MESSAGE-OPTIMAL PROTOCOLS FOR BYZANTINE AGREEMENT

Vassos Hadzilacos
Computer Systems Research Institute
University of Toronto
10 King's College Road
Toronto, Ontario M5S 1A4
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: It is often important for the correct processes in a distributed system to reach agreement, despite the presence of some faulty processes. Byzantine agreement (BA) is a paradigm problem that attempts to isolate the key features of reaching agreement. We focus here on the number of messages required to reach BA, with particular emphasis on the number of messages required in the *failure-free* runs, since these are the ones that occur most often in practice. The number of messages required is sensitive to the types of failures considered. In earlier work, Amdur *et al.* [1990] established tight upper and lower bounds on the worst- and average-case number of messages required in failure-free runs for crash failures. We provide tight upper and lower bounds for all remaining types of failures that have been considered in the literature on the BA problem: receiving omission, sending omission and general omission failures, as well as arbitrary failures with or without message authentication. We also establish a tradeoff between number of rounds and number of messages in the failure-free runs required to reach agreement in the case of crash, sending and general omission failures.

A preliminary version of this paper appears in *Proceedings of the 10th ACM Symposium on Principles of Distributed Computing*, 1991, pp. 309–323. This version is essentiall identical to one that appears in *Mathematical Systems Theory* **26**, 1993, pp. 41–102. October 2, 1996

1. Introduction

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 choose a value other than 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 of its importance as a paradigm problem, the Byzantine Agreement problem has been exceedingly well studied, using a number of different complexity measures. We focus here on one particular complexity measure: the number of messages sent. The motivation for this complexity measure is that the number of messages used by a protocol is an important, possibly the most important, factor that determines its performance. As Gray [1988] points out, the processing of each message typically requires about 2,500 instructions. In addition, sending more messages increases the likelihood of failure.

Typically, researchers who have considered message complexity have studied the worst-case message complexity over *all* runs (cf. Dolev and Reischuk [1985], Berman *et al.* [1989], Coan and Welch [1989]). We consider here instead the complexity in the *failure-free* (hereafter abbreviated f.f.) runs. The philosophy is that we wish to develop fault-tolerant protocols, but are willing to pay the price of anticipating faults *only* in the (hopefully rare) cases when faults do, in fact, occur. Even if we focus attention on the f.f. runs, there are several ways to characterize the message complexity. An obvious one is the *worst-case* complexity: the maximum number of messages sent in a f.f. run. Another one is *average-case* complexity: the expected number of messages in the f.f. runs, given a probability density over these runs. Under the reasonable assumption that the occurrence of failures is independent of the sender's initial value, this amounts to a density on the possible initial values, since a f.f. run is determined by the sender's initial value.[1] The reason for con-

[1] This last assertion is true for deterministic protocols, which are the focus of this paper. As we show in Appendix A, our lower bounds in fact hold for probabilistic protocols as well.

sidering average-case complexity is that in some applications one decision value may be much more likely than others. An example of practical interest is the problem of atomic commitment in distributed transaction processing, which is closely related to binary BA (cf. Hadzilacos [1986]). There, the decision value "commit" is much more likely than the decision value "abort". Thus, one might be willing to decrease the number of messages sent if the decision is to be "commit" at the price of increasing the number of messages sent if the decision is to be "abort". The only other paper of which we are aware that focuses on the message complexity in the f.f. runs is that of Amdur *et al.* [1990]; we discuss the relationship of our results to theirs below. The general theme of focusing on f.f. runs (although not necessarily on message complexity in f.f. runs) has been the subject of other recent papers (cf. Attiya *et al.* [1990]).

The difficulty of reaching agreement can be quite sensitive to the types of failures that can occur. A number of failure types have been considered in the literature:

a. *Crash failures:* A faulty process stops prematurely; once it has stopped, it sends no more messages.

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

c. *Receiving omission failures:* A process may fail to receive one or more messages sent to it.

d. *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.

e. *Arbitrary failures with message authentication:* Faulty processes can act arbitrarily but processes have access to a signature scheme and faulty processes cannot forge the signatures of correct ones (see Dolev and Strong [1983] for a discussion of authentication in Byzantine agreement).

f. *Arbitrary failures (without authentication):* Faulty processes can act arbitrarily, without any restriction to their possible behaviour.

We refer to crash, receiving, sending and general omission failures collectively as *benign* failures. It is easy to see that crash failures can be viewed as a special case of sending omission failures, since a process that crashes can be viewed as a process that omits to send all messages from the point that it crashes. Thus, any protocol for Byzantine agreement that handles sending omission failures automatically handles crash failures as well. It is similarly easy to show that a BA protocol that handles general omission failures can also handle sending omission failures, a BA protocol that handles arbitrary failures with message authentication can also handle general omission failures, and a BA protocol that handles arbitrary failures without authentication certainly can handle arbitrary failures with authentication. Finally, it is easy to see that a protocol that can handle general omission failures can handle receiving failures. Receiving omission failures are incomparable to crash and sending omission failures; however, as we shall see, they are easy to deal with.

Amdur *et al.* [1990] consider message complexity in f.f. runs for the case of crash failures only. They focus on *binary* BA protocols, which means that the sender wishes to broadcast the value of a single bit. Thus, there are only two f.f. runs: the run where the sender's initial value is 0, and the run where the initial value is 1. They prove that, in this case, the *total message complexity* in the f.f. runs (that is, the sum of the number of messages sent in the two f.f. runs) is $n + t - 1$, where n is the number of processes in the system, and t is a bound on the number of faulty processes. It follows that the worst-case f.f. message complexity is at least $\lceil (n + t - 1)/2 \rceil$. Also, taking P_0 and P_1 to denote the probability that the sender's initial value in a f.f. run is 0 and 1, respectively, it follows that the average-case f.f. message complexity is at least $\min(P_0, P_1) \cdot (n + t - 1)$. Amdur *et al.* [1990] also provide protocols whose worst- and average-case f.f. message complexities match the corresponding lower bounds.

In this paper we extend the results of Amdur *et al.* to deal with all classes of failures discussed above. In addition, we extend the results to deal with multiple initial values. Our results for the case of binary agreement are summarized in the table below. In all cases, the message complexity described is tight. The results for crash failures are due to Amdur *et al.*; the lower bounds on total and worst-case message complexity for arbitrary faults were proved by Dolev and Reischuk [1985].

Failure type	**Total**	**Worst case**	**Average case**
Receiving omission	$n-1$	$\lceil (n-1)/2 \rceil$	$\min(P_0, P_1) \cdot (n-1)$
Crash Sending omission General omission	$n+t-1$	$\lceil (n+t-1)/2 \rceil$	$\min(P_0, P_1) \cdot (n+t-1)$
Arbitrary (with authentication)	$n+t-1$	$\lceil (n+2t-2)/2 \rceil$ if $n \geq 8t-2$	$\min(P_0, P_1) \cdot (n+t-1)$
Arbitrary	$\lceil n(t+1)/2 \rceil$	$\lceil n(t+1)/4 \rceil$	$\min(P_0, P_1) \cdot \lceil n(t+1)/2 \rceil$

Figure 1: Message complexity for binary BA

A few remarks about the table are now in order. The results show that receiving omission failures are as benign as could be expected. The bounds are no worse than would be the case if there were no faults at all. This is not terribly surprising. Receiving failures cause no problems for Byzantine agreement. (Agreement can be achieved with a single broadcast of the sender's initial value.) What is surprising is that as far as total message complexity goes, the results for general omission failures and arbitrary failures with authentication are no worse than for crash failures. A big leap in complexity comes in the case of arbitrary failures without authentication. In this case, if t is $\Theta(n)$, we require $\Theta(n^2)$ messages rather than $\Theta(n)$. In all cases except that of arbitrary failures with authentication, we can show that we can find a family of protocols with optimal total complexity, where we can trade off messages between the two f.f. runs one-for-one; i.e., we can reduce the number of messages in one run by increasing the number of messages in the other run by the same amount. Hence, to obtain a worst-case optimal protocol we divide equally the number of messages between the two f.f. runs. To obtain an average-case optimal protocol we eliminate all the messages from the f.f. run that corresponds to the more likely initial value and put all the messages required by the lower bound on the total f.f. message complexity in the other f.f. run.

The case of arbitrary failures with authentication is somewhat anomalous. Here we cannot trade off messages between the two f.f. runs in general. We can show that there is a protocol which uses 0 messages in one failure-free run and $n+t-1$ in the other (from which the average-case complexity result stated in the table follows). However, any attempt to balance messages between the two f.f. runs results in additional total message complexity. We provide a protocol that uses $\lceil (n+2t-2)/2 \rceil$ messages in each of the two f.f. runs. Moreover, for most (but not all) combinations of n and t, we can show that this bound is optimal. In particular, if $n \geq 8t-2$, we can show that $\lceil (n+2t-2)/2 \rceil$ is a tight bound. However, there are values of n and t for which we can beat this bound. For example, if $n=5$ and $t=2$, there is a protocol which requires only 3 messages in each f.f. run. We remark that our upper bound shows that, for this type of failure, the worst-case message complexity in the f.f. runs is strictly better than in the worst-case message complexity over all runs, which was shown to be $\Omega(n+t^2)$ by Dolev and Reischuk [1985].

It is worth noting that, except in the case of arbitrary failures with authentication, our results provide tight bounds for bit complexity as well as message complexity. Clearly our lower bounds on message complexity are also lower bounds on bit complexity. In all our protocols for *binary* BA (except those involving authentication), all the messages sent are actually only one bit long, so the upper bound follows as well.[2]

We also consider tradeoffs between message complexity and number of rounds. It is easy to see that for general omission failures (and, *a fortiori*, for crash failures and sending omission failures) there is a protocol where all processes can decide in one round in the f.f. runs with worst-case f.f. complexity of $n-1$: the sender simply sends its initial value to all the processes, which decide when they receive the value (and invoke, say, a standard BA protocol if they do not receive a value in the first round). As shown in Amdur *et al.* [1990], we can achieve the optimal worst-case f.f. complexity of $\lceil (n+t-1)/2 \rceil$ messages in the case of crash failures with a protocol that takes only two rounds in the f.f. runs. However, once we move to sending omission failures, we can show that achieving optimal message complexity requires at least $\lceil (n-t+1)/2 \rceil$ rounds in the f.f. runs. We present a protocol which achieves optimal message complexity using $\lceil (n-t+3)/2 \rceil$ rounds. These and other related results we prove show how we can trade off time and message complexity in the f.f. runs.

We conclude this introduction with a few words about our model and notation. We assume 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, communication is reliable and of bounded delay, and system computations proceed in successive synchronous *rounds*. In each round each process takes exactly one step consisting of the following actions, performed in the specified order:

(a) Send messages to a subset of the processes.

(b) Receive all messages sent to it by other processes in *that* round.

(c) Possibly choose a decision value (if one hasn't already been chosen).

---

[2] This observation was already made by Amdur *et al.* [1990] in the case of crash failures.

Note how the assumption of reliable and bounded-delay communication is embodied in (b). Thus, in this model, a run is simply a sequence of such rounds. The *view of a process p in run r* is the sequence of the sets of messages p receives in each round of r. In the case of the sender, its view also includes the initial value. In other words, the view of p in r encapsulates all the information that p has about r. Sometimes we shall say that *p cannot distinguish runs r and r'* to indicate that p's views in these two runs are identical. We also say that p cannot distinguish the two runs *up to round i* to indicate that p has the same view in the first i rounds of the two runs (although the views may differ later).

Throughout the paper, we use the following notation:

- V denotes the set of initial values. In the case of binary agreement, we take $V = \{0, 1\}$.
- H_v ($v \in V$) denotes the f.f. run where the sender has initial value v.
- R_v ($v \in V$) and B denote sets of receivers. In the case of binary agreement, R_v (for $v \in \{0, 1\}$) will be the set of receivers which receive exactly one message in H_v and no message in $H_{\overline{v}}$, while B is the set of receivers which receive at least two messages in the two f.f. runs (one in each, or zero in one and two in the other), together with the sender, if the sender receives at least one message in one of the f.f. runs.

The remainder of the paper is organized as follows: In Section 2, we introduce a new problem called the *failure discovery problem*, which is simpler than BA and show how to transform a solution to this problem into a solution for full-fledged BA without introducing any additional messages in the f.f. runs. This makes it easier to present our upper bounds, since it is enough to describe protocols that solve the simpler problem of failure discovery. In Sections 3 through 7, we consider the problem of binary BA for the various failure types. In Section 8, we extend our results to deal with multiple initial values. We conclude in Section 9 with a discussion of other open problems. We defer to Appendix A the proof that our results hold for randomized protocols as well as deterministic protocols.

2. From failure discovery to Byzantine agreement

Informally, we can view our protocols as a combination of a protocol that is message-efficient and correct provided there are no failures, together with a standard BA protocol which is invoked if a failure is discovered. The notion of "correct provided there are no failures" is formalized and studied in detail in a companion paper (Hadzilacos and Halpern [1991]) in terms of the *failure discovery (FD)* problem. Techniques are given there for combining failure discovery protocols with standard BA protocols to produce protocols which are message-efficient in the f.f. runs and are correct even in the presence of failures. We now review the main ideas and results of Hadzilacos and Halpern [1991], since we shall need them here. (In fact, these results were motivated by the problems of this paper.) FD is similar to BA except that the three conditions — Termination, Agreement and Validity — need only hold if *no correct process has discovered a failure.* Formally, we say that a process p *discovers a failure* in round i of run r if p can distinguish r from both f.f. runs in round i.[3]

[3] It is interesting to note that by this definition, a process discovers a failure iff it *knows* that some

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.

We say that a protocol **A** is an *extension* of a protocol **B**, or that **A** *extends* **B**, if every run of **A** has a prefix in which precisely the same messages are sent and received as in a run of **B**, and for every run r of **B**, there is a run of **A** that has a prefix in which precisely the same messages are sent and received as in r. The two theorems we give in this section state that we can extend a FD protocol to a BA protocol with no message overhead in the f.f. runs. The first theorem applies to all types of failures; the second applies only to benign failures, but results in a more round-efficient extension. To understand these theorems it is important to explain the difference between a process deciding and a process halting. When a process halts, by definition, it cannot do anything in the future. Thus, the Termination property implies that, in BA protocols, a correct process must decide before it halts. However, it is *not* the case that a correct process can halt as soon as it decides. Its participation in subsequent rounds may be required to ensure that other correct processes decide consistently.

For our first theorem we need the following technical definition: We call a 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. (For example, cf. Srikanth and Toueg [1987] for arbitrary failures with no authentication, Dolev and Strong [1983] for arbitrary failures with authentication, Hadzilacos [1984] for benign failures.)

Theorem 2.1: *(Hadzilacos and Halpern [1991]) Let* **D** *be a FD protocol that tolerates some failure type in whose f.f. runs processes halt within M rounds, and let* **A** *be a special BA protocol that tolerates the same failure type in whose runs (not only the f.f. 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 f.f. run of* **B** *there are no messages sent other than those sent by* **D**, *and all processes halt by round $M + N$.*

The transformation of a FD to a BA protocol in Theorem 2.1, although message-preserving in the f.f. runs, is far from being round-preserving. Since N is the number of rounds required in the worst-case run of **A** (and not only in f.f. runs), it is known that $N \geq t+1$ for all failure types (cf. Fischer and Lynch [1982], Dolev and Strong [1983], Dwork and Moses [1986], Lynch [1989]). While we conjecture that an overhead of $t+1$ rounds is necessary in general, we can do much better in the case of benign failures, provided that the FD protocol we are given has a stronger property than Weak Agreement. To motivate

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]).

this property consider *Uniform Byzantine Agreement* (UBA), a version of BA first studied by Neiger and Toueg [1990]. In this problem, a protocol must satisfy Termination, Validity and the following strengthening of Agreement:

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

This property is of considerable interest. For example, Hadzilacos [1986] pointed to it as a key difference between BA and Atomic Commitment.

Uniform Failure Discovery (UFD) is to UBA what FD is to BA. More precisely, a UFD protocol must satisfy Weak Termination, Weak Validity and the following strengthening of Weak Agreement:

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

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. The next theorem says that in the case of benign failures we can extend a Uniform FD protocol to a full-fledged BA protocol with no message overhead and a round overhead of at most two in the f.f. runs. The precise amount of round overhead depends on the type of failures we are considering, whether we count the number of rounds for processes to decide or halt, and whether the UFD protocol satisfies one of two properties. A protocol is *safe* if there is no violation of Validity or Uniform Agreement in any of its runs. (This does not mean that it is a UBA protocol, as Termination may be violated.) We say that *the sender cannot discover a failure* in a protocol if the sender does not discover a failure in any of its runs.

Theorem 2.2: *(Hadzilacos and Halpern [1991]) Let* $\mathbf{D}$ *be a UFD protocol that tolerates some type of benign failure in whose f.f. runs processes halt 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 f.f. 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 f.f. 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 f.f. runs of* $\mathbf{B}_2$, *all processes decide and halt in* $M+1$ *rounds; if the sender cannot discover a failure in* $\mathbf{D}$ *then* $\mathbf{B}_2$ *is a BA protocol that tolerates the same type of failures as* $\mathbf{D}$.

(c) *in the f.f. 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 f.f. 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* $\mathbf{D}$*).*

(Part (b) is a simplification of what appears in Hadzilacos and Halpern [1991] but is all we need for this paper.)

The significance of these theorems lies in that they reduce the task of devising a BA protocol which is message efficient in the f.f. runs to the (conceptually simpler) task of devising a FD protocol which is efficient in the same way. Thus, in subsequent sections, when presenting upper bounds we shall confine ourselves to describing a FD protocol knowing that it can be extended to a full-fledged BA protocol without affecting the message complexity in the f.f. runs. This will simplify the exposition considerably. Furthermore, in the case of benign failures we shall also consider the number of rounds required for message-optimal protocols. Since our message optimal-protocols satisfy the Uniformity property we shall be able to show that the BA protocols that result when we apply the constructions of Theorem 2.2 are as round-efficient in the f.f. runs as any message-optimal BA protocol could be.

3. Receiving omission failures

It is easy to see that $n-1$ is a lower bound on the total f.f. message complexity: If there were a protocol with total f.f. message complexity less than $n-1$, some receiver would not receive a message in both f.f. runs. But then such a receiver would choose the same decision value in both f.f. runs, in violation of the Validity condition. The lower bounds on the worst- and average-case message complexities follow immediately.

Now consider the following family of protocols. Partition the set of receivers into two sets R_0 and R_1 (we get a different protocol for each such partition). The sender sends its initial value $v \in \{0,1\}$ to all receivers in R_v and sends no message to the receivers in $R_{\overline{v}}$. A receiver in R_u, $u \in \{0,1\}$, decides u if it receives message u in round 1; it decides $\overline{u}$ if it receives no message in round 1. It is easy to see that this is a correct protocol, if only receiving omission failures can occur. In addition, each protocol in the family takes only one round, which is clearly optimal. By taking $|R_0| = \lceil (n-1)/2 \rceil$ and $|R_1| = \lfloor (n-1)/2 \rfloor$, we get a worst-case optimal protocol. By taking $R_v = \emptyset$ and $|R_{\overline{v}}| = n-1$, where v is the more probable initial value, we get an average-case optimal protocol. Note that each message in the protocol is one bit long, so we get tight bounds on both the number of bits and the number of messages required.

4. Crash failures

Amdur *et al.* [1990] prove a lower bound of $n+t-1$ messages on the total f.f. message complexity for crash failures and provide protocols that achieve these bounds. We reprove and slightly generalize some of their results here, to prepare the way for the material in later sections, as well as proving new results on the number of rounds required to achieve optimal message complexity. We begin with the lower bound. Our techniques for the lower bound lead to a considerably simpler proof than that of Amdur *et al.* [1990].

We first need some definitions and a lemma, which will also be useful later. A *message chain* in a run r is a sequence of processes $p_0, p_1, \ldots, p_k$ such that there exist messages $\mu_1, \mu_2, \ldots, \mu_k$ and round numbers $\ell_1 < \ell_2 < \ldots < \ell_k$ such that for all $1 \leq i \leq k$, p_{i-1}

sends μ_i to p_i in round ℓ_i in r. We say that p *affects* q *after round* m *in* r if there is such a chain with $p_0 = p$, $p_k = q$, and $\ell_1 > m$.

Lemma 4.1: *Consider any t-resilient FD protocol. Let $p \in R_v$ for some $v \in \{0,1\}$ and suppose that p receives its message from some process p' in round m of H_v. Let*

- *A_1 be the set of processes affected by p after round m in H_v;*
- *A_2 be the set of processes that receive a message from p after round m in $H_{\overline{v}}$;*
- *A_3 be the set of processes affected by processes in A_2 after round m in H_v; and*
- *A_4 be the set of processes affected by p' after round m in H_v.*

Then:

(1) If the protocol tolerates general omission failures then $|A_1 \cup A_2| \geq t$.

(2) If the protocol tolerates sending omission failures then $|A_1 \cup A_2 \cup A_3| \geq t$.

(3) If the protocol tolerates crash failures then $|A_1 \cup A_2 \cup A_3 \cup A_4| \geq t$.

Proof: (1) Suppose $|A_1 \cup A_2| < t$. The plan is to identify a run r of the protocol such that:

(a) the only faulty processes in r are p' and those in $A_1 \cup A_2$ (so that there are at most t faulty processes),

(b) p cannot distinguish r from $H_{\overline{v}}$,

(c) all the correct processes other than p cannot distinguish r from H_v.

Since $n \geq t + 2$ there is some correct process other than p; thus (b) and (c) above show that r violates the Weak Agreement condition, giving us a contradiction.

In run r processes behave as follows: All behave just as in H_v through round m, with the exception of p', which omits to send its message to p in that round. At the beginning of round $m+1$, all processes in A_1 crash. Also, after round m, all processes in $A_2 \setminus A_1$ omit to receive any messages sent to them by p in $H_{\overline{v}}$ (that are not also sent to them in H_v). Informally, at the end of round m of r, process p "thinks" that r is $H_{\overline{v}}$ (because it receives no message in either run) while all other processes "think" that r is H_v (because they get exactly the same messages in the two runs). The only processes that are capable of detecting this inconsistency after round m are (a) those in A_1, because in H_v they expect a message that they will not receive; and (b) those in A_2, because they will be sent messages from p that they do not expect in H_v. However, the processes in A_1 will crash in r and so they will not detect the inconsistency; while the processes in A_2 will not receive those messages sent to them by p that can cause them to "think" that r is not H_v, so they will continue to behave as in H_v. Indeed, a straightforward induction on the round number i shows that:

- the processes which are faulty in r up to round i are a subset of $\{p'\} \cup A_1 \cup A_2$;
- p's view of r through round i is identical to its view of $H_{\overline{v}}$ through that round; and
- the view that every process other than p and the faulty ones has of run r through round i is the same as its view of H_v through that round.

It now immediately follows that r satisfies the properties (a), (b), and (c) above, giving us the desired contradiction.

(2) In the proof of part (1), all faulty processes in r except those in A_2 actually commit sending omission failures. The reason why we could not make the processes in A_2 crash after round m in r is that a process $q \in A_2$ may be required to send messages to other processes after round m in H_v. Thus, processes in A_3 (i.e., processes affected by processes in A_2 after round m in H_v) will "realize" that r is not H_v, foiling our argument. We can solve this problem by making processes in *both* A_2 and A_3 crash at the beginning of round $m+1$. More concretely, we can identify a run r as in the proof of part (1) where the faulty processes are p' (which commits a sending omission failure) and the processes in $A_1 \cup A_2 \cup A_3$ (which all commit crash failures). As before, we will have a violation of Weak Agreement in r unless the number of faulty processes exceeds t, which means that $|A_1 \cup A_2 \cup A_3| \geq t$, as wanted.

(3) In the proof of part (2), all faulty processes in r except p' actually crash. The reason why we can't also make p' crash in round m of H_v is similar to the reason why we couldn't make the processes in A_2 crash in the argument for part (1). The problem is that p' may send messages after round m in H_v and thus, processes in A_4 (i.e., the processes affected by p' after round m in H_v) will discover that p' has failed and will no longer "confuse" r with H_v. We can apply an analogous remedy to the problem: By making all processes in A_4 crash at the beginning of round $m+1$ in r we can assume that p' crashes in round m as well. Therefore, we can identify a run r as in the proof of part (2) where the faulty processes are p' and the processes in $A_1 \cup A_2 \cup A_3 \cup A_4$ and, moreover, all faulty processes actually suffer crash failures in r. Again, we will have a violation of Weak Agreement in r unless the number of faulty processes exceeds t, which means that $|A_1 \cup A_2 \cup A_3 \cup A_4| \geq t$, as wanted. ∎

Theorem 4.2: (Amdur *et al.* [1990]) *The total f.f. message complexity of any FD protocol for crash failures is at least $n+t-1$. Moreover, if the total f.f. message complexity is exactly $n+t-1$, then we must have $|B| = t$ (recall that B is the set of processes that receive at least two messages in the f.f. runs, together with the sender, if the sender receives at least one message in one of the f.f. runs).*

Proof: Fix a protocol **D** that achieves FD in the case of crash failures. Our goal is to show that $|B| \geq t$. Observe that this suffices, for if the sender is not in B, it follows that, in the f.f. runs of **D**, each process in B receives at least two messages, and the $n - |B| - 1$ processes in $R_0 \cup R_1$ receive one message each, for a total of at least $2|B| + n - |B| - 1 = n + |B| - 1 \geq n + t - 1$ messages. And if the sender is in B then each of the remaining $|B| - 1$ processes in B receives at least two messages, while each of the other $n - |B| + 1$ processes receives at least one message in the f.f. runs of **D**, again for a total of at least $2(|B|-1) + (n - |B| + 1) = n + |B| - 1 \geq n + t - 1$ messages. These calculations also show that if the total message complexity is exactly $n+t-1$, then we must have $|B| = t$.

If $R_0 \cup R_1 = \emptyset$ then $|B| \geq n-1$, so we are certainly done. Otherwise, let p be a process in $R_0 \cup R_1$ so that no process in $R_0 \cup R_1$ receives its message in either of H_0 or H_1 later than p does. Without loss of generality, $p \in R_1$, so it receives a message in H_1, say at round m. Using notation as in Lemma 4.1, observe that all processes in $A_1 \cup A_2 \cup A_3 \cup A_4$ are in B, since they receive a message after p in either H_0 or H_0 and thus, by choice of p, they cannot be in R_0 or R_1. Since $B \supseteq A_1 \cup A_2 \cup A_3 \cup A_4$, and $|A_1 \cup A_2 \cup A_3 \cup A_4| \geq t$ by Lemma 4.1(3), we are done. ∎

For the upper bound, we briefly review the two families of protocols presented in Amdur *et al.* [1990], which show that not only can we obtain the optimal complexity of $n+t-1$, but that we can split the $n+t-1$ messages arbitrarily between the two f.f. runs. We then consider the number of rounds used by these protocols.

For the first protocol, called CF1 (the CF stands for "Crash Failures"), we partition the set of receivers into three sets, R_0, R_1, and B, where B contains exactly t receivers (which can be thought of as witnesses to the correctness of the sender). The protocol is very simple: If the sender's initial value is v, then in the first round, it sends v to the processes in R_v, while in the second round, it sends v to the processes in B. Note that the processes in B receive a message in both H_0 and H_1, while the processes in R_v receive a message only in H_v.

At the end of round 2, the processes decide as follows:

- the sender decides on its initial value
- a process in R_v decides v if it received a message v; otherwise it decides $\overline{v}$
- a process in B decides v if it received a message v; if it did not receive a message, it discovers a failure.

It is easy to verify that this is a Uniform FD protocol. For example, if processes p and q in R_v decide different values, then it must be the case that one of them, say p, did not receive a message in round 1. Since we are dealing with crash failures here, this means that the sender must have crashed, from which it follows that no process in B will receive a round 2 message. Since there are t processes in B, if the sender is faulty, one of them must be correct. Thus, a correct process will discover a failure. The reader can consult Amdur et al. [1990] for further details of the correctness proof (cf. the proof of correctness of protocol GOF1 below).

Counting the number of messages received we get that the total f.f. message complexity of any instance of CF1 is

$$|R_0| + |R_1| + 2|B| = (n-1-|B|) + 2|B| = n-1-t+2t = n+t-1.$$

Moreover, with this family of protocols we can use anywhere between t and $n-t$ messages in one f.f. run and the balance (to the total of $n+t-1$) in the other, depending on our choice of $|R_0|$ and $|R_1|$. Thus we have:

Theorem 4.3: *Every instance of protocol family CF1 solves the Uniform Failure Discovery problem in the case of crash failures, using a total of $n+t-1$ messages in the f.f. runs and taking two rounds. By taking $|R_0| = \lceil (n-t-1)/2 \rceil$ and $|R_1| = \lfloor (n-t-1)/2 \rfloor$, we get a protocol with optimal worst-case f.f. message complexity of $\lceil (n+t-1)/2 \rceil$.*

Note that using CF1, we can obtain protocols with optimal total message complexity, and somewhere between t and $n-1$ messages in each of the f.f. runs. In order to obtain a protocol that has optimal average-case complexity, we want a protocol that has total message complexity $n+t-1$, but puts all the $n+t-1$ messages in the run of lower probability. We do this using the protocol family $\mathrm{CF2}_v$, $v \in \{0,1\}$. We partition the set of processes into three sets: R_v, B^1, and B^2_v, with $|B^1 \cup B^2_v| = t$. Just as before, the processes in R_v receive one message in H_v and none in $H_{\overline{v}}$. The processes in B^2_v receive two messages in H_v and none in $H_{\overline{v}}$, while the processes in B^1 receive one message in each of H_0 and H_1. (We can view $R_{\overline{v}}$ and $B_{\overline{v}}$ as being empty in protocol $\mathrm{CF2}_v$.) If the sender's initial value is v, then it sends v to all the processes in B^2_v in round 1, sends v to all the processes in R_v in round 2, and v to all the processes in $B^2_v \cup B^1$ in round 3. The decision rule for $\mathrm{CF2}_v$ is a simple extension of that for CF1. At the end of round 3, the processes decide as follows:

- the sender decides on its initial value
- a process in R_v decides v if it received a message v; otherwise it decides $\overline{v}$
- a process in B^1 decides v if it received a message v; otherwise it discovers a failure
- a process in B^2_v decides v if it received two messages saying v, decides $\overline{v}$ if it received no messages, and discovers a failure if it received only one message.

Again, it is easy to verify the correctness of $\mathrm{CF2}_v$. The key point is that, because we are dealing with crash failures, if a process in R_v receives a message in round 2, then it knows that all the processes in B^2_v received a round 1 message, while if a process in $B^2_v \cup B^1$ receives a round 3 message, then it knows that all the processes in $R_v \cup B^2_v$ must have received a message in round 1 or round 2. More specifically, to prove that the protocol satisfies Weak Uniform Agreement suppose that process p decides v and process q decides $\overline{v}$. Thus, it must have been the case that p received a message (in particular, a v) and q did not. Therefore the sender must be faulty. It is not possible that q is in B^1 because in that case, to decide $\overline{v}$, it would have to receive $\overline{v}$ in round 3 and then p couldn't have received a v (since only crash failures can occur). Also, it is not possible that q is in B^2_v because then q would have to receive no message in either round 1 or 3; if q does not receive a message in round 1 then no process can receive any message in round 2 or 3; but p needs to receive v in one of these rounds (depending on which set it is in) to decide v. The only remaining possibility is that q is in R_v. In this case, since we are dealing with crash failures, all processes in B^2_v receive a message in round 1 and no process in $B^1 \cup B^2_v$ receive a message from the sender in round 3. Since there are t processes in $B^1 \cup B^2_v$ and the sender is faulty, one of them must be correct and it will discover a failure.

The total number of messages in the f.f. runs of $\mathrm{CF2}_v$ is

$$(|B^2_v| + |R_v| + |B^1 \cup B^2_v|) + |B^1| = |R_v| + 2 \cdot |B^1 \cup B^2_v| = n - t - 1 + 2t = n + t - 1.$$

Moreover, with this family of protocols we can use anywhere between 0 and t messages in one f.f. run and the balance (to the total of $n+t-1$) in the other, depending on our choice of $|B^1|$. Summarizing these observations, we have the following result:

Theorem 4.4: *Every instance of protocol family $\mathrm{CF2}_v$, $v = 0,1$, solves the Uniform Failure Discovery problem in the case of crash failures, using a total of $n+t-1$ messages in the f.f. runs and taking three rounds. If $P_{\overline{v}} \geq P_v$, by taking $B^1 = \emptyset$, we get a protocol with optimal average-case f.f. message complexity of $P_v \cdot (n+t-1)$.*

We now turn our attention to the number of rounds required to attain FD. Can we do better than the two rounds required by CF1 and the three rounds required by CF2_v to attain FD? The answer is yes if we are not concerned with message complexity. The protocol in which the sender sends its initial value to all receivers solves the FD problem for crash failures (in fact, even general omission failures) and takes only one round in the f.f. runs, but has a worst-case message complexity of $n-1$. We can also get a straightforward FD protocol where all messages are sent in H_v as follows: If the sender's initial value is $\overline{v}$, it sends no messages. If the sender's initial value is v, it sends v to all processes in round 1, and again in round 2. In round 2, a receiver decides v if it receives two v messages, and decides $\overline{v}$ if it receives none; if it receives only one, it discovers a failure. This latter protocol uses $2(n-2)$ messages in H_v and none in $H_{\overline{v}}$. However, as we now show, CF1 and CF2_v are optimal in terms of number of rounds if we do not want to sacrifice message optimality.

Theorem 4.5: *Every Failure Discovery protocol for crash failures which uses a total of $n+t-1$ messages in the f.f. runs requires at least two rounds in one of the f.f. runs. If, in addition, no messages are sent in $H_{\overline{v}}$ and either $t>1$ or $n>4$, then at least three rounds are required in H_v.*

Proof: Suppose $\mathbf{D}$ is a FD that uses a total of $n+t-1$ messages in the f.f. runs. From Theorem 4.2, we have $|B|=t$ and thus $|R_0 \cup R_1| \geq n-1-|B| = n-t-1 > 0$.

Claim 4.5.1: *No receiver in R_v can receive its message in the last round of H_v, for $v \in \{0,1\}$.*

Proof of Claim 4.5.1: Suppose, by way of contradiction, that some process p in R_v received a message in the last round of H_v. Consider a run r which is just like H_v except that the process p' that sent p its message in the last round of H_v crashes after having sent all messages except the one to p. It is easy to see that p cannot distinguish p from $H_{\overline{v}}$ (where it gets no messages), while all other processes cannot distinguish r from H_v. This is a violation of Weak Agreement. ▮ Claim 4.5.1

Since $R_0 \cup R_1$ is nonempty, it cannot be the case that both f.f. runs of $\mathbf{D}$ use only one round, for then some process in R_v would receive its message on the last round of H_v, for some $v \in \{0,1\}$.

For the remainder of the proof further assume that $\mathbf{D}$ is such that no messages are sent in $H_{\overline{v}}$. We want to show that if $t>1$ or $n>4$, $\mathbf{D}$ requires at least three rounds.

Claim 4.5.2: *If $p \in R_v$ receives a message in round 1 of H_v then p sends messages to at least t processes in H_v after round 1.*

Proof of Claim 4.5.2: Suppose, by way of contradiction that t sends messages to fewer than t processes after round 1. Consider a run r where the sender is faulty, sends its initial value v only to p and then crashes. Moreover, all the processes to which p sends a message after round 1 crash at the beginning of round 2. Thus, p cannot distinguish r from H_v, while all the other correct processes (and there must be some, since $n \geq t+2$) cannot distinguish r from $H_{\overline{v}}$. Since the total number of faulty processes in r does not exceed t, we get a violation of Weak Agreement. ▮ Claim 4.5.2

Claim 4.5.3: *If $t = 1$ and $n > 4$, or if $n > t+2$ and one of the messages received by some process in B was not sent by a process in R_v, then some process in R_v receives its message in round 2 of H_v.*

Proof of Claim 4.5.3: Suppose, by way of contradiction, that all processes in R_v receive their message in round 1 of H_v. There are $|R_v| \geq n-t-1$ messages received by the processes in R_v in round 1. In addition, by Claim 4.5.2, there are at least $|R_v|t$ messages sent by the processes in R_v in later rounds. Thus, we have identified at least $n + (|R_v| - 1)t - 1$ messages in H_v. If $t = 1$ and $n > 4$ then $|R_v| \geq n - t - 1 \geq 3$, and so there are at least $n+2t-1 > n+t-1$ messages in H_v, a contradiction. If $n > t+2$ then $|R_v| \geq n-t-1 \geq 2$, but in this case there is an additional message mentioned in the hypothesis (that received by some process in B and not sent by a process in R_v). So, in this case we have at least $(n + t - 1) + 1 > n + t - 1$ messages in H_v, again a contradiction. ∎ Claim 4.5.3

Claim 4.5.4: *If $t > 1$ and all processes in B receive both their messages from processes in R_v then H_v has at least three rounds.*

Proof of Claim 4.5.4: Assume the contrary. Let $q \in B$ and p_1, p_2 be the processes in R_v which send the two messages that q receives. Since there are only two rounds in H_v and only the sender sends messages in round 1, both of these are sent in round 2. Consider the run r in which the sender has initial value v, and p_1 and p_2 crash so that they do not send their messages to q in round 2 (but send any other messages they are supposed to send). Process q cannot distinguish r from $H_{\overline{v}}$ (since it receives no message in either), while all other correct processes (which must exist, since $n \geq t+2$) cannot distinguish r from H_v. Since only two processes are faulty in r and $t > 1$, we get a violation of Weak Agreement. ∎ Claim 4.5.4

From Claims 4.5.1, 4.5.3 and 4.5.4 we get that if $n > t+2$ and either $t > 1$ or $n > 4$, **D** uses at least three rounds in H_v. It remains to show that three rounds are needed even when $n = t+2$ and $t > 1$. (Note that since $n = t+2$, $n > 4$ implies that $t > 1$, so we need not consider that possibility separately.) By Claim 4.5.1, we get our result if some process in R_v receives its message in round 2. Thus, we can assume that all processes in R_v receive their message in round 1. We have $|R_v| \leq n - t \leq 2$. If $|R_v| = 2$, then the total number of messages sent by the sender in round 1 and by the two processes in R_v in subsequent rounds (recall Claim 4.5.2) would be at least $2 + 2t = n + t$, a contradiction. Thus, R_v consists of a single process, say p. If there is no round 3 in H_v then we have enough information to know exactly what happens in H_v in this case. Namely, the sender sends a message to p in round 1, p sends a message to each of the processes in B in round 2, and the sender sends a message to the processes in B in round 1 or 2. Without loss of generality, assume that q receives a message from the sender no later than any other process in B. Consider run r in which the sender has initial value v and crashes after sending its message to p and q but to no other process, and p crashes in round 2 after sending its message to q but to no other process in B. Thus, q cannot distinguish r from H_v, while all other processes in B (which must exist since $|B| = t > 1$) cannot distinguish r from $H_{\overline{v}}$, as they receive no message in either. Since $t > 1$ and there are only two faulty processes in r, we get a violation of Weak Agreement. Thus, there must be a third round in H_v in this case as well. ∎

It is easy to check that our assumptions that $n > 4$ or $t > 1$ in the second half of Theorem 4.5 are necessary. We leave it to the reader to construct FD protocols where no messages are sent in $H_{\overline{v}}$, $n + t - 1$ messages are sent in H_v, and only two rounds are required in the two cases not covered by the theorem — namely, $n = 4$, $t = 1$, and $n = 3$, $t = 1$.

What about BA? Since CF1 and CF2$_v$ are *Uniform* FD protocols, we may use the round-efficient transformation of Theorem 2.2 to translate them into BA protocols. If $n = t + 2$ it is straightforward to check that CF1 and CF2 are safe, so we can extend them to a BA protocol using the construction in Theorem 2.2(a). If $n > t + 2$ then CF1 and CF2 are not safe, but the sender cannot discover a failure and so we can extend them using the construction of Theorem 2.2(b). Thus, we get

Theorem 4.6: *There are BA protocols* $\mathbf{A}_{wc}$ *and* $\mathbf{A}_{ac}$ *for crash failures such that* $\mathbf{A}_{wc}$ *has worst-case f.f. message complexity of* $\lceil (n + t - 1)/2 \rceil$*, and all processes decide and halt in three rounds (if* $n = t + 2$ *they decide in two rounds) in the f.f. runs; while* $\mathbf{A}_{ac}$ *has average-case f.f. message complexity of* $\min(P_0, P_1) \cdot (n + t - 1)$*, and all processes decide and halt in four rounds (if* $n = t + 2$ *they decide in three rounds) in the f.f. runs.*

By Theorem 4.2, the worst-case and average-case f.f. message complexity of $\mathbf{A}_{wc}$ and $\mathbf{A}_{ac}$ respectively cannot be improved. By Theorem 4.5, their round complexity cannot be improved by more than one round in the f.f. runs, without sacrificing their message optimality. Indeed, as we show in the next two theorems, even the one round improvement cannot be achieved!

In order to prove this result, we need the following definition and theorem from Hadzilacos and Halpern [1991].

A FD protocol is *nondecisive* if there is some run in which Agreement or Validity is violated.

Theorem 4.7: (Hadzilacos and Halpern [1991], Theorem 3(a)) *Let* $\mathbf{D}$ *be a nondecisive FD protocol for crash failures and* $\mathbf{B}$ *be a BA protocol for crash failures that extends* $\mathbf{D}$ *so that in the f.f. runs of* $\mathbf{B}$ *there are no messages other than those sent by* $\mathbf{D}$*. If all processes halt by round* M *in the f.f. runs of* $\mathbf{D}$ *then there is some f.f. run of* $\mathbf{B}$ *in which a process decides no earlier than round* $M + 1$*.*

Theorem 4.8: *In any BA protocol for crash failures that uses a total of at most* $n + t - 1$ *messages in the f.f. runs, some process does not halt, and if* $n > t + 2$ *does not even decide, until round 3.*

Proof: We first give general conditions under which a FD protocol is nondecisive.

Claim 4.8.1: *Let* $\mathbf{D}$ *be a FD protocol for crash failures that uses a total of at most* $n + t - 1$ *messages and at most two rounds in the f.f. runs. Then in at least one f.f. run, every process in* B *gets a message in round 2. Moreover, if* $n > t + 2$*, then* $\mathbf{D}$ *is nondecisive.*

Proof of Claim 4.8.1: Since D has at most two rounds, by Claim 4.5.1 all processes in R_v receive their messages in the first round of H_v. Moreover, these messages must come from the sender (for messages sent by any other process in round 1 would be sent in both f.f. runs, and would be redundant). We next claim that if $R_v \neq \emptyset$, then all processes in B receive a message in round 2 of H_v. For otherwise, suppose $p \in R_v$ and let q be a process in B which does not receive a message in round 2 of H_v. Consider the run r in which the sender has initial value v and crashes after sending all its round 1 messages except the one to p. Then all the processes in B other than q crash in round 2. Now p cannot distinguish r from $H_{\overline{v}}$ and q cannot distinguish r from H_v. Since the number of faulty processes in r is $(|B|-1)+1=t$, we get a contradiction to Weak Agreement.

Now assume that $n > t+2$. We have $|R_0 \cup R_1| \geq n-1-|B| = n-t-1 \geq 2$. Let $p \in R_v$, $v \in \{0,1\}$. Let r be a run in which the sender's initial value is v and the sender crashes after sending all its round 1 messages except the one to p. Clearly all the processes in $R_0 \cup R_1$ other than p cannot distinguish r from H_v, and so decide v, while p cannot distinguish r from $H_{\overline{v}}$, and so decides $\overline{v}$. Since $|R_0 \cup R_1| \geq 2$, this gives us a violation of Agreement; hence **D** is nondecisive. ∎ Claim 4.8.1

We are now ready to prove the theorem. First suppose, by way of contradiction, that there is a BA protocol **B** that uses a total of at most $n+t-1$ messages and all processes halt by the end of round 2 in the f.f. runs. Suppose without loss of generality that $R_v \neq \emptyset$. Since **B** is *a fortiori* a FD protocol, by Claim 4.5.1, all processes in R_v receive their message in round 1 of H_v. Consider runs r and r' such that in both, the sender starts with v, but in run r, it immediately crashes, while in run r', it crashes after sending all its round 1 messages to the processes in R_v. Since, by Claim 4.8.1, every process in $q \in B$ gets a round 2 message in some f.f. run of **B**, it must be the case that there is some f.f. run of **B** in which q gets no messages in round 1 (for otherwise q would get a total of three messages in the two f.f. runs; an easy counting argument then shows that there must be at least $n+t$ messages sent in both f.f. runs, a contradiction). Thus, no process detects a failure at the end of round 1 of either r or r'. At the end of round 2, the processes in R_v cannot distinguish r from H_v and cannot distinguish r' from $H_{\overline{v}}$, while the processes in B cannot distinguish r from r'. If processes must decide and halt at the end of round 2 in the f.f. runs of **B**, then the processes in R_v must decide v in r and halt at the end of round 2, while the processes in R_v must decide $\overline{v}$ in r' and halt at the end of round 2. The processes in B cannot distinguish r and r' at the end of round 2. Since all other processes halt at the end of that round, a straightforward induction on the round number shows that they will never be able to distinguish the two runs, hence there must be a violation of Termination or Agreement in at least one of them. This shows that not all processes halt in round 2 of the f.f. runs of **B**.

Now suppose that $n > t+2$ and that all processes decide in the f.f. runs of **B** by the end of round 2. Consider the FD protocol **D** in which all processes follow **B** up to the end of round 2, and then halt, making the same decision as in **B**, if they made a decision at all in **B**. It is easy to check that **D** is indeed a FD protocol. By Claim 4.8.1, **D** is nondecisive. Since **D** can obviously be extended to **B** without any delay for processes to decide, this gives us a contradiction to Theorem 4.7. Therefore, not all processes decide in round 2 of the f.f. runs of **B**. ∎

Theorem 4.9: *In any BA protocol for crash failures that uses at most $n+t-1$ messages* *in H_v and no messages in $H_{\overline{v}}$, if $n > 4$ or $t > 1$ then some process does not halt, and if $n > t+2$ does not even decide, until round 4.*

Proof: First we show that if $n = t+2$ and $t > 1$ then some process does not halt until round 4. (Note that if $n = t+2$ then $n > 4$ implies that $t > 1$, so we need not consider that possibility separately.)

Claim 4.9.1: *Let* **A** *be any BA protocol for crash failures that uses no messages in $H_{\overline{v}}$ and in whose f.f. runs all processes decide by the end of round $M+1$. If $t > 1$ then by the end of round M all processes can distinguish H_v from $H_{\overline{v}}$.*

Proof of Claim 4.9.1: Suppose, by way of contradiction, that $t > 1$ but that there is some process p that cannot distinguish H_v and $H_{\overline{v}}$ until round $M+1$. Since there are no messages in $H_{\overline{v}}$ this means that p receives some message(s) in round $M+1$ of H_v but does not receive any messages in earlier rounds. Let q_1 and q_2 be the senders of those messages (if there is only one such message then $q_1 = q_2$). Consider the run r which is like H_v except that q_1 and q_2 omit to send the round $M+1$ message to p. Thus, p cannot distinguish r from $H_{\overline{v}}$ (since it receives no message in either) while all other correct processes (which exist since $n \geq t+2$) cannot distinguish r from H_v. Furthermore there are at most two faulty processes in r. Since $t > 1$, we have a violation of Agreement in r, contradicting the assumption that **A** is a BA protocol. ∎ Claim 4.9.1

Suppose then, by way of contradiction, that there is some BA protocol **B** that uses $n+t-1$ messages in H_v and no messages in $H_{\overline{v}}$ so that all processes halt (and thus have decided) by round 3 in these runs. Construct a FD protocol from **B** by truncating the latter in round 2. More specifically, at the end of round 2 of some run of **B**, each process p halts as follows: If it cannot distinguish the run from H_v or $H_{\overline{v}}$ it decides v or $\overline{v}$, respectively; otherwise, it discovers a failure. Claim 4.9.1 implies that **D** is, indeed, a FD protocol. Of course, **D** uses at most $n+t-1$ messages in H_v and no messages in $H_{\overline{v}}$ (because so does **B**), contradicting Theorem 4.5.

Next we consider the case $n > t+2$ (and $t > 1$ or $n > 4$). We must show that now some process cannot even decide until round 4.

Claim 4.9.2: *If $t > 1$ or $n > 4$, and $n > t+2$, then any FD protocol* **D** *for crash failures that uses at most $n+t-1$ messages in H_v, no messages in $H_{\overline{v}}$, and at most three rounds in these runs is nondecisive.*

Proof of Claim 4.9.2: First suppose that all processes in R_v receive their message in round 1. It follows from Claim 4.5.3 that all the messages in H_v received by the processes in B are sent by processes in R_v. Let $q \in B$ receive its two messages in H_v from $p_1, p_2 \in R_v$. Of course, these messages must be received in rounds 2 or 3. Consider a run r where the sender has initial value v and in which p_1 and p_2 are faulty and crash before sending a message to q. Thus, q cannot distinguish r from $H_{\overline{v}}$ and will decide $\overline{v}$ in round 3. Since the sender is correct and has initial value v, this is a violation of Validity, proving that **D** is nondecisive in this case.

Thus, we can assume that some process in R_v receives its message after round 1 of H_v. By Claim 4.5.1, a process in R_v cannot receive its message in round 3 of H_v. Thus, some process $p \in R_v$ receives its message in round 2 of H_v. Let p' be the process which sends the message to p in round 2 of H_v. Consider now run r in which the sender's initial value is v and the only faulty process is p' which crashes after sending all its round 2 messages except the one to p. Thus, p cannot distinguish r from H_v while the other processes in R_v (which must exist since $n > t+2$ and thus $|R_v| \geq n-t-1 \geq 2$) cannot distinguish r from $H_{\overline{v}}$. Thus, Agreement is violated in r, proving that **D** is nondecisive in this case as well. ∎ Claim 4.9.2

From Theorem 4.5, Theorem 4.7, and Claim 4.9.2, we conclude that in case $n > t+2$ (and $t > 1$ or $n > 4$) some correct process cannot decide before round 4 in the f.f. runs of a BA protocol that uses $n+t-1$ messages in H_v and no messages in the other, as wanted. ∎

5. Sending omission and general omission failures

The lower bounds for crash failures apply immediately if we have sending omission or general omission failures. We thus focus on finding protocols that attain these bounds. As it stands, neither CF1 nor CF2$_v$, $v = 0, 1$, is correct once we allow omission failures. For example, in CF1, it is no longer the case that when a process in B receives a message v from the sender in round 2, it knows that all the processes in R_v received a message in round 1. Fortunately, there is a simple solution to this problem.

Consider the following modification of CF1, which gives us the protocol family GOF1 (for "General Omission Failures"). Rather than having the sender send to all the processes in R_v in the first round, we arrange the processes in R_v in a linear order, which 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. (In the special case where $R_v = \emptyset$, then the sender sends directly to the processes in B in H_v.) The protocol is illustrated in Figure 2. The key point is that when a process in B receives a message v, it again knows that all the processes in R_v also received such a message.

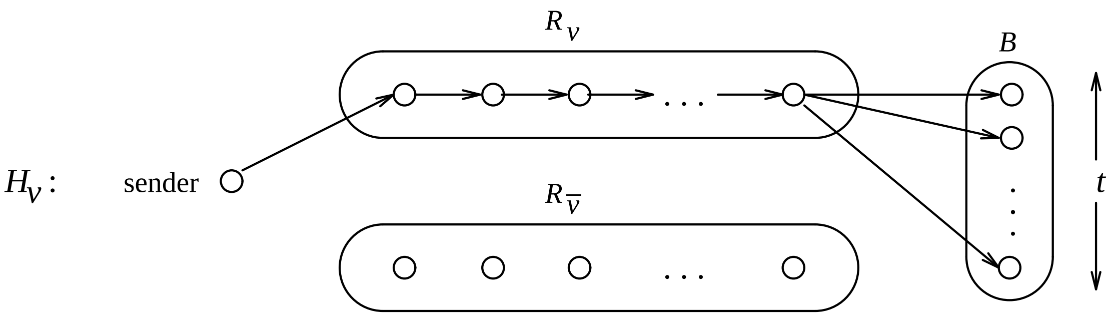


Figure 2: Protocol GOF1

The decision rules for GOF1 are identical to those of CF1, except now the decision takes place at the end of round $\max(|R_0|, |R_1|) + 1$, instead of round 2.

It is easy to verify that (any instance of) GOF1 satisfies the Weak Termination and Weak Validity conditions. Regarding Weak Uniform Agreement, consider any run r in which the sender's initial value in r is v and two processes, say p and q, decide v and $\overline{v}$, respectively. This means that either both p, q are in R_v, p receives a message but q does not, or p is in $R_{\overline{v}}$, q in R_v and neither receives a message. But then, since only general omission failures may occur, the chain of messages from the sender to B via R_v must have been interrupted by a faulty process along that chain (possibly the sender) before reaching any process in B. Hence, no process in B will receive a message in r. Since, as we have seen, some process in R_v or the sender is faulty in r, $|B| = t$ and there can be at most t faulty processes in r, it follows that at least one process in B is correct in r and, not having received any message, it will discover a failure. We have therefore shown that in any run in which two processes reach conflicting decisions, some correct process will discover a failure. Hence, Weak Uniform Agreement holds.

Just as in the case of CF1, every instance of GOF1 uses a total of $n + t - 1$ messages. This gives us

Theorem 5.1: *Every instance of GOF1 solves the Uniform Failure Discovery problem in the case of general omission failures, using a total of $n + t - 1$ messages in the f.f. runs and taking* $\max(|R_0|, |R_1|)+1$ *rounds. By taking* $|R_0| = \lceil (n-t-1)/2 \rceil$ *and* $|R_1| = \lfloor (n-t-1)/2 \rfloor$, *we get a protocol with optimal worst-case f.f. message complexity of* $\lceil (n + t - 1)/2 \rceil$ *and taking* $\lceil (n - t + 1)/2 \rceil$ *rounds.*

To obtain protocols that are optimal for average-case f.f. message complexity we introduce a family of protocols, called GOF2_v, that is based on a suitable modification of CF2_v. We partition the set of receivers into three sets: B_v^2, B^1 and R_v, where $|B_v^2 \cup B^1| = t - 1$. The processes in B_v^2 and R_v are arranged in a linear chain. The protocol is illustrated in Figure 3. In run H_v, the sender sends v through the B_v^2-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_v^2 and B^1 *as well as to the sender.* In $H_{\overline{v}}$ the sender broadcasts message $\overline{v}$ to all the processes in B^1.

The decision rule for receivers in GOF2_v is identical to that in CF2_v, except that now the decision is made at the end of round $|B_v^2| + |R_v| + 1$ rather than at the end of round 3. The decision rule for the sender is a little different: If its initial value is $\overline{v}$ then the sender simply decides $\overline{v}$. If its initial value is v then it decides v if it receives a message in round $|B_v^2| + |R_v| + 1$; if it receives no such message it discovers a failure.

The reader may be wondering why we took $|B_v^2 \cup B^1| = t-1$ and had the sender receive a message in the last round. The "natural" modification of CF2_v is to take $|B_v^2 \cup B^1| = t$ and have these be the only processes that receive a message in the last round. The reason is that this does not quite work. To see why, suppose that $B^1 = \emptyset$ (this is the most interesting case, since it is in this way that we achieve optimal average-case message complexity). Now, if the sender's initial value is v and the first process in B_v^2 is faulty, it will not pass along the v message chain initiated by the sender. All other processes, receiving no message, will think that the run is $H_{\overline{v}}$ while the sender thinks that it is H_v. What happened is that this modification of CF2_v did not preserve a key property of that protocol: In CF2_v (with crash failures), when the sender's initial value is v, the sender knows that if it is correct then all correct processes in B_v^2 and R_v will receive v. As our example shows, the "natural" modification of CF2_v does not have this property. By requiring that the sender receive a

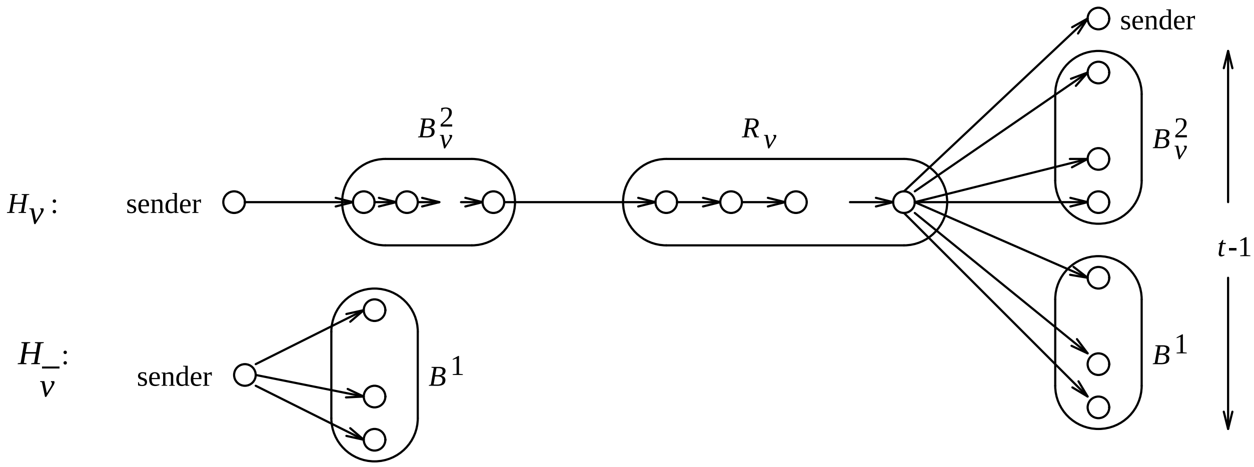


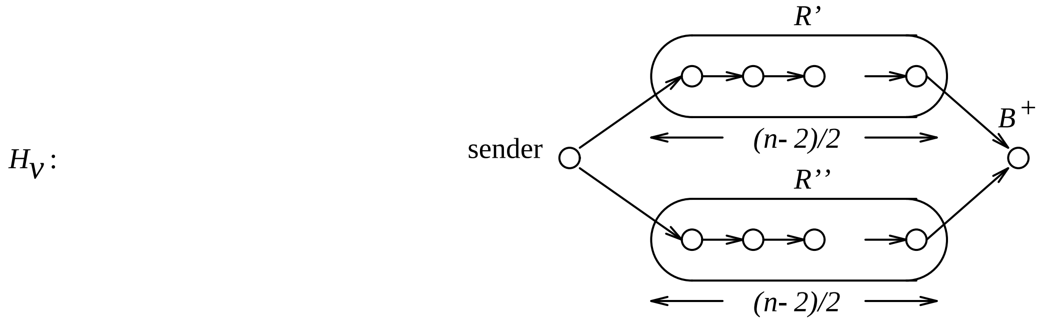
Figure 3: Protocol GOF2$_v$

message in the last round of H_v, however, we restore this property in the following sense: Now when the sender's initial value is v, it knows that if it is correct then either all correct processes in R_v and B_v^2 will receive its message *or* it will discover a failure (in case it does not receive v in the last round).

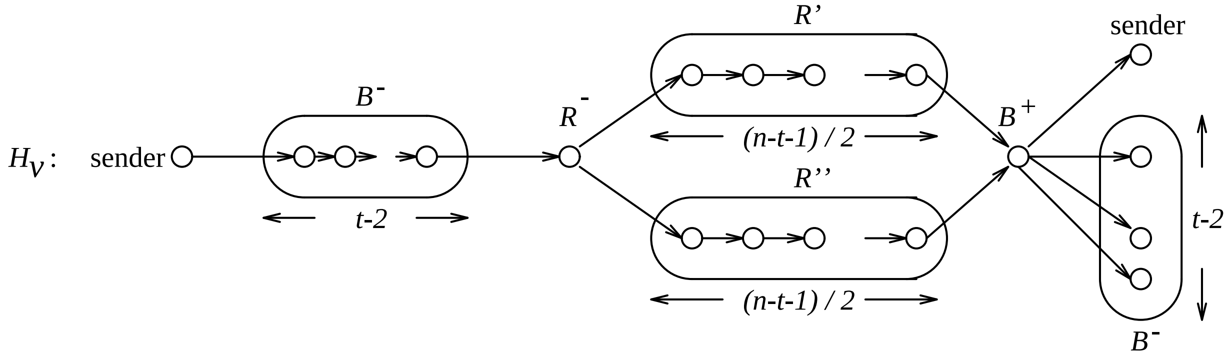
It is easy to verify that GOF2$_v$ satisfies Weak Termination and Weak Validity. Regarding Weak Uniform Agreement, consider any run r in which two processes, say p and q, decide on different values, say v and $\overline{v}$ respectively. Observe that no process decides v in GOF2$_v$ without receiving a message. Since p decides v, it must have received a message v; since we are only dealing with general omission failures here, the sender must have had initial value v and sent a message v. (Note that this is true even if p is the sender, since the last process in R_v must send a message to the sender in H_v.) Moreover, it must be the case that q is in either R_v or B_v^2, and does not receive a message. It follows that some process preceding q in the chain, say p', did not forward the sender's message, p precedes p' in the chain (or is identical to it), and all processes following q in the chain do not receive a message (or a second message, in the case of processes in B_v^2 following q). If q is in R_v, since $|B^1 \cup B_v^2| = t-1$, the sender or some process in $B^1 \cup B_v^2$ must be correct; this process will discover a failure. (The remaining case, that q is in B_v^2, cannot occur because then p is the sender or a process in B_v^2 preceding q in the chain; in either case, p will not get the message v that it is expecting in the last round, and hence will discover a failure contradicting the assumption that it decided v.) We have thus shown that if two processes reach conflicting decisions, there must be a correct process that discovers a failure. Therefore, Weak Uniform Agreement also holds.

Clearly, any instance of GOF2$_v$ uses $|B_v^2| + |R_v| + (|B^1| + |B_v^2| + 1) = n + t - 1 - |B^1|$ messages in H_v and $|B^1|$ messages in $H_{\overline{v}}$. Thus we have shown:

Theorem 5.2: *Every instance of protocol family GOF2$_v$, $v = 0, 1$, solves the Uniform Failure Discovery problem in the case of general omission failures, using a total of $n + t - 1$ messages in the f.f. runs. If $P_{\overline{v}} \geq P_v$, by taking $B^1 = \emptyset$, we get a protocol with optimal average-case f.f. message complexity of $P_v \cdot (n + t - 1)$ which takes n rounds.*



GOF3$_v$ when $t = 1$



GOF3$_v$ when $t > 1$

Figure 4: Protocol GOF3$_v$

We can get a FD protocol for general omission failures that has optimal average-case f.f. message complexity but takes fewer rounds than GOF2$_v$. For the rest of this section, define

$$\Delta(t) = \begin{cases} 0 & \text{if } t = 1 \\ 1 & \text{if } t > 1. \end{cases}$$

Theorem 5.3: *There is a Uniform Failure Discovery protocol for general omission failures that has average-case f.f. message complexity of* $\min(P_0, P_1) \cdot (n + t - 1)$, *and takes* $\lceil (n + t - 1)/2 \rceil + \Delta(t)$ *rounds.*

Proof sketch: We describe a protocol GOF3$_v$ which is a modification of GOF2$_v$. Suppose that $P_{\overline{v}} \geq P_v$. First consider the case $t = 1$. In that case, illustrated at the top of Figure 4, we partition the receivers into three groups: R', R'', and B^+, where $|R'| = \lceil (n-2)/2 \rceil$, $|R''| = \lfloor (n-2)/2 \rfloor$, and $|B^+| = 1$. We linearly order the processes in R' and R''. In $H_{\overline{v}}$, the sender sends no messages (and neither does any other process). In H_v, the sender sends its value to the first process in each of R' and R''. The value then travels down the R' and R'' chains. The last processes in the R' and R'' send the value to the process in B^+. (In the special case where R'' is empty—which can happen if $n \leq 3$—the sender sends the value directly to the process in B^+.) Processes in R' and R'' decide v if they get a message; otherwise they decide $\overline{v}$. The process in B^+ decides v if it receives two messages, decides $\overline{v}$ if it receives no message, and discovers a failure otherwise. Clearly this protocol is correct, uses $n = n + t - 1$ messages, and takes $\lceil n/2 \rceil = \lceil (n + t - 1)/2 \rceil + \Delta(t)$ rounds.

If $t > 1$, the simple idea above must be slightly modified. Refer to the bottom of Figure 4. We partition the receivers into five sets: R', R'', R^-, B^-, and B^+ where $|R'| = \lceil (n-t-1)/2 \rceil$, $|R''| = \lfloor (n-t-1)/2 \rfloor$, $|R^-| = 1$, $|B^-| = (t-2)$, and $|B^+| = 1$. We linearly order the processes in all these sets. If the sender's initial value is $\overline{v}$, it sends no messages. If its initial value is v, in the first round, it sends v to the first process in the B^- chain. The message then travels down this chain; the last process in the B^- chain passes it on to the single process in R^- which, in turn, passes it on to the first processes in the R' and R'' chains. The message then travels down these two chains. The last process in R' and the last process in R'' both pass the message on to the single process in B^+, which gets the two messages by round $\lceil (n+t-1)/2 \rceil$. (If R'' is empty — which can happen if $n = t+2$ — then the process in R^- sends one message to the first process in R' and one to B^+ directly. R' cannot be empty since $n > t+1$.) In the next and final round, this process broadcasts v to all the processes in B^- and the sender. At the end of round $\lceil (n+t-1)/2 \rceil + 1$, processes decide as follows: Processes in $R^- \cup R' \cup R''$ decide v if they receive a message, otherwise they decide $\overline{v}$. Processes in $B^- \cup B^+$ decide v if they get two messages, decide $\overline{v}$ if they get no messages, and discover a failure if they only receive one message. The sender decides $\overline{v}$ if its initial value is $\overline{v}$, it decides v if it receives v in round $\lceil (n+t-1)/2 \rceil + 1$, and it discovers a failure if its initial value is v but receives no message.

Clearly GOF3$_v$ takes $\lceil (n+t-1)/2 \rceil + \Delta(t)$ rounds and uses $n+t-1$ messages in H_v (since the sender and every process in $R^- \cup R' \cup R''$ receive one message each, while the processes in $B^- \cup B^+$ receive two messages each) and no message in $H_{\overline{v}}$. We leave the proof of correctness to the reader. ∎

Of course, the question now arises if we can do better, especially given that we can achieve FD in the crash failure case using $n+t-1$ messages and only two rounds. As we now show, our protocols have the best possible round complexity among all protocols that are message optimal. Indeed, the following proof shows much more. It shows that any protocol that gives the best possible round complexity *must* have the structure of GOF1 or GOF3$_v$, depending on whether we are considering worst-case or average-case complexity. In particular, it is crucial in protocol GOF3$_v$ that all processes in B except the sender and B^+ receive messages both before and after every process in R_v receives its message, and that the chains that convey the message to R' and R'' "split" at a process which is in R_v, not in B (cf. Claim 5.4.14 below).

Theorem 5.4: *Every Failure Discovery protocol for sending omission failures that uses a total of $n+t-1$ messages in the f.f. runs requires at least $\lceil (n-t+1)/2 \rceil$ rounds in one of the f.f. runs. If, in addition, no messages are sent in $H_{\overline{v}}$, then at least $\lceil (n+t-1)/2 \rceil + \Delta(t)$ rounds are required in H_v.*

Proof: For the rest of the proof, fix a protocol P that has total f.f. message complexity $n+t-1$.

A *pseudochain from process p to process q* in protocol P is a sequence of processes $p = p_0, p_1, \ldots, p_k = q$ such that there exist messages $m_1, m_2, \ldots, m_k$ and round numbers $\ell_1 < \ell_2 < \ldots < \ell_k$ so that p_{i-1} sends m_i to p_i in round ℓ_i of *either* H_0 *or* H_1, for all $1 \le i \le k$. We say that such a pseudochain *begins* in round ℓ_1 and *ends* in round ℓ_k. In other words, a pseudochain consists of chains in H_0 and H_1 which are juxtaposed in such a way that for any two successive chains in the juxtaposition, the later chain begins after the earlier ends. We write $p \prec q$ if there is a pseudochain from p to q.

The key idea in the proof is to show that protocol P must contain fairly long pseudochains. A long pseudochain implies that one of the f.f. runs of the protocol must have many rounds, because of the requirement of monotonically increasing round numbers in the definition of pseudochain.

Claim 5.4.1: *No process receives a total of more than two messages in the f.f. runs, and the sender receives at most one message in the f.f. runs.*

Proof of Claim 5.4.1: Let $B' \subseteq B$ be the set consisting of the receivers which receive more than two messages in the f.f. runs and the sender if it receives more than one message in the f.f. runs. Let α be either 1 or 0, according as the sender is in B or not. Then the total number of messages in the f.f. runs of P is at least

$$\begin{aligned}|R_0| + |R_1| + 2(|B| - |B'|) + 3|B'| - \alpha &= (n - 1 + \alpha - |B|) + 2(|B| - |B'|) + 3|B'| - \alpha \\ &= n - 1 + |B| + |B'|.\end{aligned}$$

Since P has optimal message complexity, we have $n + t - 1 \geq n - 1 + |B| + |B'|$, which implies that $|B| + |B'| \leq t$. From Theorem 4.2 it follows that $|B| = t$. Therefore $B' = \emptyset$, as wanted. ▮ Claim 5.4.1

Claim 5.4.2: *Suppose $p \in R_v$ receives a message in round ℓ of H_v, $v \in \{0, 1\}$, and there is no pseudochain from p to any process in $R_0 \cup R_1$ that begins after round ℓ. Then for every $q \in B$ there is a pseudochain from p to q which begins in some round later than ℓ.*

Proof of Claim 5.4.2: Suppose $p \in R_v$ is as in the statement of the claim. Using notation as in Lemma 4.1, let A_1 be the set of processes affected by p after round ℓ in H_v, A_2 be the set of processes that receive a message from p after round ℓ in $H_{\overline{v}}$ and A_3 be the set of processes affected by processes in A_2 after round ℓ in H_v. By choice of p, A_1, A_2 and A_3 cannot include any processes in $R_0 \cup R_1$ and therefore $A_1 \cup A_2 \cup A_3 \subseteq B$. By Lemma 4.1(2), $|A_1 \cup A_2 \cup A_3| \geq t$ and by Theorem 4.2, $|B| = t$. Thus, $A_1 \cup A_2 \cup A_3 = B$. But, by definition of A_1, A_2 and A_3, there is a pseudochain from p to q that begins after round ℓ for every $q \in A_1 \cup A_2 \cup A_3$. ▮ Claim 5.4.2

Claim 5.4.3: *For every receiver p, we have $s \prec p$, where s is the sender. Moreover, if $p \in R_0 \cup R_1$, then the pseudochain from s to p is unique, and for all processes p', if $p' \prec p$, then the pseudochain from p' to p is a suffix of the unique pseudochain from s to p.*

Proof of Claim 5.4.3: A straightforward induction on ℓ proves the following statement:

> For any $p \neq s$, if ℓ is the earliest round at the end of which p can distinguish H_0 from H_1 (i.e., p has a different view of H_0 than of H_1 at the end of round ℓ) then there is a pseudochain from s to p which ends in round ℓ.

Since by the end of the maximum round in H_0 or H_1 every process must be able to distinguish H_0 from H_1, it follows that for every $p \neq s$, $s \prec p$.

Now suppose $p \in R_0 \cup R_1$ and $p' \prec p$. Let $\psi(s,p)$ denote a pseudochain from s to p. Consider any pseudochain from p' to p, denoted $\psi(p',p)$ First note that $\psi(s,p)$ and $\psi(p',p)$ must share a nonempty common suffix, for otherwise, p would receive two messages in the f.f. runs, contradicting the assumption that it is in $R_0 \cup R_1$. Let $p'' \prec p$ be such that the longest common suffix of $\psi(s,p)$ and $\psi(p',p)$ is from p'' to p. If $p'' \neq p'$ then p'' must receive at least two messages in the f.f. runs *before* p receives its message. Thus $p'' \in B$. Claim 5.4.2 implies, in particular, that every process in B receives a message after the last round in which any process in $R_0 \cup R_1$ receives a message in a f.f. run. Thus, p'' must receive a message in a f.f. run *after* p receives its message, in addition to the two p'' receives before that time. Thus, p'' receives at least three messages in the f.f. runs, contrary to Claim 5.4.1. Thus, we must have $p'' = p'$, so $\psi(p',p)$ is a suffix of $\psi(s,p)$. Taking $p' = s$ in the above argument, it follows that the pseudochain from s to p is unique. ▮ Claim 5.4.3

Claim 5.4.4: *The restriction of $\prec$ to $R_0 \cup R_1$ is a strict partial order.*

Proof of Claim 5.4.4: We must show that $\prec$, restricted to $R_0 \cup R_1$, is irreflexive and transitive. To prove irreflexivity, observe that if $p \prec p$ for some $p \in R_0 \cup R_1$ then, by Claim 5.4.3, the pseudochain from p to p must be a suffix of the pseudochain from the sender to p, implying that p receives at least two messages in the f.f. runs. This contradicts the assumption that $p \in R_0 \cup R_1$. To prove transitivity, suppose that for $p_1, p_2, p_3 \in R_0 \cup R_1$ we have $p_1 \prec p_2$ and $p_2 \prec p_3$. Thus, the pseudochain from the sender to p_3 contains a suffix consisting of a pseudochain from p_1 to p_2 and a pseudochain from p_2 to p_3. That suffix is a pseudochain from p_1 to p_3, proving that $p_1 \prec p_3$, as wanted. ▮ Claim 5.4.4

Claim 5.4.5: *For any $p, p_1, p_2 \in R_0 \cup R_1$, if $p_1 \prec p$ and $p_2 \prec p$ then either $p_1 \prec p_2$ or $p_2 \prec p_1$.*

Proof of Claim 5.4.5: By Claim 5.4.3 there is a unique pseudochain from the sender to p and suffixes from p_1 to p and from p_2 to p on that pseudochain. The difference of these two suffixes is a pseudochain from p_1 to p_2 or vice-versa. Therefore, either $p_1 \prec p_2$ or $p_2 \prec p_1$.
▮ Claim 5.4.5

According to Claim 5.4.5, the partial order $\prec$ restricted to $R_0 \cup R_1$ is of a very special form. To see this, it is helpful to imagine a digraph whose nodes are labeled with the processes in $R_0 \cup R_1$ and an edge from node p to q if there is a pseudochain from p to q which does not involve any other processes in $R_0 \cup R_1$. The fact that $\prec$ partially orders the processes in $R_0 \cup R_1$ means that this digraph is acyclic. Claim 5.4.5 means that, furthermore, it is a forest. Now we shall prove (Claim 5.4.7) that even this forest has a very special property, namely that it has at most two leaves.

Claim 5.4.6: *For any $p \in R_0 \cup R_1$ and $q \in B$, $p \prec q$.*

Proof of Claim 5.4.6: Let $p \in R_0 \cup R_1$ and p' be a $\prec$-maximal element of $R_0 \cup R_1$ such that $p \prec p'$ or $p = p'$. Let the pseudochain from p to p' (by Claim 5.4.3 it must be unique) end in round ℓ. By Claim 5.4.2, there is a pseudochain from p' to q that begins after ℓ, for all $q \in B$. Thus, there is a pseudochain from p to q, i.e., $p \prec q$, for all $q \in B$. ▮ Claim 5.4.6

Claim 5.4.7: *There are at most two $\prec$-maximal elements in $R_0 \cup R_1$.*

Proof of Claim 5.4.7: Suppose, by way of contradiction, that there are three $\prec$-maximal elements in $R_0 \cup R_1$, say p_1, p_2, p_3. By Claim 5.4.6, for any $q \in B$, $p_i \prec q$, for $i = 1, 2, 3$. As we now show, there is some $q \in B$ and pseudochains from each p_i, $i = 1, 2, 3$, to q, so that no two pseudochains have a non-empty common suffix. But then q receives three distinct messages in the f.f. runs (one in each of these three pseudochains), contradicting Claim 5.4.1.

We choose q as follows. Consider all choices of $q' \in B$, and pseudochains $\psi(p_i, q')$ from p_i to q', $i \in \{1, 2, 3\}$. Select q', $\psi(p_i, q')$ and $\psi(p_j, q')$ so that the common suffix of $\psi(p_i, q')$ and $\psi(p_j, q')$ is as long as possible, among all choices of q', i, j with $i \neq j$. If this common suffix is empty then we can take $q = q'$ and we have a process in B with the desired property. Otherwise, let q be the process so that the nonempty longest common suffix of $\psi(p_i, q')$ and $\psi(p_j, q')$ is a pseudochain $\psi(q, q')$ from q to q' and let $\psi(p_i, q)$ and $\psi(p_j, q)$ be the prefixes of $\psi(p_i, q')$ and $\psi(p_j, q')$ respectively, which end with process q. Naturally, $q \in B$. Let p_k be the $\prec$-maximal process in $\{p_1, p_2, p_3\} \setminus \{p_i, p_j\}$. By Claim 5.4.6, $p_k \prec q$. No pseudochain $\psi(p_k, q)$ from p_k to q can have a non-empty common suffix with $\psi(p_i, q)$, for if so, there would be a pseudochain $\psi(p_k, q')$ from p_k to q' which would have a strictly longer common suffix with $\psi(p_i, q')$ than $\psi(p_j, q')$ does. Symmetrically, $\psi(p_k, q)$ does not have a non-empty common suffix with $\psi(p_j, q)$. Therefore, q is a process in B so that there exist pseudochains from each p_i to q, no two of which have a non-empty common suffix, as wanted. ∎ Claim 5.4.7

We have therefore shown that $R_0 \cup R_1$ is partitioned into two subsets, each of which is totally ordered by $\prec$. Since $|R_0 \cup R_1| \geq n - t - 1$, one of these subsets must contain at least $\lceil (n-t-1)/2 \rceil$ elements, say $p_1, p_2, \ldots, p_{\lceil (n-t-1)/2 \rceil}$. Therefore (using Claims 5.4.2 and 5.4.3), if q is any process in B, we have that

$$s \prec p_1 \prec p_2 \prec \ldots \prec p_{\lceil (n-t-1)/2 \rceil} \prec q.$$

This means that there exist at least $\lceil (n-t-1)/2 \rceil + 1 = \lceil (n-t+1)/2 \rceil$ rounds in one of the f.f. runs of the protocol.

For the remainder of the proof, we assume that no messages are sent in $H_{\overline{v}}$. (Therefore all pseudochains referred to below are actually chains in H_v.) We want to show that there is a pseudochain of length $\lceil (n+t-1)/2 \rceil + \Delta(t)$ in H_v. If $t = 1$ we are already done, so assume $t > 1$. Let q_0 be a process in B so that no process in B receives its last message in H_v earlier than q_0. (By Claim 5.4.1 and the assumption that there are no messages in $H_{\overline{v}}$, it follows that processes in B receive at most two messages and the sender—if it is in B—receives at most one message in H_v. Thus, "last" here means "second", in the case of processes in B other than the sender, and "first" in the case of the sender.)

Claim 5.4.8: *There are at most two pseudochains $\psi_1(s, q_0)$ and $\psi_2(s, q_0)$ from the sender to q_0 (we may have $\psi_1(s, q_0) = \psi_2(s, q_0)$), and every pseudochain ending with q_0 is a suffix of one of these two chains.*

Proof of Claim 5.4.8: Suppose there were three distinct pseudochains $\psi_i(s, q_0)$, $i = 1, 2, 3$ from s to q_0. Since q_0 receives at most two messages (by Claim 5.4.1), at least two of these pseudochains must have a common suffix. Consider the longest common suffix of two of the three pseudochains, and let q be the first process in this common suffix. Then q must receive two messages and do so before q_0 does, contradicting the choice of q_0. A similar argument shows that every pseudochain ending with q_0 must be a suffix of one of these two chains. ▮ Claim 5.4.8

It is actually true that there are at most two pseudochains from the sender to *any* process in B, not just to q_0. We do not prove this here since we shall not need this fact, but it makes for a good exercise.

Claim 5.4.9: *Suppose that $q \in B$ and q receives messages from p_1 in round m_1 and p_2 in round m_2, and $m_1 \leq m_2$. Then at least $t - 1$ processes other than p_1 and p_2 are affected by q after round m_1 in H_v.*

Proof of Claim 5.4.9: The proof is almost identical to that of Lemma 4.1(1), using the fact that no messages are sent in $H_{\overline{v}}$; we omit it here. ▮ Claim 5.4.9

Claim 5.4.10: *For every $q \in B$ and any $p \neq q$, we have $p \prec q$.*

Proof of Claim 5.4.10: If p is the sender, this follows by Claim 5.4.3; if $p \in R_v$, it follows by Claim 5.4.6. If $p \in B$ and $p \neq q$, then by Claim 5.4.9, p affects at least $t - 1$ processes after receiving its first message. If one of these is in R_v, say p', then we can concatenate the unique (by Claim 5.4.3) pseudochain from p to p' to the pseudochain from p' to q since the latter must start after p' gets the one message that it does in H_v (otherwise, p' would be sending messages in $H_{\overline{v}}$!). If p does not affect any processes in R_v, then all the $t - 1$ processes it affects must be in B. Thus, it must affect all the other processes in B. In particular, p affects q. Thus, $p \prec q$, as desired. ▮ Claim 5.4.10

From Claims 5.4.8 and 5.4.10 (taking $q = q_0$ in the latter) we get immediately,

Claim 5.4.11: *Every process must be on either $\psi_1(s, q_0)$ or $\psi_2(s, q_0)$.*

In the remainder of the proof, we abbreviate $\psi_1(s, q_0)$ and $\psi_2(s, q_0)$ by ψ_1 and ψ_2, respectively. Observe that if ψ_i is a prefix of ψ_j for $i \neq j$ then, by Claim 5.4.11, the longer of the two pseudo-chains would then have to include every process at least once, meaning that H_v must have at least n rounds. Since $n \geq \lceil (n + t - 1)/2 \rceil + \Delta(t)$, we can assume without loss of generality that neither one of ψ_1 or ψ_2 is a prefix of the other. More generally, we can assume that for *any* process $q \in B$ which receives its last message in H_v in the same round as q_0 does, the two pseudo-chains from the sender to q are not one a prefix of the other. In particular, this means that q_0 receives its two messages by virtue of being the last process on both ψ_1 and ψ_2, and does not appear in a prefix of ψ_1 or ψ_2.

Claim 5.4.12: *Given any pseudo-chain ψ starting with the sender, either ψ is a prefix of ψ_i, or vice-versa, for some $i \in \{0, 1\}$.*

Proof of Claim 5.4.12: Suppose, by way of contradiction, that ψ and ψ_i are not one a prefix of the other, for either $i \in \{0,1\}$. Let ψ' be the longest prefix of ψ which is a prefix of either ψ_1 or of ψ_2. We cannot have $\psi' = \psi$, for otherwise ψ would be a prefix of ψ_1 or ψ_2. We also cannot have q_0 on ψ', for otherwise one of ψ_1, ψ_2 would be a prefix of the other, which we have assumed is not the case. Let q_1 be the first process in ψ following the prefix ψ'. Since every process in R_v is on ψ_1 or ψ_2, and receives only one message in H_v, it must be the case that $q_1 \in B$. We now show that q_1 must receive three messages in H_v, leading to a contradiction of Claim 5.4.1. If q' is the last process on ψ', then q_1 receives a message from q'. Clearly, this message is not in ψ_1 or ψ_2. Since, by Claim 5.4.10, we have $q_0 \prec q_1$, it must be the case that q_1 receives a message on a pseudo-chain starting with q_0 (which cannot be a subchain of ψ', ψ_1, or ψ_2, for then q_0 would appear in a prefix of ψ_1 or ψ_2). Finally, by Claim 5.4.11, q_1 must receive a message on one of ψ_1 or ψ_2. This gives us the desired contradiction. ▮ Claim 5.4.12

Claim 5.4.13: *The sender s is in B.*

Proof of Claim 5.4.13: Assume, by way of contradiction, that $s \notin B$. It follows from Claim 5.4.12 that s sends at most 2 messages in H_v. Say s sends to p_1 and p_2 in H_v (we may have $p_1 = p_2$). Consider the run r where s has initial value v, both p_1 and p_2 crash at the beginning, while all other processes are correct. (There are two faulty processes in r, which is possible since we are assuming $t \geq 2$.) Since $s \notin B$, it does not receive any messages in H_v. Thus, s cannot distinguish r from H_v. All the other processes cannot distinguish r from $H_{\overline{v}}$, since they get no messages, and thus decide $\overline{v}$. This gives us a contradiction. ▮ Claim 5.4.13

Claim 5.4.14: *The longest common prefix of ψ_1 and ψ_2 contains all processes in B other than q_0 and at least one process in R_v.*

Proof of Claim 5.4.14: Let $p \in R_v$ be such that no process in R_v receives its message in H_v before p does. Without loss of generality, p is on ψ_1. First, we shall prove that

(*) All processes in B other than q_0 precede p, and p sends exactly two messages in H_v.

Let A_1 consist of all the processes that precede p on ψ_1, and let A_2 consist of all processes to which p sends messages in H_v. By choice of p, no process in R_v can precede p on ψ_1. Since $s \in B$ and q_0 cannot appear in a prefix of ψ_1, A_1 consists of at most the processes in B other than q_0; thus, $|A_1| \leq t-1$. Moreover, p can send at most two messages in H_v, for otherwise there would be three pseudo-chains starting with s, none of which is a prefix of the other, leading to a contradiction with Claim 5.4.12; thus $|A_2| \leq 2$.

Therefore, $|A_1 \cup A_2| \leq t+1$. To complete the proof of (*) it remains to show that $|A_1 \cup A_2| \geq t+1$. Suppose, by way of contradiction, that $|A_1 \cup A_2| \leq t$ and consider the run r defined as follows. In r, the sender has initial value v. All processes in A_1 omit to send messages to processes other than the messages on the ψ_1 pseudo-chain. In addition, the processes in A_2 crash immediately. Finally, all processes other than those in $A_1 \cup A_2$ are correct. Thus, there are at most t faulty processes in r. It is easy to see that p cannot distinguish r from H_v; it gets the message v in both runs. All other correct processes cannot distinguish r from $H_{\overline{v}}$, since they get no messages. This gives us a violation of Weak Agreement. Thus, we must have $|A_1 \cup A_2| \geq t+1$, completing the proof of (*).

Observe that no process that is on one of ψ_1 or ψ_2 but not the other can send more than one message, for otherwise we would have three pseudo-chains starting with s none of which is a prefix of the other, giving us a contradiction to Claim 5.4.12. By (*) then, p and all the processes that precede it must be on the longest common prefix of ψ_1 and ψ_2. Recalling that p is in R_v gives us the Claim. ∎ Claim 5.4.14

Let ℓ be the round in which q_0 receives its last message in H_v.

Claim 5.4.15: *Every process in B other than q_0 receives a message after round ℓ.*

Proof of Claim 5.4.15: Suppose $q_1 \in B$ receives its last message at round ℓ (it cannot receive it any earlier by choice of q_0). Then there must be a pseudo-chain ψ from s to q_1 ending at round ℓ (for, otherwise, some process other than the sender would send a message without receiving a message, contradicting the fact that no messages are sent in $H_{\overline{v}}$). Pseudo-chain ψ is not a prefix of either ψ_1 or ψ_2, for otherwise q_0 would be receiving its last message in H_v after round ℓ. Also, neither one of ψ_1 and ψ_2 can be a prefix of ψ. To see this note that, by Claim 5.4.14, q_1 is on the longest common prefix of ψ_1 and ψ_2. Thus if ψ_1 or ψ_2 was a prefix of ψ, q_1 would be a process receiving its last message in H_v in the same round as q_0 does, with the property that one of the two pseudo-chains from the sender to q_1 is a prefix of the other, something we have already assumed is not the case. Thus, for both $i = 0, 1$, ψ is not a prefix of ψ_i nor vice versa, contradicting Claim 5.4.12.
∎ Claim 5.4.15

Now we have all the ingredients necessary for the lower bound on the number of rounds in H_v. We first show that the longer of ψ_1 and ψ_2 must contain at least $\lceil (n+t+1)/2 \rceil$ processes. Let k be the number of processes in R_v that appear on the longest common prefix of ψ_1 and ψ_2. By Claim 5.4.14, $k \geq 1$. In addition, by Claim 5.4.14, this common prefix must contain all the processes in B except q_0, for a total of $k+t-1$ processes. Now consider the non-common parts of ψ_1 and ψ_2. By Claim 5.4.11, between them they must cover all of the $n-t-k$ processes in R_v that are not in the common prefix of ψ_1 and ψ_2. Thus, the longer of them must contain at least $\lceil (n-t-k)/2) \rceil$ processes. Finally, q_0 is on both ψ_1 and ψ_2. Thus, the longer of ψ_1 and ψ_2 has at least $\lceil (n+t+k)/2 \rceil$ processes. Since $k \geq 1$, the result follows.

Thus, it takes at least $\lceil (n+t-1)/2 \rceil$ rounds to pass messages down the longer pseudo-chain. From Claim 5.4.15, the algorithm requires at least another round after that, giving us a total of $\lceil (n+t-1)/2 \rceil + 1$ rounds. Since we are assuming $t > 1$, we have $\Delta(t) = 1$, and we are done. ∎

Theorems 4.3 and 5.2 show that there is a significant difference between the number of rounds required for message-optimal protocols in the case of crash failures and omission failures. Such results are particularly interesting because for most complexity measures, crash failures and omission failures have the same bounds.

The results in this section were stated with respect to FD protocols. What can be said about the f.f. message and round complexity of BA protocols for sending and general omission failures? Since GOF1 and GOF3$_v$ are Uniform FD protocols, we can apply the transformations of Theorem 2.2 to extend them in a round-efficient manner to BA protocols that are message-optimal in the f.f. runs. For the remainder of the section, let $M_{wc} = \lceil (n-t+1)/2 \rceil$ and $M_{ac} = \lceil (n+t-1)/2 \rceil + \Delta(t)$. These quantities are the round complexities of GOF1 and GOF3$_v$, respectively, which as shown in Theorem 5.4 are best possible for message-optimal FD protocols.

Theorem 5.5: *(a) There is a BA protocol for general omission failures that has worst-case f.f. message complexity of $\lceil (n+t-1)/2 \rceil$ in which all processes decide and halt in $M_{wc}+1$ rounds in the f.f. runs. In the special case $n = t+2$, processes can decide (but not halt) in round M_{wc} in the f.f. runs.*

(b) There is a BA protocol for sending omission failures that has average-case f.f. message complexity of $\min(P_0, P_1) \cdot (n+t-1)$ in which if $t > 1$ all processes decide in $M_{ac}+1$ and halt in $M_{ac}+2$ rounds, and if $t = 1$ all processes decide and halt in $M_{ac}+1$ rounds in the f.f. runs.

(c) There is a BA protocol for general omission failures that has average-case f.f. message complexity of $\min(P_0, P_1) \cdot (n+t-1)$ so that if $1 < t < n/2$ all processes decide and halt in $M_{ac}+2$ rounds, and if $t = 1$ all processes decide and halt in $M_{ac}+1$ rounds in the f.f. runs.

Proof: For part (a), if $n = t+2$, it is easy to check that GOF1 is safe, so we can apply the transformation of Theorem 2.2(a) to GOF1. If $n > t+2$, GOF1 is not safe, but the sender cannot detect a failure, so we can apply the transformation of Theorem 2.2(b) to GOF1. If $t > 1$, for part (b) (resp. (c)), apply the transformation of Theorem 2.2(c) (resp. Theorem 2.2(d)) to the version of GOF3$_v$ for $t > 1$, where v is the less likely initial value. If $t = 1$, for both part (b) and (c) apply the transformation of Theorem 2.2(b) to the version of GOF3$_v$ for $t = 1$, where v is the less likely initial value. (Note that in this version of GOF3$_v$ the sender cannot discover a failure, so this transformation is applicable.) ∎

By Theorem 4.3, the worst-case or average-case f.f. message complexity of the BA protocols of Theorem 5.5 cannot be improved. The question arises whether the round complexity can be improved without damage to their message optimality. Theorem 5.4 already shows that we cannot hope for an improvement of more than one or two rounds. As the next two theorems show, no improvement at all is actually possible. To prove these results we shall make extensive use of the properties of message-optimal FD protocols which were stated as Claims 5.4.1–5.4.13 in the proof of Theorem 5.4.

Lemma 5.6: *Suppose $n > t+2$ or the sender is in B, and let $\mathbf{D}$ be a FD protocol for sending omission failures that uses a total of at most $n+t-1$ messages in the f.f. runs.*

(a) If $\mathbf{D}$ uses at most M_{wc} rounds in the f.f. runs, then it is nondecisive.

(b) If $\mathbf{D}$ uses at most M_{ac} rounds in the f.f. runs and does not have any messages in one of them, then it is nondecisive.

Proof: (a) By Claim 5.4.7, there are at most two $\prec$-maximal elements in $R_0 \cup R_1$, say p and q (it is possible that $p = q$). By Claim 5.4.3, there are unique pseudochains from the sender s to p and s to q, and, by the argument in the paragraph following the proof of Claim 5.4.7, the last message on one of these pseudochains, say the one to p, is sent in round $M_{wc} - 1$. Let p' be the predecessor of p in the pseudochain from s to p. Let $v \in \{0, 1\}$ be such that $p \in R_v$. Thus, p gets a message in H_v but not in $H_{\overline{v}}$. Consider the run r of $\mathbf{D}$ in which the sender's initial value is v and the only deviation from correct behaviour is that p' omits to send its message to p in round $M_{wc} - 1$. In round M_{wc}, process p cannot distinguish r from $H_{\overline{v}}$, while all other correct processes in $R_0 \cup R_1$ cannot distinguish r from H_v. Let $\alpha = 1$ or 0, depending on whether the sender is in B or not. By Theorem 4.2, $|B| = t$, so $|R_0 \cup R_1| = (n - 1 + \alpha) - |B| = n - t - 1 + \alpha$. Therefore, if $n > t + 2$ or the sender is in B, $R_0 \cup R_1$ contains some process other than p. This means that Agreement is violated in r and so $\mathbf{D}$ is nondecisive.

(b) If $t = 1$ then $M_{ac} = M_{wc}$ and we are done by the previous case, so suppose $t > 1$. By Claim 5.4.8, there are at most two pseudochains from the sender to q_0. Let q_1 and q_2 be the predecessors of q_0 on these pseudochains. Consider a run r in which the only faulty processes are q_1 and q_2, which fail to send their message to q_0. In this case, we get a violation of Validity, which means that $\mathbf{D}$ is nondecisive. ∎

Theorem 5.7: *In any BA protocol for sending omission failures that uses a total of at most $n + t - 1$ messages in the f.f. runs, some process does not halt, and if $n > t + 2$ does not even decide, until round $M_{wc} + 1$.*

Proof: From Lemma 5.6(a), in conjunction with Theorem 4.7 and Theorem 5.4, we conclude that no process can decide before round $M_{wc} + 1$ if $n > t + 2$ or the sender is in B. It remains to show that no process can halt before round $M_{wc} + 1$ in the case $n = t + 2$ and the sender is not in B. But in this case we have $M_{wc} = 2$, so the result follows immediately from Theorem 4.8. ∎

Theorem 5.8: *Let $\mathbf{B}$ be a BA protocol for sending omission failures that uses at most $n + t - 1$ messages in one f.f. run and no messages in the other. Then in some f.f. run of $\mathbf{B}$ some process does not decide until round $M_{ac} + 1$. If $t > 1$ then some process does not halt until round $M_{ac} + 2$. If $\mathbf{B}$ also tolerates general omission failures and $t > 1$ then some process does not even decide until round $M_{ac} + 2$; in this case, if $n \geq 4$ and all processes decide in round $M_{ac} + 2$ in all f.f. runs, then we must have $n > 2t$.*

Proof: Let $\mathbf{D}$ be a FD protocol for sending omission failures with $n + t - 1$ messages in H_v and no messages in $H_{\overline{v}}$. By Claim 5.4.13, the sender is not in B and therefore, by Lemma 5.6(b), $\mathbf{D}$ is nondecisive. Thus, by Theorems 4.7 and 5.4, we conclude that some process does not decide until round $M_{ac} + 1$ in some f.f. run of $\mathbf{B}$. It remains to show that if $t > 1$ then in some f.f. run of $\mathbf{B}$ some process does not halt until round $M_{ac} + 2$; and, in the case of general omission failures, that some process does not even decide until round $M_{ac} + 2$ (and if all do decide in round $M_{ac} + 2$ then $n > 2t$). The proof requires certain definitions.

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., the italicized part of SD3 is elided). Finally, a FD protocol is *strongly sender-dependent* if it is sender-dependent and, in addition, run r contains at most $t-1$ faulty processes. In Hadzilacos and Halpern [1991] we show (cf. Theorem 3(b), (c) and (d)),

Claim 5.8.1: *Let* $\mathbf{D}$ *be a weakly sender-dependent (resp. sender-dependent) FD protocol for sending (resp. general) omission failures and* $\mathbf{B}$ *be a BA protocol for the same type of failures that extends* $\mathbf{D}$*, so that the only messages sent in the f.f. runs of* $\mathbf{B}$ *are those sent by* $\mathbf{D}$*. If all processes halt by round* M *in the f.f. runs of* $\mathbf{D}$ *then there is some f.f. run of* $\mathbf{B}$ *in which a process halts (resp. decides) no earlier than round* $M+2$*. In the case of general omission failures, if* $\mathbf{D}$ *is strongly sender-dependent, all processes decide in round* $M+2$*, and* $n \geq 4$*, then we must have* $n > 2t$*.*

Claim 5.8.2: *If* $t > 1$ *then any FD protocol for sending (resp. general) omission failures that uses* $n+t-1$ *messages in* H_v *and no messages in* $H_{\overline{v}}$ *is weakly sender-dependent (resp. strongly sender-dependent).*

Proof of Claim 5.8.2: The sender sends messages to only one process in H_v. To see this assume, by way of contradiction, that the sender sends messages to two distinct processes. Thus, by Claims 5.4.8 and 5.4.12, there are two pseudochains from the sender to q_0, and their longest common prefix would contain just the sender, contrary to Claim 5.4.14. Let p be the unique process to which the sender sends messages in H_v. Let r be the run where the sender has initial value v and the only faulty process is p, which omits to send any messages. It is easy to see that no correct process will receive any message in r, so all correct receivers cannot distinguish r from $H_{\overline{v}}$ and will therefore decide $\overline{v}$ in r. Since, by Claim 5.4.13, the sender is in B and it will not receive any message in r, the sender will discover a failure in r. Let r' be the run where the only faulty process is the sender which has initial value v, but omits to send any messages. Again, it is easy to see that no process will receive any message in r', so all correct receivers will decide $\overline{v}$. As in r, the sender expects to receive a message (since it is in B and its initial value is v); not receiving one, it discovers a failure. Thus, runs r and r' satisfy properties SD1–SD3, so $\mathbf{D}$ is weakly sender-dependent.

If $\mathbf{D}$ tolerates *general* omission failures, we define run r slightly differently. Instead of p omitting to send any messages, it omits to receive any messages sent to it by the sender. In this way, now not even p can distinguish r and r' (whereas before it could, since it received messages in one but not in the other), and the stronger version of SD3 actually holds, proving that, in this case, $\mathbf{D}$ is sender-dependent. Indeed, since $t > 1$ and r has only one faulty process, $\mathbf{D}$ is strongly sender-dependent. ▮ Claim 5.8.2

The proof now follows the same lines as that of Theorem 4.9. Suppose, by way of contradiction, that $t > 1$, the protocol **B** of our theorem tolerates sending (resp. general) omission failures, and in the f.f. runs of **B** processes halt (resp. decide) by the end of round $M_{ac}+1$. By Claim 4.9.1 (which was proved in the context of crash failures, and thus holds *a fortiori* if we have sending or general omission failures), all processes can distinguish the f.f. runs of **B** at the end of round M_{ac}. Consider the FD protocol **D** obtained from **B** by truncating it at round M_{ac}. More specifically, at the end of round M_{ac} of some run of **B**, each process p halts as follows: If it cannot distinguish the run from H_v or $H_{\overline{v}}$ it decides v or $\overline{v}$, respectively; otherwise, it discovers a failure. From Claim 4.9.1 it follows that **D** is a FD protocol for sending (resp. general) omission failures. **D** uses at most $n+t-1$ messages in H_v and no messages in $H_{\overline{v}}$ (because so does **B**), and processes halt in round M_{ac} in its f.f. runs. By Claim 5.8.2, **D** is weakly (resp. strongly) sender-dependent. In addition, **D** can be extended to a BA protocol, namely **B**, for sending (resp. general) omission failures which has the same number of messages in the f.f. runs, and in which processes halt (resp. decide) in round $M_{ac}+1$ in these runs. This contradicts Claim 5.8.1. If **B** tolerates general omission failures, all processes decide by round $M_{ac}+2$ in the f.f. runs, and $n \geq 4$, then, by Claim 5.8.1, we must have $n > 2t$. ∎

6. Arbitrary failures (without authentication)

We first prove that any t-resilient BA protocol that tolerates arbitrary failures must have a total of at least $n(t+1)/2$ messages in the two f.f. runs. This result was also proved by Dolev and Reischuk [1985]. We give a slightly different proof which makes it easy to see how we arrive at the family of protocols that achieves this lower bound.

Theorem 6.1: *The total number of messages sent in the f.f. runs of any FD protocol that tolerates arbitrary failures is at least $\lceil n(t+1)/2 \rceil$.*

Proof: Consider any FD protocol and construct a digraph $\vec{G}$ with nodes corresponding to the processes and an edge (p,q) iff p sends a message to q in either one of the two f.f. runs of the protocol. Let G be the (undirected) graph resulting when we drop the direction of $\vec{G}$'s edges. We claim that G is $(t+1)$-connected. Suppose not and let A be a set of t nodes whose removal disconnects the graph, and G_0, G_1 be two disconnected graphs that make up $G \setminus A$. Now consider a run r in which only the processes in A are faulty. In r, the processes in G_0 and G_1 behave as in H_0 and H_1, respectively. The processes in A (which are the faulty ones) behave towards the nodes in G_0 as in H_0; and towards the nodes in G_1 as in H_1. (They have the power to do this because we are dealing with arbitrary failures, so the behaviour of faulty processes is not constrained in any way.) Thus, to the processes in G_v, r is indistinguishable from H_v, for both $v \in \{0,1\}$. Hence, the processes in G_0 will decide 0 and those in G_1 will decide 1. Since there are only t faulty processes in r and Weak Agreement is violated, we have a contradiction.

Since G is $(t+1)$-connected, each node has degree at least $t+1$. Therefore there are at least $\lceil n(t+1)/2 \rceil$ edges in G. (We can associate $t+1$ edges with each node, but then we are counting each edge twice.) Hence, a total of at least $\lceil n(t+1)/2 \rceil$ messages are sent in the two f.f. runs. ∎

Corollary 6.2: *In the case of arbitrary failures, the worst-case f.f. message complexity is at least* $\lceil n(t+1)/4 \rceil$, *and the average-case f.f. message complexity is at least* $\min(P_0, P_1) \cdot \lceil n(t+1)/2 \rceil$.

We now describe a family of protocols for BA (actually FD, but this suffices, given Theorem 2.1) that tolerate up to t arbitrary failures, based on a graph-theoretic construction. Particular instances of this family are worst-case optimal and average-case optimal.

Consider an *undirected* graph G on n nodes which is $(t+1)$-connected. Arbitrarily choose a correspondence between nodes and processes. Now assign "round" labels and directions to all the edges in any way consistent with the rules below. Intuitively, the edges will correspond to messages sent in the f.f. runs. The direction will indicate who will be the sender and who the receiver of the message, and the round label will indicate in which round the message will be sent. The rules are:

a. Some (≥ 1) of the edges incident to the sender are labeled with positive natural numbers and directed away from the sender.

b. An edge incident to a node p other than the sender can be labeled with any number greater than k and directed away from p if there is an edge that is already labeled with k and directed into p.

It is certainly possible to assign directions and round labels consistently with these rules: For example, a breadth-first search of the graph started at the node appointed as the sender will accomplish this while minimizing the round labels assigned.

Let $\vec{G}$ be a digraph resulting from the above process. If there is an edge from p to q in $\vec{G}$, that means that process p sends to process q in one of the two f.f. runs. The question is, in which f.f. run should it send? It turns out not to matter. We assign to each edge in $\vec{G}$ an "initial value" label, which is either 0 or 1. The assignment of such labels to edges is completely arbitrary. In this way, we can view each edge of $\vec{G}$ as being labeled with a pair (v, k) where v is the initial value label and k is the round label. This labeled digraph determines a protocol in the following way. In the f.f. runs, the processes send messages according to the labeling. That is, if there is an edge labeled (v, k) from p to q in the graph, then p sends a message to q in round k of H_v. The graph lets us compute, for each process p, what its view ought to be in round k of both H_0 and H_1.

At the end of round M, where M is the maximum round label of any edge, each process p concludes the failure discovery protocol as follows:

If p's view at the end of round M is the same as its view in H_v then it decides v, for $v \in \{0, 1\}$. Otherwise, p discovers a failure.

It should be noted that this rule would be ambiguous if it was possible for a process to have the same view in both H_0 and H_1. However, this can't happen. This is obvious for the sender, since its view includes the initial value. For a receiver p, we first show that it must have an incoming edge in $\vec{G}$. To see this, recall that G is connected and therefore p must have either an incoming or an outgoing edge in $\vec{G}$. If it is incoming, we are done. If it is outgoing, let k be the round label of the edge. Since p is not the sender, the edge must have been directed and labeled by rule (b). But then p must have an incoming edge (labeled less than k), as wanted. Since each receiver has an incoming edge in $\vec{G}$, there is a message that it receives in one of the f.f. runs but not in the other. Therefore, it cannot have the same view in both f.f. runs.

Different instances of this family of protocols are obtained by using different labelings consistent with the rules described above. We next give the proof that the protocol (or, more accurately, any instance of the protocol family), is correct.

Theorem 6.3: *The above protocol is a t-resilient Failure Discovery protocol for arbitrary failures.*

Proof: It is easy to check that Weak Termination is satisfied. Regarding Weak Agreement, consider any run r of the protocol in which no more than t processes are faulty and no correct process discovers a failure. It remains to show that no two correct processes reach conflicting decisions in r. Suppose, by way of contradiction, that p' and q' are correct in r and decide v and $\overline{v}$, respectively, for some $v \in \{0,1\}$. Since G is $(t+1)$-connected, by Menger's theorem, there are at least $t+1$ node-disjoint paths between p' and q'. Since there are at most t faulty processes, some path joining p' and q' in G consists entirely of correct processes. By the assumption that no correct processes discover a failure in r and the fact that Weak Termination holds, each process along that path must reach a decision. Since p' and q' reach conflicting decisions, there must be two correct processes, say p and q, on this path that are joined by an edge in G and decide v and $\overline{v}$, respectively. By the decision rule of the protocol, p's view in r is the same as in H_v and q's view in r is the same as in $H_{\overline{v}}$. Without loss of generality, assume that the edge connecting p and q is directed from p to q. If this edge is labeled (v,k), p sends a message to q in round k of H_v but not of $H_{\overline{v}}$. Therefore p sends that message to q in round k in r as well. But then q receives a message in r that it does not receive in $H_{\overline{v}}$ contradicting the assumption that q's view in r is the same as in $H_{\overline{v}}$. A similar contradiction is obtained if the edge from p to q is labeled $(\overline{v},r)$. Thus, p and q, and hence p' and q', do not reach conflicting decisions. This shows that Weak Agreement holds. Weak Validity follows from Weak Agreement and the fact that the sender, if it is correct, decides on its initial value. ∎

As we have seen, the number of edges in G corresponds to the total number of messages in the two f.f. runs of the protocol. To make the protocol message efficient we want G to be a $(t+1)$-connected graph on n nodes with the minimum possible number of edges. Since the degree of each node in a $(t+1)$-connected graph is at least $t+1$, at least $\lceil n(t+1)/2\rceil$ edges are necessary for this, and we can actually achieve this bound. The construction on which this is based is known in graph theory (see Bollobas [1978], Theorem 1.6, p. 6), so we omit details here. (The simple case is when $t+1$ is even, in which case we construct the graph as follows: put the n nodes along the circumference of a circle and connect each node to the $(t+1)/2$ clockwise neighbours and the $(t+1)/2$ counter-clockwise neighbours. The general case is similar.)

To obtain a worst-case optimal protocol, we assign initial value label 0 to half of the edges and initial value label 1 to the other half. To obtain an average-case optimal protocol, we assign initial value label $\overline{v}$ to all edges, where v is the more likely initial value. Again, note that the protocols use only one-bit messages in the f.f. runs, so our protocols are optimal in terms of both bits and messages.

It is not hard to show that we can obtain an FD protocol that is message optimal and runs in $\lceil n/(t+1) \rceil$ rounds using this construction. We do not know if we can do better, although we conjecture that we cannot. By Theorem 2.1, we can convert this to a BA protocol with optimal f.f. message complexity at a cost of $t+1$ rounds; again, we do not know if we can do better.

7. Arbitrary failures with message authentication

The assumption that processes have access to message authentication, i.e., a scheme that implements unforgeable signatures, restricts the range of possible behaviours of faulty processes and hence their ability to "confuse" the correct processes. To see this, consider three processes p, q and f, where p and q are correct but f is faulty. Suppose that at some point, f wishes to send a message to q informing q that f has received a message m from p. If f can behave completely arbitrarily, it can do this even if it had never received such a message. Such behaviour could be prevented if message authentication was available: p would sign all messages it sent to f; and q would require f to produce a copy of m duly signed by p before trusting f's claim to have received such a message. Thus, message authentication prevents faulty processes from lying about messages they have received from correct processes (on pain of being discovered as faulty). Note, however, that message authentication does not prevent *all* forms of "lying". For example, a faulty *sender* can perfectly well affix its signature to two messages sent to different processes, one claiming that the initial value is 0 and the other claiming that the initial value is 1.

It is clear that the lower bound of $n+t-1$ for the total message complexity in the crash failure case immediately applies in the case of arbitrary failures, even with authentication. Somewhat surprisingly, it is in fact a tight bound on the total message complexity. In particular, there is a protocol that has no messages in one run and $n+t-1$ messages in the other; this is therefore an average-case optimal protocol. In contrast to all other failure types we have considered, however, it is not possible to construct other correct protocols by transferring messages from one f.f. run to the other on a one-to-one basis. In fact, we can show that the lower bound on the worst-case f.f. message complexity is $\lceil (n+2t-2)/2 \rceil$, rather than the expected $\lceil (n+t-1)/2 \rceil$, provided that $n \geq 8t-2$. Thus, arbitrary failures with message authentication represent an anomaly in the pattern we have observed in the other failure types.

Let us first look at a protocol that uses no messages in one f.f. run and $n+t-1$ messages in the other. The protocol, called AUTH2, is a straightforward adaptation of the average-case optimal instance of $\mathrm{GOF2}_v$ (cf. Figure 3, with $B^1 = \emptyset$, $|B_v^2| = t-1$ and $|R_v| = n-t$). The only difference is that a process relays a message to the next process down some chain *after signing it*. Thus, the message sent in round k of run H_v contains the value v and the signatures of the first k processes in the order they appear in Figure 3. Each process that receives a message verifies that the message is *valid* — i.e., that it contains all the required signatures and that all these signatures are valid. If so, it adds its own signature to the message and passes it down the chain; if not, it ignores that message and does not forward anything to the next process down the chain. Processes decide at the end of round n, using the same decision rule as in $\mathrm{GOF2}_v$ (except that now a process must check that all messages received are valid; if not, it discovers a failure).

The proof that AUTH2 solves the Failure Discovery problem is much the same as that showing that $GOF2_v$ solves the Uniform Failure Discovery problem. We cannot guarantee Uniform Agreement though, since we now allow faulty processes to behave arbitrarily. We leave details to the reader. Thus we get

Theorem 7.1: *AUTH2 is a Failure Discovery protocol for arbitrary failures with total f.f. message complexity of* $n+t-1$ *and average-case f.f. message complexity of* $\min(P_0, P_1)\cdot(n+t-1)$.

One might hope that a similar adaptation of GOF1 would work for arbitrary failures with message authentication. Unfortunately, this is not so. To see this consider the following scenario (cf. Figure 2): The sender is faulty and sends 0 (appropriately signed) to the first process in R_0 and 1 (also appropriately signed) to the first process in R_1. This sort of faulty behaviour is allowed, as we have seen. Now the message carrying the value 0 propagates through the chain of processes in R_0 (accumulating signatures as it goes) and all of these processes decide 0. Similarly, the message carrying the value 1 propagates through the chain of processes in R_1, which decide 1. Now suppose the last process in R_1 is faulty and does not broadcast the message containing 1 to the processes in B, but the last process in R_0 is correct. Thus, only the message containing 0 (and the accumulated signatures) will reach the processes in B. If all of these processes are correct they will all decide 0 and none will discover a failure. Thus, we have a run with just two faulty processes in which agreement is foiled.

There is a way of fixing this problem, but it requires introducing $t-1$ more messages. The problem with the protocol we just found to be incorrect lies in that R_0 and R_1 naively trust the value in a message that is only signed by the sender. The solution is to force the value v to get the endorsement of t processes that receive messages in both runs, before a process in R_v "trusts" a message that carries v and decides that value. The protocol, called AUTH1, is illustrated in Figure 5. The processes (including the sender) are partitioned into four sets B^-, B^+, R_0 and R_1, so that the sender is in B^- and $|B^-| = |B^+| = t$. The processes in B^-, R_0 and R_1 are arranged in linear chains, where the sender is the first process in the B^--chain. The sender sends its initial value v to the next process in the B^--chain. The message passes through the B^-- and R_v-chains accumulating signatures in each round. The last process in R_v signs and sends it to all processes in B^+. As in AUTH2, a process receiving an invalid message (i.e., one which is missing some of the required signatures or contains invalid signatures) will ignore it and not pass it on.

At the end of round $\max(|R_0|, |R_1|) + t$,

- the sender decides on its initial value
- a process in R_v decides v if it received a message v; otherwise it decides $\overline{v}$
- a process in B^+ or B^- decides v if it received a message v; otherwise it discovers a failure.

It is easy to verify that AUTH1 satisfies Weak Termination and Weak Validity. To see that it also satisfies Weak Agreement suppose that there is a run in which two correct

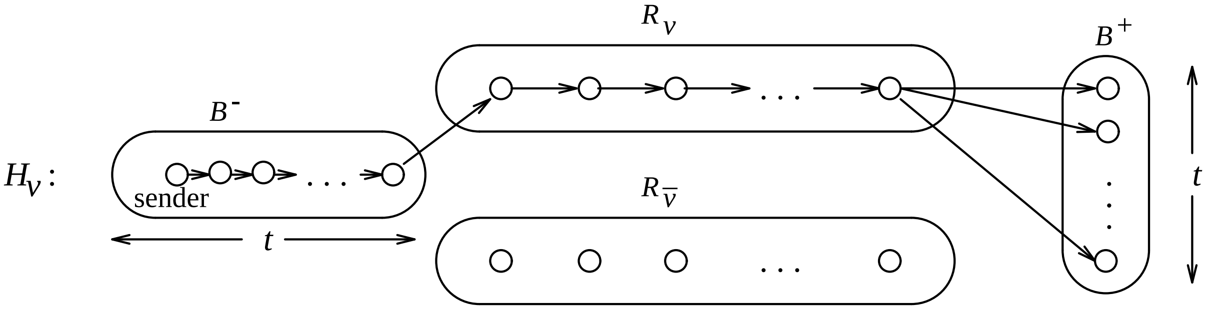


Figure 5: Protocol AUTH1

processes, p and q, decide 0 and 1 respectively. We shall show that some other correct process must discover a failure in that run. We proceed by cases on which group p and q are in.

First suppose that p is either in B^+ or R_0. Notice that q cannot be in B^- for then the message whose receipt caused p to decide 0 would have to contain q's signature to be valid. However, q only signed a message with value 1 (otherwise it would not have decided 1) and this would contradict the assumption that correct processes' signatures cannot be forged. Therefore, q must be in R_0, R_1 or B^+. If q is in R_1 or B^+ then all processes in B^- must be faulty, since they must have all signed a message containing value 0 (the one that causes p to decide 0) and a message containing value 1 (the one that causes q to decide 1). Since $|B^-| = t$, all other processes must be correct. Therefore, all processes in B^+ (which are correct) will receive messages with both initial values and will discover a failure, as wanted. If q is in R_0, then q did not receive a message. This means that q must follow p in the R_0 chain and that, further, some process between p and q in that chain is faulty. Since $|B^-| = |B^+| = t$, there exist correct processes in both B^- and B^+. The fact that not all processes in B^- are faulty and that p receives a message containing 0 and signed by (at least) all processes in B^- implies that no process can receive a message containing 1 and signed by (at least) all processes in B^-. Thus, the correct process in B^+ cannot decide 1. It also cannot decide 0, for this would require the receipt of a message containing 0 and signed, among others, by q, which is impossible since q is correct and never signs a message. Thus, the correct process in B^+ will discover a failure, as wanted. We have therefore proved Weak Agreement is satisfied if p is in B^+ or R_0. By symmetric arguments we are also done if q is in B^+ or R_1.

Next suppose that p is in R_1. Hence, p does not receive a message. We have already shown that q cannot be in B^+ or R_1. If q is in R_0, it too does not receive a message. That means that there must be a faulty process preceding p or one preceding q. Thus, some process in B^+ is correct. Furthermore, a correct process in B^+ cannot decide because doing so requires the receipt of a message signed, among others, by either p or q, neither of which, however, signs a message. Thus, the correct process in B^+ will discover a failure, as wanted. If q is in B^- then q must precede p in the message chain of H_1 and some process between them must be faulty. Arguments similar to those used above show that a correct process in B^+ will discover a failure. This shows that if p is in R_1 then Weak Agreement is satisfied. A symmetric argument shows that if q is in R_0 we are done as well.

The only remaining possibility, therefore, is for both p and q to be in B^-. But this is impossible, because the first of the two in the chain must receive messages containing both 0 and 1 and will therefore discover a failure (rather than decide 0 or 1, as we assumed). This completes the proof that AUTH1 is a correct FD protocol. The number of messages in run H_v of AUTH1 is $2t + |R_v| - 1$. To minimize the maximum number of messages in the f.f. runs we must divide the $n - 2t$ processes that are not in B^- or B^+ equally between R_0 and R_1. This yields a worst-case f.f. message complexity of $\lceil (n + 2t - 2)/2 \rceil$ for the protocol.

We can do a little better if t is large relative to n. Consider the trivial protocol where we order all the receivers. The sender signs and sends its value to the first receiver, which forwards it to the second receiver after signing it, and so on up to the t-th receiver, which broadcasts it to the remaining processes. This protocol is easily seen to be correct, and uses $n - 1$ messages in each of the f.f. runs. If $n \leq 2t$, then this is better than $\lceil (n + 2t - 2)/2 \rceil$. Thus we get

Theorem 7.2: *There is a Failure Discovery protocol for arbitrary failures with authentication that has worst-case message complexity* $\min(n - 1, \lceil (n + 2t - 2)/2 \rceil)$.

AUTH1 uses a total of $n + 2t - 2$ messages in the f.f. runs, which is $t - 1$ more than the optimum attained by AUTH2. This raises the question: Can we do better than AUTH1 as far as the *worst-case* f.f. message complexity is concerned? The following theorem says that the answer is no, at least if n is sufficiently larger than t.

Theorem 7.3: *If* $n \geq 8t - 2$ *then the worst-case f.f. message complexity of any FD protocol for arbitrary failures with message authentication is at least* $\lceil (n + 2t - 2)/2 \rceil$.

Proof: We show here that the bound holds for a restricted set of protocols, where, for $v = 0, 1$, we have (1) no process other than the sender sends a message in either H_v without having first received a message and (2) a process in R_v sends at most one message to other processes in R_v. We consider this special case both because it illustrates the main ideas of the result, and because the restrictions apply to all the protocols we have considered thus far, so that this proof already shows that we would need a different style of algorithm in order to hope to beat the bound. In Appendix B, we give a considerably more involved argument which proves that the theorem actually holds in general, without any restrictions on the form of the protocol.

We first need a technical lemma.

Lemma 7.4: *In any FD protocol for arbitrary failures with message authentication, if a process* p *receives the signatures of* $t - k$ *processes by round* ℓ *in* H_v *then either (a)* p *receives a message after round* ℓ *in* H_v*, (b)* p *sends at least* $k + 1$ *messages after round* ℓ *in* H_v*, or (c)* p *sends a message after round* ℓ *in* $H_{\overline{v}}$.

Proof of Lemma 7.4: Suppose, by way of contradiction, that p receives the signatures of $t - k$ processes by round ℓ in H_v, receives no further messages in H_v, sends no messages after round ℓ in $H_{\overline{v}}$, and sends at most k messages after round ℓ in H_v. Now consider a run r in which the only faulty processes are the $t - k$ whose signatures p receives in H_v

and the at most k processes which (directly) receive a message from p in H_v after round ℓ. (Note that the total number of faulty processes in r is at most t.)

In run r, the $t-k$ faulty processes whose signatures p receives in H_v "conspire" to present to p exactly the view p has in H_v. (They can do this because, being all faulty, they can forge each other's signature.) The at most k processes that receive a message from p in H_v "pretend" not to receive any such message and behave as they would in $H_{\overline{v}}$. All correct processes other than p behave as in $H_{\overline{v}}$, while p behaves as in H_v. A straightforward induction on the round number i shows that:

- the processes that are faulty in r up to round i are a subset of those whose signatures p receives in H_v and those that receive a message from p in H_v,
- p's view of r through round i is identical to its view of H_v through that round, and
- the view that every process other than p and the faulty ones has of run r through round i is identical to its view of $H_{\overline{v}}$ through that round.

Therefore run r is a run of the protocol where at most t processes are faulty, p decides v while all other correct processes decide $\overline{v}$. Since $n \geq t+2$, there is some correct process other than p in r, contradicting Weak Agreement. ∎ Lemma 7.4

The total number of messages received in the two f.f. runs is at least $|R_0|+|R_1|+2(n-|R_0|-|R_1|-1) = 2n-|R_0|-|R_1|-2$. If $|R_0|+|R_1| < n-2t$, then the total number of messages sent is at least $n+2t-2$, and the result immediately follows. Thus, we can assume without loss of generality that $|R_0|+|R_1| \geq n-2t$. Moreover, we can also assume that R_0 and R_1 both have at least two processes, for if $|R_v| \leq 1$, then at least $|R_{\overline{v}}| \geq n-2t-1 \geq \lceil (n+2t-2)/2 \rceil$ messages are sent in $H_{\overline{v}}$. (For the last inequality, we use the fact that $n \geq 8t-2$, although in fact $n \geq 6t$ would suffice for the restricted version of the theorem which we are proving here.)

Let p_0 be the first process in R_v to receive a message in H_v, and let p_1 be the last process in R_v to receive a message in H_v. (If all processes in R_v receive a message in the same round, then it suffices to take $p_0 \neq p_1$; this can be done since $|R_v| \geq 2$.) Suppose p_0 receives its message in round ℓ_0 and p_1 receives its message in round ℓ_1. From Lemma 4.1(1) and the fact that, by assumption (1), p_1 does not send any messages in $H_{\overline{v}}$, it follows that t messages are received that are in an chain starting with p_1 after round ℓ_1 in H_v. Since p_1 was the last process in R_v to receive a message, none of these are received by a process in R_v. Suppose p_0 receives the signatures of $t-k$ messages. From Lemma 7.4, since p_0 does not send any messages in $H_{\overline{v}}$ (by assumption (1)), p_0 must send messages to $k+1$ processes, of which at most one is in R_v (by assumption (2)). In addition, each of the $t-k$ signatories on the message that p_0 receives (p_0 receives only one message since it is in R_v) must have sent a message that is received by round ℓ_0. Since at most one of these messages is received by p_0, we have identified another $t-k-1$ messages not received by a member of R_v. Altogether, we have identified $2t-1$ messages sent in H_v received by processes other than those in R_v. Thus, in H_0 and H_1, there are at least $4t-2$ messages not received by a process in $R_0 \cup R_1$. Since there are at least $n-2t$ processes in $R_0 \cup R_1$, and each receives exactly one message in the two f.f. runs, this means that a total of at least $n+2t-2$ messages are sent in the two f.f. runs. The result now follows. ∎

It is interesting to note that in the proof of this theorem we made use of Lemmas 4.1(1) and 7.4. The former holds even for protocols that tolerate only general omission failures. However, the proof of Lemma 7.4 uses, in a critical manner, the fact that failures are allowed to be arbitrary (except for forging signatures of correct processes). In particular, faulty processes must be able to "conspire" in order to present a certain view to a correct process, a behaviour impossible under omission failures.

We conjecture that, except in the special case where $n = 5$ and $t = 2$, $\min(n-1, \lceil (n+2t-2)/2 \rceil$ is in fact a tight bound on the worst-case f.f. message complexity. In this special case, as we now show, we can actually achieve a worst-case f.f. message complexity of 3. The protocol, illustrated in Figure 6, is quite simple: Suppose that besides the sender s, we have four receivers a, b, c, d. We take $R_0 = \{a\}$ and $R_1 = \{b\}$. If the sender has initial value 0 (resp. 1), it signs and sends 0 to a (resp. signs and sends 1 to b), which signs and sends this message to both c and d. It is easy to show that this is a correct FD protocol: If s is correct, no process can be fooled, no matter how many faulty processes there are. If s and another process is faulty, a straightforward case-by-case analysis shows that no other process can be fooled; we leave details to the reader.

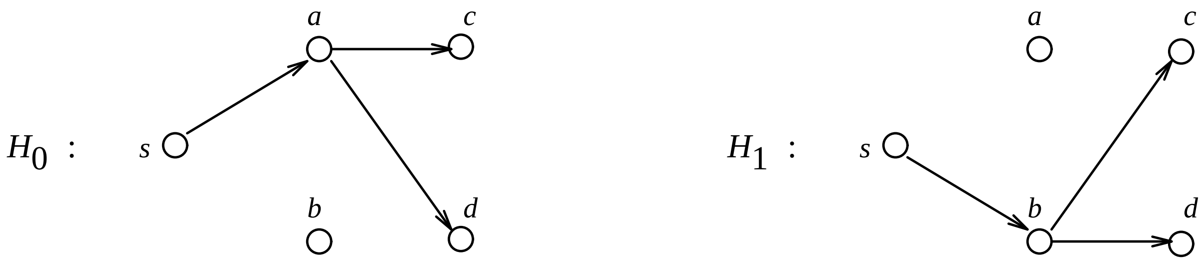


Figure 6: Protocol for $n = 5$, $t = 2$ that beats the bound of Theorem 7.3

8. Non-binary agreement

Most of our results have natural analogues if there are more than two initial values. We briefly discuss the situation here, omitting many details.

Let V be the set of possible initial values. Without loss of generality, we can take $V = \{0, \ldots, |V|-1\}$. Thus, each protocol now has $|V|$ f.f. runs, one for each possible initial value. Given a fixed protocol, for any $v \in V$ we shall let H_v denote the f.f. run of the protocol with initial value v, and $\overline{R}_v$ denote the set of receivers which receive no message in H_v and receive exactly one message in every other f.f. run; in other words, $\overline{R}_v$ consists of the receivers which receive exactly $|V| - 1$ messages in the f.f. runs. (In case $V = \{0, 1\}$, $\overline{R}_v$ is precisely $R_{\overline{v}}$, as defined at the end of Section 1.)

Receiving omission failures: Clearly, every receiver must get at least a total of $|V| - 1$ messages in the f.f. runs, one in every run except possibly one (otherwise a receiver would receive no messages in two different f.f. runs, and would have to violate Validity in one of the two). Thus, we get a lower bound of $(|V|-1)(n-1)$ for the total f.f. message complexity and, therefore, of $\lceil (|V| - 1)(n - 1)/|V| \rceil$ for the worst-case f.f. message complexity. These bounds are tight in the case of receiving omission failures, as the following simple protocol shows. We partition the $n-1$ receivers into $|V|$ sets $\overline{R}_v$, $v \in V$. If the sender's initial value is v, it sends v to all the processes in $\cup_{v' \neq v} \overline{R}_{v'}$. A receiver decides v if it receives v from the sender in round 1 or if it receives no message and is in $\overline{R}_v$. Clearly, any such protocol achieves the lower bound on the total number of messages and, by taking the $\overline{R}_v$'s of roughly equal size, we achieve the lower bound on the worst-case number of messages.

Observe that the protocols described above have total f.f. bit complexity of $O\big(\log(|V|) \cdot (|V|-1)(n-1)\big)$: A different message is sent in each of the f.f. runs (except possibly one), so each one must contains roughly $\log(|V|)$ bits. We can improve the bit complexity to $(|V|-1)(n-1)$, that is, make it equal to the message complexity, by the following trick: in H_v, the sender sends a single bit to all the processes in $\cup_{v' \neq v_i} \overline{R}_{v'}$ in round $v+1$. The processes decide in round $|V|$ in the obvious way. Clearly there is a tradeoff here between bit complexity and number of rounds required to reach agreement. We can improve the round complexity by a factor of two without increasing the bit complexity, using both 0 and 1 to encode the value of V; no further improvement seems possible.

For average-case complexity, let P_v be the probability of v being the initial value. For the remainder of this section, we let v_{max} denote a particular element in V such that $P_{v_{max}} \geq P_v$ for all $v \in V$. Since every receiver must receive a message in every run but one, we clearly optimize the average-case complexity by making $H_{v_{max}}$ the one run where it does not receive a message. Thus, by using the simple protocol described above where $\overline{R}_{v_{max}}$ consists of all the receivers and $\overline{R}_{v'} = \emptyset$ for $v' \neq v_{max}$, we get the optimal average-case complexity of $(1-P_{v_{max}})(n-1)$.

Crash, sending omission, and general omission failures: Moving to crash failures, observe that we can generalize Theorem 4.2 as follows:

Theorem 8.1: *Let $W \subseteq V$ with $|W| \geq 2$. The total number of messages sent in the f.f. runs $\{H_w : w \in W\}$ in any BA protocol for crash failures is at least $(|W|-1)(n-1)+t$.* ∎

Essentially, the idea is that we have sets of size $n-t-1$ that get $|W|-1$ messages (one in all but one of the runs H_w, $w \in W$) and a witness set of size t that gets $|W|$ messages. The proof of this result is very similar to the proof of Theorem 4.2, and is based on a suitable generalization of Lemma 4.1(3), so is omitted here.

Corollary 8.2: *The worst-case f.f. message complexity of any FD protocol for crash failures is at least $\big\lceil \big((|V|-1)(n-1)+t\big)/|V| \big\rceil$.* ∎

The analysis for the lower bound for average-case complexity is a little more delicate. From Theorem 8.1, we can see that even if we arrange (by a suitable extension of GOF2$_v$, for example) to have no messages at all in the f.f. run with the most likely initial value, this can only be accomplished at the cost of having at least $n+t-1$ messages in *all* other f.f. runs. Intuitively, this is the optimal strategy if the probability of the most likely initial value is sufficiently greater than the probability of all other values. It will not be optimal if the probabilities of all values are roughly equal.

To study the situation more formally, recall that v_{max} is such that $P_{v_{max}} \geq P_v$ for all $v \in V$. We clearly want to send the least number of messages in the runs with highest probability. Let m_v be a variable whose possible values are natural numbers. Intuitively, m_v stands for the number of messages sent in H_v. To derive lower bounds on the average-case f.f. message complexity we can formulate an integer linear programming problem. We wish to minimize

$$\sum_{v \in V} P_v \cdot m_v \tag{1}$$

subject to the following constraints:

$$m_v \geq 0 \qquad \text{for all } v \in V \tag{2}$$

$$\sum_{w \in W} m_w \geq (|W| - 1)(n - 1) + t \qquad \text{for all } W \subseteq V \text{ with } |W| \geq 2 \tag{3}$$

Constraint (3) follows from Theorem 8.1.

Fortunately, this problem is easy to solve in this case. Since v_{max} is the most likely value, a straightforward symmetry argument shows that in order to minimize (1), we can assume without loss of generality that $m_{v_{max}} \leq m_v$ for all $v \in V$. Also, as we now prove, to minimize (1), we can assume that $m_{v_{max}} \leq t$. To see this, observe that if we take $m_{v_{max}} = t$ and $m_v = n - 1$ for $v \neq v_{max}$, then this choice satisfies all the constraints and gives us an average-case message complexity of $P_{v_{max}} t + (1 - P_{v_{max}})(n - 1)$. Now suppose we choose m_v's for all $v \in V$ that satisfy the constraints (2) and (3), and assume that $m_{v_{max}} > t$. Let $k = m_{v_{max}} - t$ (so, by our assumption, $k > 0$) and $k_v = n - 1 - m_v$ for all $v \in V \setminus \{v_{max}\}$. By Theorem 8.1, if $W' \neq \emptyset$, $v_{max} \notin W'$ and $W = W' \cup \{v_{max}\}$, then

$$\begin{aligned}(|W| - 1)(n - 1) + t \leq \sum_{w \in W} m_w &= (k + t) + \sum_{w \in W'} (n - 1 - k_w) \\ &= (k + t) + (|W| - 1)(n - 1) - \sum_{w \in W'} k_w\end{aligned}$$

and therefore

$$k \geq \sum_{w \in W'} k_w. \tag{4}$$

We next claim that

$$P_{v_{max}} k \geq \sum_{v \neq v_{max}} P_v k_v. \tag{5}$$

To see this, let $W' = \{w \neq v_{max} | k_w \geq 0\}$. We may assume that $W' \neq \emptyset$ for, if $k_v < 0$ for all $v \neq v_{max}$ then we can take $k_v = 0$ for all $v \neq v_{max}$ without violating the constraints and doing at least as well in minimizing (1) as with $k_v < 0$. Clearly

$$\sum_{v \neq v_{max}} P_v k_v \leq \sum_{w \in W'} P_w k_w \leq P_{v_{max}} \sum_{w \in W'} k_w \leq P_{v_{max}} k$$

where the last inequality is due to (4). Putting all this together, we get

$$\begin{aligned}
\sum_{v \in V} P_v \cdot m_v &= P_{v_{max}} \cdot m_{v_{max}} + \sum_{v \neq v_{max}} P_v \cdot m_v \\
&= P_{v_{max}}(t+k) + \sum_{v \neq v_{max}} P_v(n-1-k_v) \\
&= P_{v_{max}} t + \sum_{v \neq v_{max}} P_v(n-1) + P_{v_{max}} k - \sum_{v \neq v_{max}} P_v k_v \\
&\geq P_{v_{max}} t + (1 - P_{v_{max}})(n-1)
\end{aligned}$$

where the last step follows by (5). This shows that we do at least as well by taking $m_{v_{max}} = t$ as by taking $m_{v_{max}} > t$.

Thus, for the remainder of the proof, we assume $m_{v_{max}} \leq t$. By Theorem 8.1 (taking $W = \{v, v_{max}\}$), we have that $m_v \geq n+t-1-m_{v_{max}}$ for all $v \neq v_{max}$. It is straightforward to check that taking $m_v = n+t-1-m_{v_{max}}$ for $v \neq v_{max}$ satisfies every instance of constraint (3).

Plugging these values for m_v, $v \neq v_{max}$ into (1) we get that we must minimize the following expression:

$$P_{v_{max}} \cdot m_{v_{max}} + (1 - P_{v_{max}})(n+t-1-m_{v_{max}})$$

Rearranging this as a function in $m_{v_{max}}$ we get

$$(2P_{v_{max}} - 1)m_{v_{max}} + (1 - P_{v_{max}})(n+t-1). \tag{5}$$

The value of $m_{v_{max}}$ that minimizes (5) depends on the sign of $2P_{v_{max}} - 1$. Specifically, if $2P_{v_{max}} - 1 \geq 0$, i.e., $P_{v_{max}} \geq 1/2$, (5) is minimized when $m_{v_{max}}$ is as small as possible, that is 0. If, on the other hand, $P_{v_{max}} < 1/2$ then (5) is minimized when $m_{v_{max}}$ is as large as possible, which, as we have seen, is t. Summarizing, we have:

Theorem 8.3: *The average-case f.f. message complexity of any FD protocol for crash failures is at least*

$$\begin{array}{rl}
(1 - P_{v_{max}})(n+t-1) & \text{if } P_{v_{max}} \geq 1/2 \\
P_{v_{max}} \cdot t + (1 - P_{v_{max}})(n-1) & \text{if } P_{v_{max}} < 1/2
\end{array} \qquad \blacksquare$$

We can extend the ideas of protocols CF1 and CF2$_v$ to obtain failure discovery protocols that tolerate crash failures and general omission failures and achieve the bounds of Corollary 8.2 and Theorem 8.3. The generalization of CF1 works as follows. We form a set B consisting of t receivers and partition the remaining $n-t-1$ receivers among the $\overline{R}_v$'s. The protocol now works as follows: In f.f. run H_v, the sender sends v to the processes in $\cup_{u \neq v} \overline{R}_u$ in the first round, and sends v to the processes in B in the second round. The decision rule is analogous to that of CF1, as is the proof of correctness: Each process p in

$\overline{R}_u$ decides v if it receives a message containing that message and decides u if it receives no message at all. A process p in B decides v if it receives a message with that value and discovers a failure if it receives no message at all.[4] Notice that the protocol again terminates in two rounds. The analogue of GOF1 is similar, except that the processes in $\cup_{u \neq v} \overline{R}_u$ are arranged in a chain, and the message is sent down the chain.

Straightforward counting shows that the total number of messages used in the f.f. runs of these protocols match the lower bound of Theorem 8.1. Furthermore, by taking all the $\overline{R}_v$'s of roughly equal size (i.e., $\lceil (n-t-1)/|V| \rceil$ or $\lfloor (n-t-1)/|V| \rfloor$ each) and observing that $n-1-\lfloor (n-t-1)/|V| \rfloor + t = \left\lceil \left((|V|-1)(n-1)+t \right)/|V| \right\rceil$, we get a protocol whose worst-case f.f. message complexity matches that of Corollary 8.2.

For average-case complexity, if $P_{v_{max}} \geq 1/2$, we can use a a straightforward extension of CF2 for the crash failure case: we partition the set of receivers into two sets $\overline{B}_{v_{max}}$ and $\overline{R}_{v_{max}}$ where $|\overline{B}_{v_{max}}| = t$ and $\overline{R}_{v_{max}} = n-t-1$. In $H_{v_{max}}$ no messages are sent, while in H_v, for any $v \neq v_{max}$, the sender sends v to all the processes in $\overline{B}_{v_{max}}$ in the first round, sends v to all the processes in $\overline{R}_{v_{max}}$ in the second round, and then sends v to all the processes in $\overline{B}_{v_{max}}$ again in the third round. The decision rule is analogous to that of CF2. In the case of general omission failures, we order the processes in $\overline{B}_{v_{max}}$ and $\overline{R}_{v_{max}}$ into chains, and the message v travels down the chain.

If $P_{v_{max}} < 1/2$, the extensions to CF1 and GOF1 presented above with $\overline{R}_v = \emptyset$, for all $v \neq v_{max}$ give us optimal average-case complexity in the case of crash failures and general omission failures, respectively.

Thus, we have

Theorem 8.4: *The lower bounds of Theorem 8.1, Corollary 8.2, and Theorem 8.3, are all attainable.* ∎

Arbitrary failures (without message authentication): In the case of arbitrary failures (without message authentication), we can extend the ideas in the proof of Theorem 6.1 to get a lower bound on the total number of messages in the f.f. runs. Given any protocol P, we construct a directed multigraph $\vec{G}$ whose nodes are (correspond to) the processes and we put and edge labeled $v \in V$ from p to q if p sends a message to q in H_v. Let G be the undirected multigraph resulting when we drop the direction of $\vec{G}$'s edges. For $v, v' \in V$, let G_v be the subgraph of G consisting only of edges labeled v, and let $G_{v,v'}$ be the subgraph of G consisting only of edges labeled v or v'. The same argument as that used in the proof of Theorem 6.1 shows that the following property must hold:

$$G_{v,v'} \text{ must be } (t+1)\text{-connected, for every pair of distinct } v, v' \in V. \qquad (*)$$

Let $f(|V|, n, t)$ be the minimum number of edges required to construct a graph G on n nodes, with edges labeled by $0, \ldots, |V|-1$ such that (*) holds. We have shown that $f(|V|, n, t)$ is a lower bound on the total message complexity of V-valued Byzantine

[4] This idea was also used by Amdur *et al.* [1990] in connection with a protocol for crash failures.

agreement. This bound can be attained, and in fact the ideas underlying the lower bound proof form the basis of a family of (failure discovery) protocols for arbitrary failures. Suppose we are given a multigraph G whose edges are labeled with values in V and which has property (*). From such a graph we can construct a protocol just as we did for the binary case. We direct the edges and assign round numbers to them according to the rules given in Section 6. The protocol then has process p send message v to process q in round k of H_v iff the directed multigraph has an edge labeled v which is directed from p to q and is assigned round number k. At the end, each process decides v iff its view is the same as in H_v, for some $v \in V$; otherwise it discovers a failure. The proof of correctness is analogous to that of Theorem 6.3.

All that remains now is to compute $f(|V|, n, t)$. Fix a graph G with property (*). Since $G_{v,v'}$ is $(t+1)$-connected, it must have at least $n(t+1)/2$ edges. Thus, the sum of the number of edges in the $G_{v,v'}$ graphs, for all (unordered) pairs of distinct $v, v' \in V$, is at least $|V|n(t+1)/4$. This represents a lower bound on $f(|V|, n, t)$. It is also to easy to see that $f(|V|, n, t)$ is at most $(|V|-1)\lceil n(t+1)/2 \rceil$: Fix a $(t+1)$-connected graph G on n nodes with $\lceil n(t+1)/2 \rceil$ edges, and then label each of the edges with all the values in V except one, say 0. When viewed as a multigraph (with different edges corresponding to each of the $|V|-1$ labels), this clearly satisfies (*) and has the right number of edges. Observe that this construction gives us a protocol with worst-case f.f. message complexity of $n(t+1)/2$.

This is essentially all we can say about bounds for $f(|V|, n, t)$. A beautiful construction due to Alon [1990] shows that if n is prime, $t+1$ is divisible by 4, and $|V|(t+1) \leq 2(n-1)$, then $f(|V|, n, t) = |V|n(t+1)/4$. On the other hand, Alon [1990] has shown that for all n, t, and $\epsilon > 0$, if $|V|$ is sufficiently large, then $f(|V|, n, t) > \big(|V|n(t+1)/(2+\epsilon)\big)$. This means that for fixed n and t, as $|V|$ gets large, the total message complexity approaches $|V|n(t+1)/2$. In addition, as $|V|$ gets large, the worst-case message complexity is $n(t+1)/2$. The proof is based on a counting argument: Let $g(n, t)$ be a bound on the number of graphs on n nodes with less than $\lceil n(t+1)/2 \rceil$ edges. If $|V| > g(n, t)$, then by the pigeon-hole principle, either (a) there is some $v \in V$ such that G_v has at least $\lceil n(t+1)/2 \rceil$ edges, or (b) for some $v, v' \in V$, we have $G_v = G_{v'}$ and G_v has fewer than $\lceil n(t+1)/2 \rceil$ edges. If (b) holds, then $G_{v,v'} = G_v$ has too few edges to be $(t+1)$-connected, thus (a) must hold. But if (a) holds, then at least $\lceil n(t+1)/2 \rceil$ are sent in H_v, giving us the desired lower bound on worst-case f.f. message complexity. Similar arguments give us the desired lower bound on total f.f. message complexity: Let $\alpha = (2+\epsilon)/\epsilon$ and take $|V| > \alpha g(n, t)$. Similar arguments to those above show that G_v has at least $\lceil n(t+1)/2 \rceil$ edges for at least $(\alpha - 1)g(n, t)$ choices of $v \in V$. An easy calculation now yields the desired total message complexity.

Summarizing the preceding discussion we have

Theorem 8.5: *Any BA protocol that tolerates arbitrary failures requires a total of at least $f(|V|, n, t)$ messages and at least $\lceil f(|V|, n, t)/|V| \rceil$ messages in some f.f. run. Furthermore, there is a BA protocol that achieves these bounds. Finally,*

$$\frac{|V|n(t+1)}{4} \leq f(|V|, n, t) \leq \frac{(|V|-1)n(t+1)}{2}$$

(and these bounds cannot be improved). ∎

As was the case with crash, sending omission and general omission failures, the average-case f.f. message complexity depends on the precise relationship of the probabilities of the initial values. Some lower bounds can be obtained by using integer linear programming techniques such as those we examined in detail in the case of crash failures.

Arbitrary failures with authentication: Finally, in the case of arbitrary failures with authentication, we can modify AUTH1 along the same lines as we modified GOF1 above to get a protocol with total f.f. message complexity $(|V|-1)(n-1)+2t-1$ and worst-case f.f. message complexity $\lceil((|V|-1)(n-1)+2t-1)/|V|\rceil$. We believe that this is optimal (at least if n is sufficiently large relative to t), but do not have a proof of this conjecture. Just as with the other types of failures, the optimal average-case complexity will depend on the precise relationship between the probabilities of the initial values, and some lower bounds can be obtained through an integer linear programming formulation of the problem.

9. Open questions

We have characterized the f.f. message complexity of Byzantine Agreement for most failure types. The only technical lacuna remaining is to tighten the bounds for values of n less than $8t-2$ in the case of worst-case f.f. message complexity for arbitrary failures with authentication. A more interesting question along these lines is to understand *why* there is a gap between the total number of messages that must be sent in order to achieve the optimal worst-case complexity $(n+2t-2)$ and the achievable upper bound for the total number of messages $(n+t-1)$. This result came as somewhat of a surprise to us. Although our discussion in Section 6 of why the obvious modification of GOF1 does not work is suggestive, it does not seem to get at the heart of the difficulty. Nor does our rather technical proof of Theorem 7.3.

Another line of research is that of considering the message complexity over *all* runs, not just the f.f. runs. How much worse is the worst-case complexity over all runs than the worst-case f.f. complexity? In the case of arbitrary failures, as results of Berman *et al.* [1989] and Coan and Welch [1989] show, the difference is at most a constant factor, since the worst-case message complexity over all runs is $O(nt)$. (In fact, Brian Coan [1990] has observed that from the proof of Coan and Welch [1989], a bound of $42nt+o(nt)$ can be obtained.) As we mentioned in the introduction, in the case of general omission failures and of arbitrary failures with authentication, there is a gap between the worst-case complexity over all runs and the worst-case f.f. complexity, since the former was shown to be $\Omega(n+t^2)$ by Dolev and Reischuk [1982], while, as we have shown, the latter is $\lceil(n+t-1)/2\rceil \in \Omega(n)$. For sending omission failures, the best known upper bound on the worst-case number of messages over all runs is $O(n+t^2)$ (cf. Dolev and Reischuk [1982], and Weber [1989] — both of these protocols actually work for more general failures than sending omission), while for crash failures it is $O(n+t\sqrt{t})$ (cf. Bracha [1984]). (Work in progress by Dwork, Halpern, and Waarts [1991] suggests that the upper bound in the case of crash failures may be $O(n+t\log t)$.) For neither case is a non-trivial lower bound known. If, however, the upper bounds turn out to be tight, our results would imply that the worst-case message complexity in f.f. runs would be strictly lower than the worst-case message complexity over all runs, in both cases.

Finally, and perhaps most interesting of all, is the issue of the tradeoff between number of messages and the number of rounds. We have proved some preliminary results along these lines, but many questions remain. Note that the simple one-round failure discovery protocol, where the sender sends its initial value to all processes, also works in the context of general omission failures. Thus, we have a tradeoff here: there is a one-round protocol that has worst-case f.f. message complexity of $n-1$, while any protocol that has worst-case f.f. message complexity of $\lceil (n+t-1)/2 \rceil$ must take at least $\lceil (n-t+1)/2 \rceil$ rounds. It would be interesting to obtain other points on the spectrum. How many messages do we need to use, for example, in a two-round protocol in the case of omission failures? What happens in the case of arbitrary failures? In particular, how many rounds are necessary to achieve BA (and not simply FD) if we allow arbitrary failures? The early stopping BA protocols of Dolev *et al.* [1990], and of Berman and Garay [1990] achieve BA in the f.f. runs in two rounds, using low polynomial message complexity. To what extent can the message complexity be improved in these algorithms? It seems that to answer these questions, we need a better understanding of the knowledge conveyed by the receipt of a message and the knowledge conveyed by the passage of time.

Appendix A: Randomized protocols

Throughout this paper we assumed that we were dealing with deterministic protocols. It is due to this assumption, for instance, that we could say that a f.f. run is completely determined by the sender's initial value and, therefore, in the case of binary BA there are only two f.f. runs. This may appear to be a suspect assumption, given that for certain complexity measures use of randomization can lead to much more efficient protocols than is possible with deterministic ones (cf. Chor and Dwork [1989]). For example, it is known that, in asynchronous systems, BA cannot be achieved by deterministic protocols (cf. Fischer *et al.* [1985]) while it is solvable by randomized protocols (cf. Ben-Or [1983]). More relevant to our case of synchronous systems, it is known that the worst-case round complexity over *all* runs (not just f.f. ones) for deterministic BA protocols is at least $t+1$ for all types of failures (cf. Fischer and Lynch [1982], Dolev and Strong [1983], Dwork and Moses [1986], Lynch [1989]). However, randomized protocols can solve the problem in *constant* expected rounds, even for arbitrary failures without authentication (cf. Feldman and Micali [1985, 1990]). However, as we shall show, this is not the case for the complexity measures of interest to us in this paper, i.e., various forms of message complexity *in the f.f. runs*. This will justify our assumption.

Intuitively this is not very surprising: Randomization is an effective technique when the problem at hand involves a great deal of uncertainty that must be removed in order to solve it. In such a case, an "adversary" can exploit this feature of the problem to force any solution to expend many units of some resource (e.g., rounds or messages) to dispel all the uncertainty. In the case of BA, most of the uncertainty is due to the possibility of failures. By restricting our attention to f.f. runs we are taking away much of the power of the adversary to create confusion. Now the only uncertain thing is the sender's initial value.

In a randomized protocol, in each round every process is allowed to toss a fair coin and base its actions on the outcome of that toss, in addition to its current state and the messages it receives in that round. Thus, associated with an N-round run of a protocol is a "coin toss sequence" $\vec{c} = c_1, c_2, \ldots, c_N$, where c_i specifies the outcome of the coin toss of each process in the i-th round. A randomized BA protocol satisfies the Agreement and Validity conditions and also a probabilistic version of Termination:

Probabilistic Termination: For any $0 < \epsilon \leq 1$ there is an N_ϵ such that with probability at least $1 - \epsilon$, every correct process chooses a decision value by the end of round N_ϵ (irrespective of the initial value or the number of failures).

That is, such a protocol only guarantees termination with (arbitrarily) high probability.

Fix any randomized protocol R for BA. Let EM_v denote the expected number of messages in the f.f. runs with initial value v, over all possible coin toss sequences. We can define the total, worst-case and average-case f.f. message complexity of R simply as $EM_0 + EM_1$, $\max(EM_0, EM_1)$ and $P_0 \cdot EM_0 + P_1 \cdot EM_1$, respectively, where as usual P_v denotes the probability of the initial value being v. We shall show that $EM_0 + EM_1 \geq total$, where $total$ is a lower bound on the total f.f. message complexity of deterministic BA protocols. (For example, in the case of general omission failures, $total = n + t - 1$.) Similar results hold for worst-case and average-case f.f. message complexity. This shows that randomization does not help in the context of f.f. message complexity!

Claim: $EM_0 + EM_1 \geq total$.

Proof: Pick any $0 < \epsilon \leq 1$ and let N_ϵ be as in the Probabilistic Termination condition. If $\vec{c}$ is a coin toss sequence, let $H_v^{\vec{c}}$ be the f.f. run with initial value v in which the outcomes of coin tosses performed by processes are as in $\vec{c}$. (Note that a f.f. run is now completely determined by the initial value and the coin tosses.) Let C_{N_ϵ} be the set of coin toss sequences $\vec{c}$ such that in *both* $H_0^{\vec{c}}$ and $H_1^{\vec{c}}$ all processes decide within N_ϵ rounds. Finally, let EM_{v,N_ϵ} be the expected number of messages in f.f. runs with initial value v where processes decide within N_ϵ rounds. Thus,

$$EM_{v,N_\epsilon} \geq \sum_{\vec{c} \in C_{N_\epsilon}} \Pr(\vec{c}) \cdot \mu(H_v^{\vec{c}}),$$

where $\Pr(\vec{c})$ is the probability of coin toss sequence $\vec{c}$ and $\mu(H_v^{\vec{c}})$ is the number of messages in $H_v^{\vec{c}}$.

Since the f.f. runs of R terminate within N_ϵ rounds with probability at least $1 - \epsilon$,

$$EM_0 + EM_1 \geq (1-\epsilon)(EM_{0,N_\epsilon} + EM_{1,N_\epsilon}) \geq (1-\epsilon) \sum_{\vec{c} \in C_{N_\epsilon}} \Pr(\vec{c}) \cdot \big(\mu(H_0^{\vec{c}}) + \mu(H_1^{\vec{c}})\big).$$

Note that if we fix a coin toss sequence $\vec{c}$ we can turn the randomized protocol R into a deterministic one, by having processes pretend that their "random" coin tosses always come out as in $\vec{c}$. The two f.f. runs of this deterministic protocol are precisely $H_0^{\vec{c}}$ and $H_1^{\vec{c}}$. Thus, $\mu(H_0^{\vec{c}}) + \mu(H_1^{\vec{c}}) \geq total$. Plugging this into the previous equation we get,

$$EM_0 + EM_1 \geq (1-\epsilon)\cdot total \cdot \sum_{\vec{c}\in C_{N_\epsilon}} \Pr(\vec{c}).$$

Since the probability of a f.f. run to terminate within N_ϵ rounds is at least $1-\epsilon$, and because in order for $\vec{c}$ to be in C_{N_ϵ} it must be that *both* $H_0^{\vec{c}}$ and $H_1^{\vec{c}}$ terminate within N_ϵ rounds, we have that $\sum_{\vec{c}\in C_{N_\epsilon}} \Pr(\vec{c}) \geq 2(1-\epsilon)-1 = 1-2\epsilon$. Thus, $EM_0+EM_1 \geq (1-\epsilon)(1-2\epsilon)\cdot total$. Since we can take ϵ to be arbitrarily small, we must in fact have $EM_0 + EM_1 \geq total$, as desired. ∎

The proofs of the corresponding results for worst-case and average-case f.f. message complexity are quite similar, and are left to the reader. Also, although for ease of exposition we presented these results in the case where $V = \{0,1\}$, the same arguments apply to any set V of agreement values.

Appendix B: Proof of Theorem 7.3

In this appendix we give the general proof of Theorem 7.3 without making any assumptions about the structure of protocols. We repeat the theorem here for the reader's convenience

Theorem 7.3: *If $n \geq 8t-2$ then the worst-case f.f. message complexity of any BA protocol for arbitrary failures with message authentication is at least $\lceil (n+2t-2)/2\rceil$.*

Proof: From run H_v we define a directed acyclic graph G_v. The edges of G_v correspond one-to-one to the messages in H_v and the paths of G_v correspond one-to-one to the chains of H_v. Each node of G_v is labeled with the process which receives the incoming edges (messages) and/or sends the outgoing edges (messages). (Note that several nodes may be labeled by the same process. In particular, if a process p sends a message in some round and then receives a message at a *later* round, there will be (at least) two distinct nodes with label p.) We can assume that G_v has fewer than $\lceil (n+2t-2)/2\rceil$ edges (otherwise the theorem holds and we are done). We can also assume $t \geq 2$, for otherwise again the theorem follows from Theorem 4.2.

In order to help us carry out some of our counting argument, we colour each node in G_v either black, gray, yellow, orange, silver, red, or white according to the rules below.

- A node in G_v is coloured *black* iff it is labeled by a process in R_v (i.e., a process which receives only one message in H_v and no message in $H_{\overline{v}}$). Note that a black node cannot be a root (since, by definition, it must have an incoming edge). A black node with no black proper ancestors is called *minimal.* A black node with no black proper descendents is called *maximal.*
- A node in G_v is coloured *gray* iff it is labeled by a process in $R_{\overline{v}}$. Such a node must necessarily be a root of G_v. (Otherwise it would have an incoming edge and therefore the process that labels it would be receiving a message in H_v, so it could not possibly be in $R_{\overline{v}}$.) By definition, for each gray node in G_v labeled with p there is exactly one black node labeled with p in $G_{\overline{v}}$. We say that the former *corresponds* to the latter.

To define the other colours of nodes, we must choose a particular node in G_v which is labeled by a process in R_v, which we shall call the *special* node of G_v, as follows: If some non-maximal black node in G_v does not have a corresponding gray node in $G_{\overline{v}}$, then let the special node be a non-maximal black node in G_v which does not have a corresponding gray node but all of whose black proper ancestors have corresponding gray nodes in $G_{\overline{v}}$. If every non-maximal black node in G_v has a corresponding gray node, and some maximal black node in G_v has at least t ancestors, let that node be the special node. (If there is more than one such node, we choose the special node among them arbitrarily.) Otherwise, the special node can be taken to be any minimal node. The special node may have some minimal black ancestors. A *critical* node is either a minimal black node which is not an ancestor of the special node, or the special node itself. Let $crit_v$ be the number of critical nodes in G_v.

- A node is coloured *orange* in G_v if it is a descendant of a maximal black node in G_v.
- A node is coloured *yellow* in G_v if it is a non-black node on a path between a critical node and another black node or it is the non-black child of a non-maximal black node.
- A node is coloured *red* in G_v if it is a proper ancestor of a critical black node that has not already been coloured black or gray, and has fewer than t proper ancestors.
- A node is coloured *silver* in G_v if it is a proper ancestor of a critical black node that has not already been coloured black or gray, and has t or more proper ancestors.
- Finally, all the remaining nodes are coloured *white*.

Next we shall divide up the edges in G_v into eight disjoint categories denoted $\mathcal{A}_v$ through $\mathcal{H}_v$. We shall denote the number of edges in each group by the small letter corresponding to that group (e.g., $a_v = |\mathcal{A}_v|$, $b_v = |\mathcal{B}_v|$ etc). We also establish an association between edges and nodes: With each edge we associate either its source or its target or, in the case of $\mathcal{H}_v$, both. We say process p is associated with edge e if process p labels a node associated with edge e in the association we describe below.

$\mathcal{A}_v$: For each black node that is a proper ancestor of the special black node, we include in $\mathcal{A}_v$ one edge out of that node that lies on a path to the special black node (if there are several such edges, the one picked to be in $\mathcal{A}_v$ is chosen arbitrarily); we associate with these edges their black source. For all other noncritical black nodes, we include in $\mathcal{A}_v$ the (unique) edge that leads into them. (The edge is unique since processes in R_v get exactly one message.) We associate with these edges their black target. Note that every process in R_v is either the label of a node associated with an edge in $\mathcal{A}_v$ or labels a critical node. Thus $|R_v| = a_v + crit_v$.

$\mathcal{B}_v$: Edges that lead out of a red node on a path to a critical black node. We associate with these edges their red source. Note that each red node is the source of at least one edge in $\mathcal{B}_v$.

$\mathcal{C}_v$: One edge leading out of each silver node on a path to a critical black node. Note that there will always be one such edge. There may be more than one, in which case we can pick the one in $\mathcal{C}_v$ arbitrarily. We associate with these edges their silver source.

$\mathcal{D}_v$: Edges between two yellow nodes or from a black node to a yellow node. We associate with these edges their yellow target. Note that each yellow node is the target of at least one edge in $\mathcal{D}_v$.

$\mathcal{E}_v$: Edges that lead into an orange node on a path starting at a maximal black node. We associate with these edges their orange target. Note that each orange node is the target of at least one edge in $\mathcal{E}_v$.

$\mathcal{F}_v$: Edges out of gray nodes. We further subdivide $\mathcal{F}_v$ into two disjoint subsets, $\mathcal{F}_v^m$ and $\mathcal{F}_v^n$, the edges leading out of gray nodes that correspond to maximal and non-maximal black nodes, respectively. We associate with these edges their gray source. Note that each gray node is the source of at least one edge in $\mathcal{F}_v$.

$\mathcal{G}_v$: All edges with silver sources and silver or black targets not counted in $\mathcal{C}_v$. With each node in $\mathcal{G}_v$ we associate its silver source.

$\mathcal{H}_v$: All other edges. We associate with each of these edges both their source and target. We leave it to the reader to check that if a white node is either the source or target of an edge, then that edge must be in $\mathcal{H}_v$. (The proof is a straightforward case-by-case analysis of the colour of the other node on the edge.)

Let N denote the number of messages in the two f.f. runs or, equivalently, the number of edges in G_0 and G_1. We have

$$N = \sum_{v\in\{0,1\}} (a_v + b_v + c_v + d_v + e_v + f_v + g_v + h_v). \tag{1}$$

As we shall show (cf. Claim 7.3.3 below),

$$\sum_{v\in\{0,1\}} (a_v + c_v + crit_v) + \frac{1}{2} \sum_{v\in\{0,1\}} (b_v + d_v + e_v + 2h_v) \geq n$$

Combining with (1) yields

$$N \geq n + \sum_{v\in\{0,1\}} (f_v + g_v - crit_v) + \frac{1}{2} \sum_{v\in\{0,1\}} (b_v + d_v + e_v). \tag{2}$$

Let max_v be the number of maximal black nodes in G_v; and max_v^t be the number of maximal black nodes in G_v with fewer than t proper ancestors. Let

$$\delta_v = \begin{cases} 0 & \text{if } max_v^t < max_v \\ 1 & \text{if } max_v^t = max_v. \end{cases}$$

We shall also show (cf. Claims 7.3.5 and 7.3.6 below) that

$$b_v + d_v + f_v + f_{\overline{v}}^n + g_v \geq t - max_v - max_v^t + 2crit_v - 1 + \delta_v$$

and

$$e_v + f_{\overline{v}}^m \geq t + max_v + max_v^t - 1 - \delta_v.$$

Combining these two equations, and using the observation that $\mathcal{F}_{\overline{v}}$ is the disjoint union of $\mathcal{F}^n_{\overline{v}}$ and $\mathcal{F}^m_{\overline{v}}$, we get

$$b_v + d_v + e_v + f_v + f_{\overline{v}} + g_v \geq 2t - 2 + 2crit_v$$

Substituting this into (2), and observing that $\Sigma_{v\in\{0,1\}}(f_v + f_{\overline{v}}) = 2\Sigma_{v\in\{0,1\}} f_v$, we get

$$N \geq n + 2t - 2 + \frac{1}{2}(g_0 + g_1) \geq n + 2t - 2.$$

Since the sum of the number of messages in the two f.f. runs is at least $n+2t-2$, it follows that the maximum number of messages in one of the two f.f. runs is at least $\lceil (n+2t-2)/2 \rceil$, as wanted.

It remains to establish the claims mentioned above, and some additional facts needed along the way. The first two claims are immediate corollaries of Lemmas 4.1(1) and 7.4, respectively. All that we have to do is translate messages and message chains in the f.f. runs into edges and paths in the two graphs.

Claim 7.3.1: *If a black node in G_v has $k \leq t$ proper descendents in G_v then it has a corresponding gray node in $G_{\overline{v}}$ which has at least $t-k$ children in $G_{\overline{v}}$.* ∎ Claim 7.3.1

Claim 7.3.2: *If a node labeled p in G_v has $t-k$ proper ancestors in G_v then either (a) that node has at least $k+1$ children in G_v, or (b) there is another node labeled p in G_v with an incoming edge, or (c) there is another node labeled p in $G_{\overline{v}}$ with an outgoing edge.* ∎ Claim 7.3.2

Claim 7.3.3: $\sum_{v\in\{0,1\}}(a_v + c_v + crit_v) + \frac{1}{2}\cdot\sum_{v\in\{0,1\}}(b_v + d_v + e_v + 2h_v) \geq n.$

Proof: In order to prove the result, we need to do a very careful count of the total number of processes. We shall do this by considering the various colours and making sure we count the processes that label nodes of every colour.

As we mentioned earlier, all the processes in $R_0 \cup R_1$ label nodes that are coloured black or gray, and there are $a_0 + a_1 + crit_0 + crit_1$ of these. Clearly, an upper bound on the number of processes labeling silver nodes is $c_0 + c_1$.

All that remains is to get an upper bound on the processes we have not yet considered, which all must label nodes in $G_0 \cup G_1$ coloured orange, yellow, red, or white and do not label any node coloured silver. Let $X = \bigcup_{v\in\{0,1\}}(\mathcal{B}_v \cup \mathcal{D}_v \cup \mathcal{E}_v \cup \mathcal{H}_v)$. We claim that

every such process is associated with (at least) *two* distinct edges in X $\qquad (*)$

Thus, the number of remaining processes is no more than half of the number of nodes associated with the edges in X. Since each edge in $\mathcal{H}_v$ is associated with two nodes while each edge is $\mathcal{B}_v$, $\mathcal{D}_v$ and $\mathcal{E}_v$ with only one, this means that the number of remaining processes is bounded by $\frac{1}{2}(b_v + d_v + e_v + 2h_v)$. Thus, proving (*) will complete the proof of Claim 7.3.3.

Suppose process p labels a node coloured orange, yellow, red, or white, and does not label a node coloured silver. First, consider the case where p is the sender. Without loss of generality we can assume that there is at least one node labeled by the sender which is a root in G_0 or G_1. This is because messages that are sent before the earliest round in which the sender sends a message in either f.f. run must be present in both f.f. runs and are completely redundant as they do not help any process in discriminating between the two f.f. runs; we can therefore assume that such messages do not exist. Let x be a node labeled by p which is a root in G_v. By the rules used to colour nodes, a root can only be gray, red or white. Node x cannot be gray (because the sender is not in $R_0 \cup R_1$). Thus it must be red or white. It is easy to see that the edges out of x are in $\mathcal{B}_v \cup \mathcal{H}_v$, and x is associated with each such edge. If there is more than one such edge, we are done. If there is only one, then by Claim 7.3.2, there must be another node y in $G_0 \cup G_1$ which is also labeled by p. We now show that there is an edge incident to y which is associated with y. We proceed by cases on the colour of y (which must be yellow, white, red, or orange). If y is yellow then there is at least one edge in $\mathcal{D}_0 \cup \mathcal{D}_1$ whose target is y, so this is the second edge we associate with p. If y is white, there must be an edge with y as source or target, and this edge is in $\mathcal{H}_0 \cup \mathcal{H}_1$ and is associated with p. We leave the remaining cases to the reader.

Next, suppose that p is a receiver. Since p is not in $R_0 \cup R_1$, it must receive at least two messages in the two f.f. runs. Thus, there must exist two distinct edges e and e' whose targets are labeled by p. Let x and x' be the targets of these two edges (it is possible that $x = x'$). If x (resp. x') is not coloured red then it can be checked that e (resp. e') is in X. If x (resp. x') is red then there is an edge out of it which is in $B_0 \cup B_1$, and hence is in X. The only case in which we don't already have two edges in X associated with p is if $x = x'$ and that node is coloured red. The argument now is identical to that where p is the sender: either we have two edges with source x, or there is another node y labeled by p. In either case, we get two edges in X associated with p. ▮ Claim 7.3.3

Claim 7.3.4: $crit_v + a_v \geq t$.

Proof: Let $u \in \{0, 1\}$. Recall that we have assumed that G_u has fewer than $\lceil (n+2t-2)/2 \rceil$ edges. Thus, we certainly have $a_u + b_u + c_u + f_u + g_u \leq (n + 2t - 2)/2$. It also follows that $N < n + 2t - 2$. A critical node, other than possibly the special node in G_u is the target of an edge in $\mathcal{B}_u$, $\mathcal{C}_u \cup \mathcal{G}_u$ or $\mathcal{F}_u$, according to whether its parent is red, silver or gray. (The special black node, in addition to these possibilities, may also be the target of an edge in $\mathcal{A}_v$, in case its parent is black.) Therefore, $crit_u \leq b_u + c_u + f_u + g_u + 1$. Thus, we have

$$|R_u| = a_u + crit_u \leq (n + 2t - 2)/2 + 1 = (n + 2t)/2 \quad \text{for } u = 0, 1.$$

Every receiver that is not in R_0 or R_1 must receive at least two messages in the two f.f. runs. There are $n - |R_0 \cup R_1| - 1$ such receivers (all the processes other than the sender that are not in $R_0 \cup R_1$), so we have

$$N \geq a_0 + a_1 + crit_0 + crit_1 + 2(n - a_0 - a_1 - crit_0 - crit_1 - 1).$$

Thus, $n+2t-2 > 2n-2-a_0-a_1-crit_0-crit_1$, which implies that $a_0+a_1+crit_0+crit_1 > n-2t$. Therefore,

$$a_v + crit_v > n - 2t - a_{\overline{v}} - crit_{\overline{v}} \geq n - 2t - (n+2t)/2 = (n-6t)/2 \geq t-1$$

as wanted (the last inequality follows because, by hypothesis, $n \geq 8t-2$). ▮ Claim 7.3.4

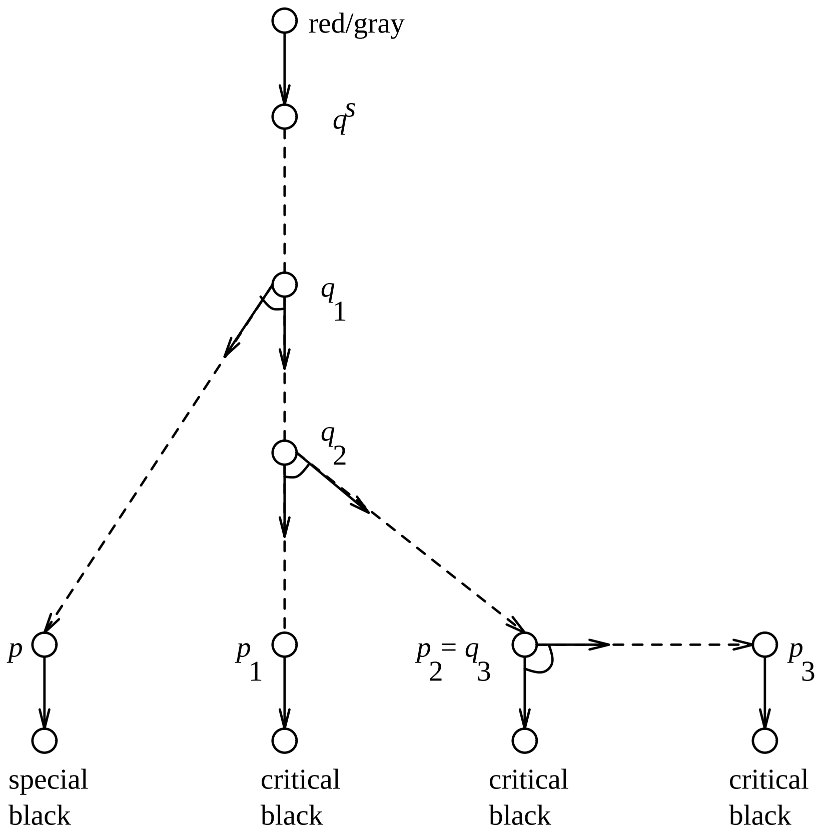
Claim 7.3.5: $b_v + d_v + f_v + f^n_{\overline{v}} + g_v \geq t - max_v - max^t_v + 2crit_v - 1 + \delta_v$.

Proof: We consider two cases, depending on the number of proper ancestors of the special node.

Case (a): The special node has at least t proper ancestors. Let p be the special node in G_v. If p has a silver ancestor, let p' be a silver ancestor of p with no silver proper ancestors. Otherwise, let p' be p. Notice that p' has at least t proper ancestors. (If p' is silver, this is true by definition; if p' is p, it is true by assumption.) We first identify t distinct edges in $\mathcal{B}_v \cup \mathcal{F}_v \cup \mathcal{F}^n_{\overline{v}}$, one associated with each of the proper ancestors of p'. Consider a proper ancestor q of p'. By our definitions, q must be either red, gray, or black. If it is red or gray, there is an edge in $\mathcal{B}_v \cup \mathcal{F}_v$ with source q on the path to p'. Finally, if it is black, by the choice of the special node, there is a corresponding node in $G_{\overline{v}}$ which is the source of an edge in $\mathcal{F}^n_{\overline{v}}$.

Now consider the $crit_v$ critical nodes in G_v. If $crit_v \geq max_v$, then then there are at least $crit_v - max_v$ edges in $\mathcal{D}_v$ which are on paths that begin at the critical nodes. This is because if there is a maximal black node which is a descendent of two different critical nodes, the paths from these nodes to the maximal black node must use distinct edges in $\mathcal{D}_v$.

We plan to identify $crit_v - 1$ more edges in $\mathcal{B}_v \cup \mathcal{F}_v \cup \mathcal{G}_v$, one corresponding to each of the critical nodes other than the special node. Note that the parent of a critical node other than the special must be either red, gray, or silver. We divide the critical nodes in G_v other than the special node into two sets: W_1, consisting of all those whose parent is either red or gray, and W_2, those whose parent is silver. If $q \in W_1$, then the edge leading to q must be in $\mathcal{B}_v \cup \mathcal{F}_v$ and has not yet been counted. This gives us $|W_1|$ new edges. With each $q \in W_2$ we associate a silver ancestor q^s of q which itself has no silver proper ancestors. It is possible to associate the same silver ancestor with more than one node in W_2. Let q^s be a silver node associated with ℓ different nodes $p_1, \ldots, p_\ell$ in W_2. We shall identify ℓ edges in $\mathcal{B}_v \cup \mathcal{F}_v \cup \mathcal{G}_v$ that were not previously considered. First consider the edge from q^s's parent to it (there must be at least one such edge since q^s is silver; there may be several but it is enough to consider just one). The parent must be either red or gray (for q^s has no silver proper ancestors) and thus the edge that leads from it to q^s is in $\mathcal{B}_v \cup \mathcal{F}_v$. This edge has not been considered yet if q^s is not an ancestor of the special black node. If it is, we can identify an edge in $\mathcal{G}_v$ that has not been considered as follows: Let p be the special node and p_1 be any one of the ℓ nodes in W_2 with which q^s has been associated. Thus, there exist paths from q^s to p and p_1. Let q_1 be the first node after which these two paths diverge — it may be that $q_1 = q^s$. (Refer to Figure 7.) Node q_1 must be silver (it cannot be black as it is a proper ancestor of p_1 which, being a critical black node other than the special, must be a minimal black node). Then one of the outgoing edges from



$p_1, p_2, p_3 \in W_2$ are all associated with minimal silver node q^s
Solid arrows indicate edges identified in the argument (when two arrows are joined by an arc, at least one of them is an edge of interest)
Broken arrows indicate paths.

Figure 7

q_1 on the paths to p and p_1 may be in $\mathcal{C}_v$ but then the other is in $\mathcal{G}_v$. Thus, even if q^s is an ancestor of the special node, we can identify an edge as wanted. Now we need to identify $\ell - 1$ more such edges. Consider the $\ell - 1$ nodes with which q^s has been associated, *other* than p_1. For each one of them, say p_i, $i > 1$, consider the first node q_i after which the paths from q^s to p_1 and p_i diverge. Again, we can identify another edge in $\mathcal{G}_v$ just as in the case we considered above (see Figure 7). In this manner we can identify one edge in $\mathcal{B}_v \cup \mathcal{F}_v \cup \mathcal{G}_v$, not previously counted, with every node in W_2, for a total of $|W_2|$ new edges. Since $W_1 \cup W_2$ is the set of all critical nodes in G_v except the special one, we have identified $crit_v - 1$ new edges, as promised.

Thus, we have identified $t + crit_v - max_v + crit_v - 1$ edges in $\mathcal{B}_v \cup \mathcal{F}_v \cup \mathcal{F}_{\overline{v}}^n \cup \mathcal{G}_v$ thus far. This shows that $b_v + f_v + f_{\overline{v}}^n + g_v \geq t + crit_v - max_v + crit_v - 1$. Notice that if $\delta_v = 1$, then $max_v^t = max_v \geq 1$, so that under all circumstances $\delta_v - max_v^t \leq 0$. Thus, it follows that $b_v + f_v + f_{\overline{v}}^n + g_v \geq t - max_v - max_v^t + 2crit_v + \delta_v - 1$.

Case (b): The special node has fewer than t proper ancestors. There are two subcases to consider: (i) all non-maximal black nodes have a corresponding gray node, or (ii) they don't. In subcase (i), it must be the case that all maximal nodes have less than t ancestors (otherwise the special node would be a maximal node with t or more ancestors, and we would be in case (a)), so $max_v = max_v^t$. Moreover, since there are $a_v + crit_v - max_v$ non-maximal black nodes in G_v, all of which have a corresponding gray node in $G_{\overline{v}}$, using Claim 7.3.4 we have at least $t - max_v^t$ edges in $\mathcal{F}_{\overline{v}}^n$. Using the same arguments as in the second paragraph of case (a), we can get a further $crit_v - max_v$ edges in $\mathcal{D}_v$; and using the same arguments as in the third paragraph of case (a), we can identify an additional $crit_v - 1$ edges in $\mathcal{B}_v \cup \mathcal{F}_v \cup \mathcal{G}_v$. Since $\delta_v = 0$ for this subcase, we have shown that $b_v + d_v + f_v + f_{\overline{v}}^n + g_v \geq t - max_v^t - max_v + 2crit_v - 1 + \delta_v$, as desired.

Finally, for subcase (ii), suppose that the special node has $t-k$ proper ancestors, with $k > 0$. Using the same arguments as in the first paragraph of case (a), we can identify $t - k$ edges in $\mathcal{B}_v \cup \mathcal{F}_v \cup \mathcal{F}_{\overline{v}}^n$. By Claim 7.3.2, we also have that the special node has at least $k + 1$ children (because, by definition, in this subcase it has no corresponding gray node in $G_{\overline{v}}$, and, being black, there surely can't be another node in G_v with an incoming edge). Consider the $k + 1$ edges that go out of the special node. A certain number, say α, go to yellow nodes, and are therefore in $\mathcal{D}_v$, while the remaining $k + 1 - \alpha$ go to black nodes. Consider these $k + 1 - \alpha$ nodes together with the $crit_v - 1$ critical nodes other than the special node. By analogous arguments as in the second paragraph of case (a), if $k + crit_v - \alpha - max_v \geq 0$, we can show that there are $k + crit_v - \alpha - max_v$ additional edges in $\mathcal{D}_v$ which are on paths that begin at the critical nodes other than the special one or at one of the $k + 1 - \alpha$ black children of the special node. This gives us at least $t + crit_v - max_v$ edges. Arguing just as in the third paragraph of case (a), we can identify a further $crit_v - 1$ edges in $\mathcal{B}_v \cup \mathcal{F}_v \cup \mathcal{G}_v$. Since, as we have seen, $\delta_v - max_v^t \leq 0$, we again get the required number of edges.

This completes the proof. ▮ Claim 7.3.5

Claim 7.3.6: $e_v + f_{\overline{v}}^m \geq t + max_v + max_v^t - 1 - \delta_v$.

Proof: If $max_v \neq max_v^t$, let p be a maximal black node with t or more proper ancestors; otherwise, let p be an arbitrary maximal black node. Let M be the set of edges which are on paths that begin at p and the edges in $G_{\overline{v}}$ which come out of the gray node that corresponds to p (if any). Note that these edges are in $\mathcal{E}_v \cup \mathcal{F}_{\overline{v}}^m$. By Claim 7.3.1, the number of edges in M is at least t.

Next, we show that for each maximal node other than p with fewer than t proper ancestors, we can identify two edges in $\mathcal{E}_v \cup \mathcal{F}_{\overline{v}}^m$ but which are not in M. Let q be a maximal black node with fewer than t proper ancestors. Then, by Claim 7.3.2, it follows that either q has at least two outgoing edges or q has a corresponding gray node q' in $G_{\overline{v}}$ with at least one outgoing edge which is in $\mathcal{F}_{\overline{v}}^m$. (Note that the remaining alternative of Claim 7.3.2 is impossible, because the fact that q is black implies that there cannot be another node in G_v that has an incoming edge and is labeled with the same process as q, since that process is in R_v.) In the former case we are done (since both outgoing edges are in $\mathcal{E}_v$ and not in M). In the latter case, if q has a child in G_v, again we are done, since we have one edge in $\mathcal{E}_v$ together with one in $\mathcal{F}_{\overline{v}}^m$. Finally, if q has no children in G_v, then by Claim 7.3.1, q' has $t + 1 \geq 2$ outgoing edges, which are all in $\mathcal{F}_{\overline{v}}^m$ and not in M.

Finally, if q is a maximal black node other than p with t or more proper ancestors, then by Claim 7.3.1, either q has an outgoing edge in $\mathcal{E}_v$ not included in M, or it has a corresponding gray node in $G_{\overline{v}}$ which has an outgoing edge that belongs to $\mathcal{F}^m_{\overline{v}}$ and is not included in M.

Putting all this together, we get: If $max^t_v = max_v$ (i.e., if all maximal black nodes have at most t proper ancestors), we have a total of $t + 2(max^t_v - 1) = t + max_v + max^t_v - 2$ edges in $\mathcal{E}_v \cup \mathcal{F}^m_{\overline{v}}$. However, $max^t_v = max_v$ implies that $\delta_v = 1$ in this case and therefore we get a total of $t + max_v + max^t_v - (1 + \delta_v)$ edges in $\mathcal{E}_v \cup \mathcal{F}^m_{\overline{v}}$, giving the claim.

If, on the other hand, $max^t_v < max_v$, we have a total of $t + 2max^t_v + (max_v - max^t_v - 1) = t + max_v + max^t_v - 1$ edges in $\mathcal{E}_v \cup \mathcal{F}^m_{\overline{v}}$. The claim now follows immediately. ▮ Claim 7.3.6

Acknowledgements

We'd like to thank Brian Coan, Jennifer Welch, and Juan Garay for helping us to track down the message complexity of Byzantine Agreement for arbitrary failures over all runs, and Noga Alon for useful discussions on the message complexity of the multi-valued case. Jim Caldwell, Guerney Hunt, Prasad Jayanti and Sridhar Sundaram pointed out various deficiencies of an earlier draft. Finally, we would like to thank the anonymous referees for their careful reading of the paper, and numerous useful suggestions for improvement. The work of Hadzilacos was supported, in part, by a grant from the Natural Sciences and Engineering Research Council of Canada.

Bibliography

- Alon, N. Private communication, June 1990.
- Amdur, E.S, S.M. Weber and V. Hadzilacos. "On the Message Complexity of Binary Byzantine Agreement Under Crash Failures". Submitted for publication. March 1990.
- 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.
- Ben-Or, M. "Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols". In *Proc. of the 2nd ACM Symp. on Principles of Distributed Computing,* pp. 27–30, August 1983.
- Berman, P. and J. A. Garay. "Optimal Early Stopping in Distributed Consensus". IBM Research Report RC 16746, December 1990.
- Berman, P., J. A. Garay and K. J. Perry. "Recursive Phase King Protocols for Distributed Consensus". Penn State Report CS-89-24, August 1989.
- Bollobas, B. *Extremal Graph Theory*, Academic Press, London, New York, 1978.
- Bracha, G. Unpublished manuscript, Department of Computer Science, Cornell University, July 1984.
- Chor, B. and C. Dwork. "Randomization in Byzantine Agreement". In *Randomness and Computation*, edited by S. Micali. Advances in Computing Research, vol. 5, pp. 433–498, JAI Press, 1989.

- Coan, B. and J. Welch. "A Byzantine Agreement Protocol with Optimal Message Bit Complexity". In *Proceedings of the 27th Annual Allerton Conference on Communication, Control, and Computing*, pp. 1062–1071, 1989.
- Coan, B. Private communication. 1990
- Dolev, D. and R. Reischuk. "Bounds on Information Exchange for Byzantine Agreement". *Journal of the ACM*, 32(1):191–204, January 1985.
- Dolev, D., R. Reischuk, and H.R. Strong. "Early Stopping in Byzantine Agreement". *Journal of the ACM*, 37(4):720–741, 1990.
- Dolev, D. and H.R. Strong. "Authenticated Algorithms for Byzantine Agreement". *SIAM Journal on Computing* 12(4):656–666, November 1983.
- Dwork, C., J.Y. Halpern and O. Waarts, unpublished manuscript, 1991.
- Dwork, C. and Y. Moses. "Knowledge and Common Knowledge in a Byzantine Environment I: Crash Failures". In *Proc. of the Conf. on Theoretical Aspects of Reasoning About Knowledge*. pp. 149–169, March 1986.
- Feldman, P. and S. Micali. "Byzantine Agreement in Constant Expected Time (and Trusting No One)". In *Proc. of the 26th Annual IEEE Symp. on Foundations of Computer Science*. pp. 267–276, October 1985.
- Feldman, P. and S. Micali. "An Optimal Algorithm for Synchronous Byzantine Agreement". Technical Report MIT/LCS/TM-425, Laboratory for Computer Science, Massachusetts Institute of Technology, June 1990.
- Fischer, M.J. "The Consensus Problem in Unreliable Distributed Systems". Research Report RR-273, Department of Computer Science, Yale University, June 1983.
- Fischer, M.J. and N.A. Lynch. "A Lower Bound for the Time to Assure Interactive Consistency". *Information Processing Letters*, 14(4):183–186, June 1982.
- Fischer, M.J., N.A. Lynch and M.S. Paterson. "Impossibility of Distributed Consensus with One Faulty Process". *Journal of the ACM*, 32(2):374–382, April 1985.
- Gray, J.N. "The Cost of Messages". In *Proc. of the 7th ACM Symp. on Principles of Distributed Computing*. pp. 1–7, August 1988.
- 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 to be published by Springer-Verlag.)
- Hadzilacos, V. and J.Y. Halpern. "The Failure Discovery Problem". Submitted for publication, June 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.

- Lynch, N.A. "A Hundred Impossibility Proofs for Distributed Computing". In *Proc. of the 8th ACM Symp. on Principles of Distributed Computing.* pp. 1–27, August 1989.
- 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.
- Srikanth, T.K. and S. Toueg. "Simulating Authenticated Broadcasts to Derive Simple Fault-Tolerant Algorithms". *Distributed Computing*, 2:80–94, 1987.
- Weber, S.M. *Bounds on the Message Complexity of Byzantine Agreement.* Masters' Thesis, Department of Computer Science, University of Toronto, October 1989.