correct and has initial value v, or

(ii) some correct process p decides v in round M,

then no correct process q decides $v' \neq v$ in round $M+t+2$. Suppose, by way of contradiction, that (i) or (ii) holds, yet a correct process q decides $v' \neq v$ in round $M+t+2$. Since q did not reach a decision in round M, it will send a "failure discovered" message in round $M+1$. The sender or p, according as (i) or (ii) holds, will receive that message, and in round $M+2$ will broadcast $\langle tag, v\rangle$, which q will receive in the same round. Given the decision rule, the only way that q can decide v' is if it also receives $\langle R, u\rangle$, for some $u \neq v$ (possibly $u = v'$), during the relay protocol. By Claim 1, this means that some receiver sends this message in round $M+2$, and therefore that receiver had decided u in round M under **D**. Since (i) or (ii) holds, this means that Validity or Uniform Agreement is violated by **D**, which contradicts the assumption that **D** is safe.

(b) Protocol $\mathbf{B}_2$ is quite similar to $\mathbf{B}_1$. One difference is that a process p that does not discover a failure under **D** does not decide in round M. Instead, it waits until round $M+1$: If it does not receive a "failure discovered" message in round $M+1$, p then adopts the decision it reached under **D** as its decision under **B** and halts. Otherwise, p participates in the relay protocol, as in $\mathbf{B}_1$. (As Theorem 3(b) shows, the delay of the decision by one round is necessary in general.) The other difference is that in round $M+t+2$, p decides using a slightly different decision rule than $\mathbf{B}_1$ (recall that X_p is the set of pairs received by p during the relay protocol):

Decision rule for $\mathbf{B}_2$: If X_p contains contains a single value v, then p decides v; if X_p contains $\langle S, v\rangle$, then p decides v; otherwise p chooses a default value as its decision.

In other words, if X_p contains a value broadcast by the sender and a different value broadcast by a receiver, the sender's decision value is given priority. (This is in contrast to the decision rule of $\mathbf{B}_1$ which gives preference to the receiver's value.)

It is straightforward to check that in the failure-free runs of $\mathbf{B}_2$ there are no messages other than those used in **D**, and that processes decide and halt in round $M+1$. We now show that if **D** tolerates crash failures, then $\mathbf{B}_2$ is a BA protocol tolerating crash failures, while if **D** tolerates sending (resp. general) omission failures, and the sender cannot discover a failure in **D**, then $\mathbf{B}_2$ is a BA protocol tolerating sending (resp. general) omission failures. $\mathbf{B}_2$ certainly guarantees Termination since every process reaches a decision by round $M+t+2$. To show that Agreement and Validity are also satisfied, we consider two cases. Let r be any run of $\mathbf{B}_2$.

Case 1: Some process, say p, that is correct in r, discovered a failure under **D** (i.e., by round M of r). In this case p broadcasts "failure discovered" in round $M+1$. Therefore no correct process will decide in round $M+1$ and all will participate in the relay protocol. By Claim 2, at the end of round $M+t+2$ they will all have received the same set of pairs, so they will all choose the same decision, proving Agreement. Regarding Validity, assume that the sender is correct and has initial value v. The sender will broadcast $\langle S, v\rangle$ in round $M+2$ and all correct processes will receive it. The decision rule for $\mathbf{B}_2$ guarantees that all correct processes will decide v.

Case 2: No process that is correct in r discovered a failure during the execution of protocol **D**. There are at most t processes that are faulty or discovered a failure in **D**. (Otherwise, some process that discovered a failure in **D** would remain correct throughout r, contrary to the hypothesis of Case 2). Thus, by Lemma 1,

(*) The decisions reached under **D** in round M satisfy Uniform Agreement and, if the sender does not discover a failure, they are equal to the sender's initial value.

From (*) we conclude that all processes that decide in round $M+1$, reach the same decision and, furthermore, if the sender is correct in r then that decision will be the sender's initial value. (Regarding the second conclusion, note that by hypothesis of Case 2, since the sender is correct in r it cannot have discovered a failure in **D**.) By Claim 2, all correct processes that decide in round $M+t+2$ will have received the same set of pairs in the relay protocol and will therefore decide on the same value. It remains to show that if

(i) the sender is correct and has initial value v, or

(ii) some correct process p decides v in round $M+1$,

then any correct process that decides in round $M+t+2$ must also decide v. Assume, by way of contradiction, that (i) or (ii) holds, yet some correct process q decides $v' \neq v$ in round $M+t+2$. Thus, q must have decided under **D** in round M (by the hypothesis of Case 2) but received a "failure discovered" message in round $M+1$. Since (i) or (ii) holds, by (*), the value that q decided under **D** must have been v. (In case (i), recall that the sender cannot discover a failure, by hypothesis of Case 2.) Thus, q will broadcast and receive $\langle tag, v\rangle$ in round $M+2$. Given the decision rule, the only way for q to decide v' in round $M+t+2$ is if it also receives $\langle S, v'\rangle$ or $\langle R, u\rangle$ for some $u \neq v$ (possibly $u = v'$) during the relay protocol. By Claim 1, this means that such a message is broadcast in round $M+2$. We shall show that this leads to a contradiction. If a receiver sent $\langle R, u\rangle$ in round $M+2$, it must have decided u under $\mathbf{D}$ in round M. Given that (i) or (ii) holds this contradicts (*). Now, if the sender sends $\langle S, v' \rangle$ in round $M+2$, (i) cannot hold. It remains to derive a contradiction if case (ii) holds and the sender sends such a message. First we shall argue that, in this case, the sender did not discover a failure in $\mathbf{D}$. This is true by hypothesis if we are dealing with sending or general omission failures. For crash failures we reason as follows: Since the sender sent $\langle S, v' \rangle$ in round $M+2$, it must have successfully sent all messages in previous rounds. If the sender had discovered a failure in $\mathbf{D}$, it would have sent "failure discovered" in round $M+1$ which would have been received by p (because p is correct), and p would not have decided in round $M+1$, contrary to assumption. Thus, the sender did not discover a failure in $\mathbf{D}$ and, by (*), the decision reached by any process in $\mathbf{D}$ is the same as the sender's initial value. By (ii), p decided v in $\mathbf{D}$ and so the sender's initial value must have been v, which contradicts the assumption that the sender sent $\langle S, v' \rangle$ in round $M+2$. Since neither $\langle R, u \rangle$, for $u \neq v$, nor $\langle S, v' \rangle$ can be sent in round $M+2$, q cannot decide v' in round $M+t+2$, as wanted.

(c) The structure of protocol $\mathbf{B}_3$ is quite similar to that of the previous two protocols. The main difference is that there are now two rounds between the end of $\mathbf{D}$ and the relay protocol. The rule used to decide at the end of the relay protocol is that used in $\mathbf{B}_1$, not $\mathbf{B}_2$. More specifically, for each process p,

In rounds 1 **through** M**:** p executes protocol $\mathbf{D}$.

In round $M+1$**:** If p discovered a failure under $\mathbf{D}$, it broadcasts the message "failure discovered".
If p receives a "failure discovered" message in round $M+1$, it discards its decision under $\mathbf{D}$, if any. Otherwise, p adopts its decision under $\mathbf{D}$ as its decision under $\mathbf{B}$.

In round $M+2$**:** If p did not reach a decision in round $M+1$, it broadcasts the message "no decision".
If p decided in round $M+1$ and does not receive a "no decision" message in round $M+2$, it halts.

Again, in rounds $M+3$ to $M+t+3$, processes that have not halted take part in a relay protocol. (Note that the relay protocol is now "shifted" later by one round.) It is important to emphasize that processes which discarded their decision under $\mathbf{D}$ in round $M+1$ will *not* broadcast a message in the first round of the relay protocol (round $M+3$), Processes decide in round $M+t+3$ using the decision rule of $\mathbf{B}_1$.

It is clear from the construction that in the failure-free runs of $\mathbf{B}_3$, precisely the same number of messages are sent as in the corresponding run of $\mathbf{D}$, and correct processes decide in round $M+1$ and halt in round $M+2$. Also, $\mathbf{B}_3$ satisfies Termination since correct processes decide in all runs by round $M+t+3$. To show that $\mathbf{B}_3$ satisfies Agreement and Validity consider the same two cases as in the proof of correctness of $\mathbf{B}_2$. Let r be any run of $\mathbf{B}_3$.

Case 1: Some process, say p, that is correct in r, discovered a failure under $\mathbf{D}$ (i.e., by round M of r). The argument that Agreement holds is identical to the corresponding case of the previous protocol. Regarding Validity, assume that the sender is correct and has initial value v. The sender will broadcast $\langle S, v \rangle$ in round $M+3$ and all correct processes will receive it. Furthermore, any process (whether correct or faulty) that participates in the relay protocol will have received the "failure discovered" message sent by p (this is where we use the fact that we are dealing with *sending* omissions only) and will discard its decision under $\mathbf{D}$, if any. Thus the only message broadcast in the relay protocol is $\langle S, v\rangle$, so all correct processes will decide v, the sender's initial value, as required for Validity.

Case 2: No process that is correct in r discovered a failure during the execution of protocol $\mathbf{D}$. The proof is quite similar to the corresponding case of protocol $\mathbf{B}_2$, except for the step proving that if

(i) the sender is correct and has initial value v, or

(ii) some correct process p decides v in round $M+1$,

then any correct process that decides in round $M+t+3$ must also decide v. To see why this holds now, assume, by way of contradiction, that (i) or (ii) holds, yet some correct process q decides $v' \neq v$ in round $M+t+3$. Thus, q must have received a "failure discovered" message in round $M+1$. Since q is correct, it must have sent a "no decision" message in round $M+2$ that was received by all correct processes. Thus, the sender or p (depending on whether (i) or (ii) holds) will broadcast $\langle tag, v\rangle$ in round $M+3$ and this message will be received by q (because the sender of the message as well as q are correct). Thus, given the decision rule for $\mathbf{B}_3$, the only way that q could decide v' in round $M+t+3$, is if it also received $\langle R, u\rangle$, for some $u \neq v$ (possibly $u = v'$). By Claim 1, such a message must have been sent in round $M+3$. However, if (i) or (ii) holds no such message could have been sent because, by (*), the decision values reached under $\mathbf{D}$ (which are the only values that can be broadcast by receivers in round $M+3$) cannot be different from v. Thus, if (i) or (ii) holds, no correct process decides $v' \neq v$ in round $M+t+3$, as wanted.

(d) Finally, we describe $\mathbf{B}_4$. This protocol is identical to $\mathbf{B}_3$, except for three differences: Firstly, processes do not decide in round $M+1$; rather, they delay their decision until round $M+2$, making one only if they do not receive a "no decision" message. (As Theorem 3(d) shows, such a delay is necessary in general.) Secondly, if the sender sends a "failure discovered" message in round $M+1$ and does not receive at least $t+1$ "no decision" messages in round $M+2$, it halts in round $M+2$ (since it knows it must be faulty) and does not participate in the relay protocol. Finally, we use the decision rule of $\mathbf{B}_2$, not that of $\mathbf{B}_1$.

The proof of correctness proceeds along the same lines as those of $\mathbf{B}_2$ and $\mathbf{B}_3$. Termination is obvious. For Agreement and Validity we consider our familiar two cases. Case 1 is similar to the corresponding case in the previous two protocols, except for the proof that Validity holds. (We urge the reader to determine why the proof of Validity in the corresponding case of $\mathbf{B}_3$ does not work now, as this reveals the subtle differences between the two protocols.) Here is why Validity holds now: Suppose that the sender is correct and has initial value v. Since in Case 1 some correct process p discovered a failure in $\mathbf{D}$, in round $M+1$ p will send a "failure discovered" message that will be received by all correct processes. Since $n > 2t$, at least $t+1$ processes are correct and will send a "no decision" message in round $M+2$ (this is the only place where we need the assumption $n > 2t$). The sender, being correct, will receive all of these and will therefore have the required number of "no decision" messages to broadcast $\langle S, v\rangle$ in round $M+3$. All correct processes will receive that message and, because we are now using the decision rule of $\mathbf{B}_2$, they will all decide v, as wanted.

The proof of Case 2 is similar to the corresponding case of the previous two protocols except for the step proving that if

(i) the sender is correct and has initial value v, or

(ii) some correct process p decides v in round $M+2$,

then any correct process that decides in round $M+t+3$ must also decide v. To show this now, assume, by way of contradiction, that (i) or (ii) holds, yet some correct process q decides $v' \neq v$ in round $M+t+3$. Thus, q must have received a "failure discovered" message in round $M+1$. Since q is correct, it will send a "no decision" message in round $M+2$ that will be received by all correct processes. Thus, the sender or p (depending on whether (i) or (ii) holds) will broadcast $\langle tag, v\rangle$ in round $M+3$ and this message will be received by q (because the sender of the message as well as q are correct). Given the decision rule for $\mathbf{B}_4$ (the same as for $\mathbf{B}_2$), the only way that q could decide v' in round $M+t+3$, is if it also received $\langle S, v'\rangle$ or $\langle R, u\rangle$, for some $u \neq v$ (possibly $u = v'$). By Claim 1, such a message must have been sent in round $M+3$. We shall show that this leads to a contradiction. If a receiver sent $\langle R, u\rangle$, for some $u \neq v$, in round $M+3$ it must have decided $u \neq v$ in round M under $\mathbf{D}$. Given that (i) or (ii) holds, this contradicts (*). (If case (i) holds, recall that the sender cannot discover a failure, by hypothesis of Case 2.) If the sender sent $\langle S, v'\rangle$ in round $M+3$ then clearly (i) cannot hold. The fact that the sender broadcast such a message means that in round $M+2$ it received a "no decision" message from at least $t+1$ processes. One of these must have been correct. Thus all correct processes, in particular p, will receive such a "no decision" message in round $M+2$ which will prevent them from deciding in that round, contrary to (ii). Thus if (i) or (ii) holds no correct process q can decide $v' \neq v$ in round $M+t+3$. ▮

We interject the proof of Theorem 4, because it is closely related to the proof of part (a) of Theorem 2.

Proof of Theorem 4: The WBA protocol $\mathbf{B}$ that extends the given UFD protocol $\mathbf{D}$ is, in fact, a simplification of protocol $\mathbf{B}_1$. All that is required is to prevent the sender from sending its initial value in the relay protocol if it discovers a failure under $\mathbf{D}$. That is, in the description of what happens in round $M+2$ of $\mathbf{B}_1$, we now omit the statement "If p is the sender and has initial value v, it broadcasts $\langle S, v\rangle$. (Since all the messages that will be sent have a tag of R, we can thus omit the tag.) The decision rule is the same as that for $\mathbf{B}_1$. The resulting protocol does not necessarily satisfy Validity, because a correct sender that discovered a failure is prevented from sending its initial value. This may result in processes deciding on some other value, even though the sender is correct. However, this can only happen if some failure occurred — otherwise the sender would not have discovered one — and therefore in WBA there is no obligation to fulfill Validity in this case. The details of the proof that this in fact achieves WBA are left as an exercise.▮

Proof of Theorem 3:

(a) Let **D** be a nondecisive FD protocol that halts in M rounds. Suppose, by way of contradiction, that **B** is a BA protocol extending **D** in which processes decide by the end of round M in failure-free runs. Since **D** is nondecisive there exists a run r of **D** in which Agreement or Validity is violated. Let r_* be the run of **B** extending r in which no process fails after round M. In case Agreement is violated in r, let p and p' be correct processes that decide v and v', respectively, in r, for some $v \neq v'$. Since **D** is a (pure) FD protocol, the fact that p (resp. p') decides v (resp. v') in r, implies that p (resp. p') cannot distinguish r from the failure-free run where the sender's value is v (resp. v'). Let r_*^v and $r_*^{v'}$ be the failure-free runs of **B** which extend the failure-free runs of **D** with initial values v and v', respectively. At the end of round M, p cannot distinguish r_* from r_*^v, while p' cannot distinguish r_* from $r_*^{v'}$ Since, by assumption, processes decide in round M in failure-free runs of **B**, p must decide v and p' must decide v' in r_*. Since both p and p' are correct in r, they are also correct in r_* and thus r_* violates Agreement, which contradicts the assumption that **B** is a BA protocol. In case Validity is violated in r, a similar argument shows that if a correct process p decides v at the end of round M and its decision differs from the correct sender's initial value v', we get a contradiction to Validity in the run r_* of **B**. Thus, it is not the case that all processes can decide by the end of round M in all failure-free runs of a BA protocol extending **D**.

(b) Let **D** be a weakly sender-dependent FD protocol that halts in M rounds, and **B** be a BA protocol extending **D**, where both **B** and **D** tolerate sending omission failures. Suppose, by way of contradiction, that all processes halt by the end of round $M+1$ in the failure-free runs of **B**. By hypothesis, there are no messages beyond those used in **D** in the failure-free runs of **B**. From this we obtain the following useful fact:

(†) If at the end of round M a process p cannot distinguish a run r from a failure-free run and p does not receive any message in some round j, $M < j < i$, then p does not send a message in round i of r.

Let r and r' be the runs of **D** satisfying SD1–SD3 guaranteed to exist by the definition of weakly sender-dependent. Pick a receiver p as follows: If there are fewer than t faulty processes in r, let p be any correct receiver in r'. If there are t faulty processes in r, let p be any one of them which is not faulty in r'. (Such a receiver must exist because in r' the sender is faulty and so at most $t-1$ receivers can be faulty. Since the sender is not faulty in r, if there are t faulty processes in r, they must all be receivers, so there is at least one faulty receiver in r which is correct in r'.) Now consider runs r_* and r'_* of **B** defined as follows:

r_* is the extension of r in which the only deviation from correct behaviour after round M is that all processes that are faulty in r as well as p (whether or not it is faulty in r), omit to send any messages after round M. (By the choice of p, the number of faulty processes in r_* does not exceed t.)

r'_* is the extension of r' in which the only deviations from correct behaviour after round M are: (i) the sender omits to send a message to p in round $M+1$, and (ii) all processes that are faulty in r other than p omit to send all messages after round M. (Again, the choice of p ensures that the number of faulty processes in r'_* does not exceed t.)

Since the sender is correct and has initial value v in r_*, all correct processes must decide v in that run. We shall show that p halts and decides v' in round $M+1$ in r'_*. To see this, first observe that no process sends a message to p in round $M+1$. (This is so by definition of r'_* for the sender and the receivers that are faulty in r. For the remaining receivers, SD1 states that none of them discovered a failure in r. Thus, these processes cannot distinguish r from some failure-free run, and by SD3, they cannot distinguish r' from some failure-free run. Hence, by (†), these processes do not send any message at all in round $M+1$ of r'_*.) By SD2, p cannot distinguish r' from some failure-free run of **D** and, therefore, at the end of round M it cannot distinguish r'_* from some failure-free run of **B**. Since no process sends a message to p in round $M+1$ of r'_*, at the end of that round, p cannot distinguish r'_* from some failure-free run of **B** (because in such runs there are no messages after round M). Therefore, p must decide and halt by the end of round $M+1$ in r'_*. It remains to show that p actually decides v'. By SD1, some receiver q, which is correct in r, decides $v' \neq v$. By SD3, q decides v' in r' as well. Furthermore, by SD2, q is correct in r' and no correct process discovers a failure. Therefore, Agreement must hold in r'. This means that all correct processes in r', and in particular p, decide v'. Hence, the failure-free run which p cannot distinguish from r'_* is that in which the sender's initial value is v'. This implies that p decides v' at the end of round $M+1$ of r'_*, as wanted.

Next we shall show that all processes except p and those that are faulty in r cannot distinguish r_* and r'_* in any round. This is obvious, by SD3, for rounds 1 through M. By SD3, (†), and the definition of r_* and r'_*, only the sender sends messages in round $M+1$ in r_* and r'_*. Indeed, with the exception of the message it sends to p, the sender sends the same messages in round $M+1$ in both runs. Thus, all receivers other than p and those that are faulty in r cannot distinguish r_* and r'_* in round $M+1$. As we have seen, p halts at the end of round $M+1$ in r'_* and so does not send messages after that round. Thus, recalling that, by definition, p does not send any messages after round M in r_* and that the processes that are faulty in r do not send any messages after round M in r_* and r'_*, the only processes that send messages after round $M+1$ in the two runs are those that cannot distinguish them at the end of that round. A straightforward induction now shows that these processes will continue being unable to distinguish the two runs in all subsequent rounds. We conclude that all processes except p and those that are faulty in r cannot distinguish between r_* and r'_* in any round, as wanted.

Let q be a process other than p which is correct in r (such a process exists, since we are assuming $n \geq t+2$). We have shown that all correct processes decide v in r_*. Since, as we just saw, q cannot distinguish r_* from r'_*, it must decide v in r'_* as well. And since it is correct in r and is different from p, q is correct in r'_*. On the other hand, we have also shown that p, which is correct in r'_*, decides v' in that run. Thus r'_* violates Agreement, which contradicts the assumption that **B** is a BA protocol. Therefore, it is not possible that all processes halt by round $M+1$ in failure-free runs of **B**, as wanted.

(c) Let **D** be a sender-dependent FD protocol that halts in M rounds, and **B** is a BA protocol extending **D**, where both **B** and **D** tolerate general omission failures. Suppose, by way of contradiction, that all processes decide by the end of round $M+1$ in the failure-free runs of **B**. The proof is analogous to that in case (b), except that the runs r_* and r'_* are defined in a slightly different way:

> r_* is the extension of r in which the only deviation from correct behaviour after round M is that p omits to receive a message from the sender in round $M+1$.
>
> r'_* is the extension of r' in which the only deviation from correct behaviour after round M is that the sender omits to send a message to p in round M_1.

Note that, unlike the previous case, p and the processes that are faulty in r are not prevented from sending messages after round M in r_* and r'_*. However, since now *all* processes are unable to distinguish r and r' (and not merely the processes that are correct in r — recall that therein lies the difference between weakly sender-dependent and sender-dependent protocols), all processes will continue being unable to distinguish r_* and r'_* in all rounds after round M. The rest of the proof proceeds as in the previous case.

(d) Now suppose that **D** is a strongly sender-dependent FD protocol that halts in M rounds, **B** is a BA protocol extending **D**, so that processes decide by round $M+2$ in the failure-free runs of **B**, and that both protocols tolerate general omission failures. We have to show that if $n \geq 4$ then $n > 2t$. Since **D** is *strongly* sender-dependent, $t > 1$.

Suppose, by way of contradiction, that $n \leq 2t$, and let r and r' be the runs satisfying SD1$'$, SD2, and SD3$'$ guaranteed to exist by the definition of strongly sender-dependent. Let p be a correct process that decides v' in r (such a process exists by SD1). Partition the set of $n-2$ receivers other than p into two sets X and Y, of size $\lfloor (n-2)/2 \rfloor$ and $\lceil (n-2)/2 \rceil$, respectively, so that all faulty processes in r to a maximum of $|Y|$ are in Y and the remaining, if any, are in X. Since $n \geq 4$, both X and Y are nonempty, and since we are assuming that $n \leq 2t$, each has size at most $t-1$. Now consider three runs of **B**, defined as follows:

> r_* is an extension of r in which the faulty processes are all the processes in Y, p and, of course, those processes in X that were already faulty in r (if any). The only deviations from correct behaviour in r_* after round M are that p and the processes in Y omit to receive any message from the sender in round $M+1$, and that p omits to receive any message from the sender or the processes in X in round $M+2$. Given the way we partitioned the processes into X and Y and recalling that each of these has size at most $t-1$, the number of faulty processes in r_* does not exceed t.
>
> r'_* is an extension of r' in which the only faulty processes are the sender (which is the only faulty process in r') and the processes in X. The only deviations from correct behaviour after round M are that the sender omits to send messages to p and to the processes in Y in round $M+1$, and that the the sender and the processes in X omit to send messages to p in round $M+2$. Since $|X| \leq t-1$, the number of faulty processes in r'_* does not exceed t.
>
> r''_* is the extension of r' in which the only faulty processes are the sender and p and in which the only deviations from correct behaviour are that the sender omits to send messages to p and to the processes in Y in round $M+1$, and that p omits to receive any message from the sender or the processes in X in round $M+2$. Since $t > 1$, the number of faulty processes in r''_* does not exceed t.

It is not difficult to see that no process can distinguish between any two of r_*, r'_* and r''_* in any round. This is immediate from SD3$'$ for rounds 1 through M; for subsequent rounds it follows by a straightforward induction and the definitions of the three runs.

Now, in r_* all correct processes must decide v, since the sender is correct and has initial value v. There is at least one process in X that is correct in r_*. (This is so because the only processes in X that are faulty in r_* are those that are already faulty in r. By SD1$'$ there are at most $t-1$ faulty receivers in r, and given the way in which we partition them into X and Y, X contains at least one correct process.) Therefore, some process in X decides v in r_*.

Next we shall show that some (correct) process in Y decides v' in r'_*. To prove this, first we show that in r'_* (as in the other two runs), p does not receive any messages in rounds $M+1$ and $M+2$. To see this note that, by SD2, only the sender can distinguish r' from a failure-free run; therefore the same is true regarding r'_* at the end of round M. Since processes do not send messages after round M in runs which they cannot distinguish from failure-free runs, the only process that can send messages in round $M+1$ is the sender. However, by definition of r'_*, the sender does not send any such messages to p or to processes in Y in that round. Hence, the only processes that can distinguish r'_* from a failure-free run at the end of round $M+1$ are the sender and the processes in X. Thus, processes in Y do not send any messages in round $M+2$ and, by construction, the sender and processes in X do not send any messages to p in round $M+2$. Therefore, p receives no messages in rounds $M+1$ and $M+2$. By choice of p, that process decides v' in r. Since **D** is a (pure) FD protocol, p does not distinguish r from the failure-free run in which the sender's initial value is v'. Then, by SD3$'$, p does not distinguish r' from that same failure-free run. Since p does not receive any messages in rounds $M+1$ and $M+2$ of r'_* (which extends r') and since there are no messages beyond those already sent in **D** in the failure-free runs of **B**, p cannot distinguish r'_* from the failure-free run of **B** in which the sender has initial value v'. But processes must decide by the end of round $M+2$ in failure-free runs of **B**, so that p decides v' at the end of round $M+2$ in r'_*. Since p and all processes in Y are correct in r'_*, by Agreement all processes in Y must decide v' in r'_*.

However, as we have seen processes cannot distinguish r''_* from r_* and r'_* in any round. Thus, the correct process in X which decides v in r_* will decide v in r''_*; and the processes in Y that decide v' in r'_* will decide v' in r''_*. But the processes in both X and Y are correct in r''_*, so that r''_* violates Agreement, contradicting that **B** is a BA protocol. Therefore it is not possible to have $n \leq 2t$ in this case. ∎

Acknowledgements

We thank Ken Birman and Robbert Van Renesse for useful discussions. We are grateful to the anonymous referees for their helpful comments and suggestions. The work of Hadzilacos was supported, in part, by a grant from the Natural Sciences and Engineering Research Council of Canada.

Bibliography

- Amdur, E.S, S.M. Weber and V. Hadzilacos. "On the Message Complexity of Binary Byzantine Agreement Under Crash Failures". To appear, *Distributed Computing*, 1992.
- Attiya, H., N.A. Lynch and N. Shavit. "Are Wait-Free Algorithms Fast?". In *Proc. of the 31st Symp. on Foundations of Computer Science*, pp. 55–64, Oct. 1990.
- Bar-Noy, A., D. Dolev, C. Dwork and H.R. Strong. "Shifting Gears: Changing Algorithms on the Fly to Expedite Byzantine Agreement". In *Proc. of the 6th ACM Symp. on Principles of Distributed Computing.* pp. 42–51, August 1987.
- Bernstein, P.A., V. Hadzilacos and N. Goodman. *Concurrency Control and Recovery in Database Systems.* Addison Wesley, 1987.
- Chandra, T.D. and S. Toueg. "Unreliable Failure Detectors for Asynchronous Systems". In *Proc. of the 10th ACM Symp. on Principles of Distributed Computing.* pp. 325–340, August 1991.
- Dolev, D. "The Byzantine Generals Strike Again". *Journal of Algorithms*, 3(1):14–30, January 1982.
- Dwork, C. "Strong Verifiable Secret Sharing". In *Proc. of the 4th Int. Workshop on Distributed Algorithms.* Lecture Notes in Computer Science, Vol. 486. Springer-Verlag, pp. 213–227, 1990.
- Eager, D.L. and K.C. Sevcik. "Achieving Robustness in Distributed Database Systems". *ACM Transactions on Database Systems*, 8(3):354–381, September 1983.
- Fischer, M.J. "The Consensus Problem in Unreliable Distributed Systems". Research Report RR-273, Department of Computer Science, Yale University, June 1983.
- Hadzilacos, V. *Issues of Fault-Tolerance in Concurrent Computations.* Ph.D. dissertation, Aiken Computation Laboratory, Harvard University, June 1984.
- Hadzilacos, V. "On the Relationship Between the Atomic Commitment and Consensus Problems". Workshop on Fault Tolerant Distributed Computing. March 17–19, 1986, Pacific Grove, CA. (Proceedings published as Lecture Notes in Computer Science Vol. 448 (B. Simons and A. Spector, eds.), Springer-Verlag, 1990.)
- Hadzilacos, V. and J.Y. Halpern. "Message-Optimal Protocols for Byzantine Agreement". In *Proc. of the 10th ACM Symp. on Principles of Distributed Computing.* pp. 309–323, August 1991.
- Halpern, J.Y. and Y. Moses. "Knowledge and Common Knowledge in a Distributed Environment". *Journal of the ACM*, 37(3):549–587, 1990. (Preliminary version in *Proc. of the 3rd ACM Symp. on Principles of Distributed Computing*, pp. 50–61, August 1984.)
- Lamport, L., R. Shostak and M. Pease. "The Byzantine Generals Problem". *ACM Transactions on Programming Languages and Systems*, 4(3):382–401, July 1982.
- Lamport, L., and M.J. Fischer. "Byzantine Generals and Transaction Commit Protocols". Technical Report Op. 62, SRI International, April 1982.

- Lamport, L. "The Weak Generals Problem". *Journal of the ACM*, 30(3):668–676, July 1983.
- Moses, Y., and O. Waarts. "Coordinate Traversal: $(t+1)$-round Byzantine Agreement in Polynomial Time". In *Proc. of the 29th Symp. on Foundations of Computer Science*, pp. 246–255, Oct. 1988.
- Neiger, G. and S. Toueg. "Automatically Increasing the Fault-Tolerance of Distributed Algorithms". *Journal of Algorithms*, 11(3):374–419, September 1990.
- Pease, M., R. Shostak and L. Lamport. "Reaching Agreement in the Presence of Faults". *Journal of the ACM*, 27(2):228–234, April 1980.
- Ricciardi, A.M., and K.P. Birman. "Using Process Groups to Implement Failure Detection in Asynchronous Environments". In *Proc. of the 10th ACM Symp. on Principles of Distributed Computing.* pp. 341–351, August 1991.
- Skeen, M.D. *Crash Recovery in a Distributed Database System.* Ph.D. dissertation, Department of Electrical Engineering and Computer Science, University of California at Berkeley, 1982.
- Srikanth, T.K. and S. Toueg. "Simulating Authenticated Broadcasts to Derive Simple Fault-Tolerant Algorithms". *Distributed Computing*, 2:80–94, 1987.
- Zwaenepoel, W. "Protocols for large data transfers over local area networks". In *Proc. of the 9th Data Communications Symposium,* pp. 22–32, Sept. 1985.