Early-Deciding Consensus is Expensive

Danny Dolev
Hebrew University of Jerusalem
Edmond Safra Campus
91904 Jerusalem, Israel
dolev@cs.huji.ac.il

Christoph Lenzen
Massachusetts Institute of Technology
32 Vassar Street
02139 Cambridge, USA
clenzen@csail.mit.edu

ABSTRACT

In consensus, the n nodes of a distributed system seek to take a consistent decision on some output, despite up to t of them *crashing* or even failing maliciously, i.e., behaving *"Byzantine"*. It is known that it is impossible to guarantee that synchronous, deterministic algorithms consistently decide on an output in fewer than $f+1$ rounds in executions in which the *actual number of faults* is $f \leq t$. This even holds if faults are crash-only, and in this case the bound can be matched precisely. However, the question of whether this can be done efficiently, i.e., with little communication, so far has not been addressed.

In this work, we show that algorithms tolerating Byzantine faults and deciding within $f+2$ rounds must send $\Omega(nt + t^2 f)$ messages; as a byproduct, our analysis shows that decision within $f+1$ rounds is impossible in this setting (unless $f = t$). Moreover, we prove that any crash-resilient algorithm deciding in $f+1$ rounds has worst-case message complexity $\Omega(n^2 f)$. Interestingly, this changes drastically if we restrict the fault model further. If crashes are *orderly*, i.e., in each round, each node picks an order in which its messages are sent, and crashing nodes successfully transmit a prefix of their sequence, deciding in $f+1$ rounds can be guaranteed with $\mathcal{O}(nt)$ messages.

Categories and Subject Descriptors

C.4 [**Computer Systems Organization**]: Performance Of Systems—*Fault-Tolerance*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems

General Terms

Algorithms, Reliability, Theory

Keywords

lower bounds; cubic message complexity; Byzantine faults; crash faults; early-stopping

PODC'13, July 22–24 2013, Montréal, Québec, Canada.

1. INTRODUCTION & RELATED WORK

In consensus, each node of a distributed system has an input value, and all nodes that do not fail need to decide on the same output value. To make this output meaningful, it is required that the output must be input of at least one node.[1] This easily stated task is very fundamental, as it captures the essence of exploiting redundancy to establish resilience to faults of (some of) the components of a system.

Consequently, consensus has been studied very deeply and extensively in the last three decades. There are scores of lower bounds on many aspects of the problem in a variety of models. In the current paper, we come back to the question of *early-deciding* and study its message complexity. Informally, an algorithm is early-deciding if, in each execution, it minimizes the number of rounds until all nodes have decided depending on the number of *actual* faults f, as opposed to t, the maximal number of faults it can tolerate, or the number of nodes n. This property is highly desirable, as in most scenarios it is reasonable to assume that most of the executions will experience few or no faults. It is closely related to the concept of *early-stopping* algorithms, where the requirement is to not only decide on the output, but also stop sending messages. Roughly speaking, early decision and termination go hand in hand; to the best of our knowledge, in all published early-deciding algorithms, after taking a decision nodes can stop communicating once they have shared their decision value with all other nodes, i.e., at most one round after taking a decision.

The question of early decision so far has not been considered for randomized algorithms. With randomization and/or asynchrony involved (and the resulting variety of adversarial models) the current understanding of both time and message complexity is much poorer than in the synchronous, deterministic case. For this reason, we consider synchronous, deterministic algorithms only, for which at the least one parameter—the time complexity—has been analyzed in great detail and precision. The basic time lower bound in this setting states that $\min(f+2, t+1)$ rounds are required for termination, even if we consider crash faults only [7, 8]; showing that $f+1$ rounds are required for decision (in the worst case) follows from standard bivalency arguments in virtually any reasonable fault model [11]. A message complexity lower bound of $\Omega(nt)$ has been known for long [6]. The question whether stopping early is more

[1]The precise requirement depends on the considered model. The common theme is that pre-existing agreement must be maintained; in particular, the trivial algorithm always returning a default value is not a feasible solution.

costly in terms of communication was not considered up to the present date.

Contribution

In the current paper, we show a lower bound of $\Omega(t^2 f)$ on the number of messages sent, in the worst case, by any protocol that decides within $f+2$ rounds and tolerates up to t Byzantine faults. Doing so, we also see that in the case of Byzantine faults, decision before round $f+2$ is not always possible. Moreover, we establish a lower bound of $\Omega(n^2 f)$ on the worst-case message complexity of crash-resilient algorithms that decide within $f+1$ rounds. In other words, there are executions in which, essentially, in each round all pairs of operational nodes need to exchange messages. In contrast, we provide a simple algorithm that decides in $f+1$ rounds (and stops in $f+2$) using $\mathcal{O}(nt)$ messages, provided that crashes are *orderly*. A crash is orderly if the crashing node succeeds in sending exactly the messages in some prefix of a priority list; this list is chosen by the algorithm.

Related Work

The message complexity of achieving consensus (without the constraint of deciding or stopping early) was studied extensively, with many efforts in matching and circumventing the known lower bounds. Reischuk looked at improving the efficiency by looking at average complexity [25]. Perry and Toueg [23] focused on marching the number of rounds in weaker fault models. This approach was later followed by Parvedy and Raynal [21] as well.

Fitzi and Martin [12] skirted the message complexity lower bound of $\Omega(nt)$ by introducing a small probability of error. King and Saia [16, 17] used randomization to simultaneously beat the time and message complexity lower bounds that hold for deterministic algorithms, with "practical and scalable protocols". Many sublinear-time randomized algorithms have been published before, but all of them are much more costly in terms of communication; we make no attempt to cover the related work on randomized algorithms here.

Researchers looked at the difference between early-decision and early-stopping mainly in the context of *uniform* consensus. In uniform consensus, the decision value needs to be the same for *all* nodes that decide, even those that fail after deciding. Keidar and Rajsbaum [15] proved a slightly stronger lower bound for crash-resilient algorithms for uniform consensus: deterministic protocols cannot always *decide* before round $\min\{f+2, t\}$. We show essentially the same bound, but for Byzantine-tolerant algorithms without requiring uniformity. Charron-Bost and Schiper [2] proved that this bound does not hold for crash-tolerant consensus algorithms, where it is possible to decide by the end of round $f+1$. Michel Raynal [24] presents a thorough study of early-deciding algorithms in a crash-fault model.

Matching the round complexity lower bound of $\min\{f+2, t+1\}$ for the general case of Byzantine faults was solved by Berman et al. [1].[2] *Omission* faults, where besides crashing nodes may also fail to send (or receive) messages non-faulty nodes would send (or receive), but continue executing the algorithm afterwards, strictly contain crash faults. Hence, in particular all lower bounds for crash faults apply. Parvedy and Raynal [21] give an algorithm for uniform consensus that matches the decision and stopping time lower bounds of $\min\{f+2, t+1\}$. The algorithm has message complexity $\mathcal{O}(n^2 f)$, but its optimality is not immediate since our lower bound for crash faults requires decision in $f+1$ rounds. We conjecture, however, that it is straightforward to adapt the techniques we introduce to establish a matching lower bound for uniform consensus.

Finally, the communication costs of early-deciding and early-terminating algorithms must be contrasted against the ones of algorithms that sacrifice these properties or incur suboptimal bounds in exchange for a smaller number of sent messages. Coan and Welch [4, 5] considered the special case of $n = 3t+1$, achieving asymptotically optimal $\mathcal{O}(t^2)$ communicated bits. Hadzilacos and Halpern devised algorithms that ensure optimality of fault-free runs [14], for an entire spectrum of fault models. Dwork and Moses [9] and later Mizrahi and Moses [19, 20] considered continuous agreement, where one can exploit that over time faulty processes may be identified and ignored. Lately, Liang and Vaidya attained asymptotically optimal bit complexity for very long output (or a sequence of outputs) [18]; Patra [22] leverages randomization to obtain the same bound in a constant expected number of rounds. None of these algorithms are early-deciding or -stopping, in the sense that they may run for at least $t+1$ rounds even if there are few faults. In contrast, Galil et al. [13] manage to achieve a strong combination of early-decision and message complexity: they present a crash-resilient algorithm that solves consensus in $\mathcal{O}((f+1)8^{1/\epsilon})$ rounds with $\mathcal{O}(n + fn^{\epsilon})$ messages, for arbitrary $\epsilon > 0$. Slightly better bounds are known if the requirement of early-stopping is dropped altogether: In $\mathcal{O}(t)$ rounds, consensus can be solved with up to $t \in n - \Omega(n)$ crash faults using $\mathcal{O}(n \log^2 t)$ messages [3]. The authors of [1] claim that a nearly round-optimal and Byzantine tolerant solution exists that is also asymptotically bit (and thus message) optimal, i.e., sends $\mathcal{O}(nt)$ bits.[3]

2. MODEL AND PROBLEM

We are given a fully connected system of n nodes, up to t of which may become faulty. In each synchronous round $1, 2, \ldots$, each node (i) performs deterministic local computations, (ii) sends (possibly different) messages to (a subset of) the other nodes, and (iii) receives the messages sent to it by other nodes. Since we are concerned with a worst-case lower bound on the number of messages sent by non-faulty nodes, we assume that nodes have unique identifiers $V = \{1, \ldots, n\}$, the recipient of a message can identify the source of a message (i.e., messages are authenticated), and we put no bound on the size of messages or the amount of local computations performed. Clearly, the derived lower bound then extends to any weaker system model. An algorithm can thus be interpreted as a collection of functions that, for each round $r \in \mathbb{N}$, maps the input of each node v and the history $H_v(r)$ of messages it received up to and including round $r-1$ (including the information who sent which message) to (i) the messages it sends in round r and (ii) whether it decides on an output o_v. We do permit v to send messages also after deciding on o_v, but v may decide on an output only once, i.e., such a decision is final.

An *execution* $\mathcal{E}$ of a *binary consensus Algorithm* $\mathcal{A}$ is specified by (i) the input values $b_v^{\mathcal{E}} \in \{0, 1\}$, (ii) the sets of faulty

[2]The extended abstract contains a proof sketch; no full version is available.

[3]No proof of this result has been published.

nodes $F_1^{\mathcal{E}} \subseteq F_2^{\mathcal{E}} \subseteq \ldots$ in each round, where $|F_r^{\mathcal{E}}| \leq t \in \mathbb{N}$ for all rounds $r \in \mathbb{N}$, and (iii) the messages sent by faulty nodes in $F_r^{\mathcal{E}}$ to non-faulty nodes in $V \setminus F_r^{\mathcal{E}}$ in each round r. The faulty nodes may send arbitrary messages, whereas the messages sent by non-faulty nodes as well as when they decide on what output are determined by the algorithm. By $f^{\mathcal{E}} := \max_{r\in\mathbb{N}}\{|F_r|\} \leq t$ we denote the number of (actual) faults in execution $\mathcal{E}$. For ease of notation, we may omit the superscript $\mathcal{E}$ denoting the execution whenever it is clear from the context. We remark that considering the nodes in the sets F_r non-faulty until they actually fail merely serves to facilitate intuition; the same asymptotic lower bound on the number of messages sent by non-faulty nodes follows from our arguments if we consider the set $\bigcup_{r\in\mathbb{N}} F_r$ faulty for the entire execution. We may also talk of an *r-round execution*, which is the restriction of an execution to its first r rounds, and *extensions*, which add further rounds to a given execution.

We say that an execution $\mathcal{E}$ is *indistinguishable* at node v from an execution $\mathcal{E}'$ (before round $r+1$), if $b_v^{\mathcal{E}} = b_v^{\mathcal{E}'}$ and $H_v^{\mathcal{E}}(r') = H_v^{\mathcal{E}'}(r')$ for all $r' \in \mathbb{N}$ (for $r' = r$). Note that since the actions of node v until round r and the messages it sends in round $r+1$ are a function of b_v and $H_v(r)$ only, v will behave identically in $\mathcal{E}$ and $\mathcal{E}'$ until receiving messages in round $r+1$ of these executions if they are indistinguishable before round $r+1$.

For $\mathcal{A}$ to be a correct binary consensus algorithm, it must satisfy the following properties in all feasible executions $\mathcal{E}$.

Termination: Nodes v that remain non-faulty eventually output some $o_v \in \{0,1\}$.

Agreement: There is a value o such that $o_v = o$ for all non-faulty nodes v.

Validity: There is some node $v \in V$ so that $b_v = o$.

3. BYZANTINE FAULTS

As deterministic consensus is impossible if $n \leq 3t$ and faults are arbitrary (i.e., Byzantine), we assume that $n > 3t$ throughout this section. In the introduction we discussed that for all deterministic binary consensus algorithms and all $0 \leq f < t$, there are executions so that a non-faulty node stops sending messages in a round $r \geq f+2$ [7, 8]. This bound can be matched for any $t < n/3$, i.e., there is an *early-stopping* algorithm that guarantees that all non-faulty nodes stop by round $\min\{f+2, t+1\}$ [1]. We consider a weaker constraint in that we require that nodes decide quickly, but may continue to send messages in support of the yet undecided nodes.

Definition 3.1 (Early-Deciding and -Stopping). *A consensus algorithm is* d-deciding *if in each execution $\mathcal{E}$ of the algorithm with $f^{\mathcal{E}} = f$, each node $v \in V \setminus F_{d(f)}$ has decided in some round $r_v \leq d(f)$. It is called* d-stopping, *if it is d-deciding and non-faulty nodes do not send messages in rounds $r > d(f)$ of executions with f faults.*

As a byproduct of our analysis, we will show that also early-deciding algorithms must run for at least $f+2$ rounds for all nodes to decide on the output, for any $0 \leq f < t$. Our main result in this section, however, is the following lower bound on the number of messages sent by deterministic early-deciding consensus algorithms.

Theorem 3.2. *For any $(f+2)$-deciding binary consensus algorithm and any $1 \leq f \leq t/2$, there is an execution $\mathcal{E}$ of the algorithm in which $f^{\mathcal{E}} = f$ and non-faulty nodes send at least $t^2 f/44$ messages.*

In the remainder of this section we will prove Theorem 3.2. Let us start with an outline of the key ideas of the proof.

Proof outline for Theorem 3.2. Clearly, as possibly $t \in \Theta(n)$ and in any round $\mathcal{O}(n^2)$ messages are exchanged, we must construct executions that have $\Omega(f)$ rounds. In fact, it will be vital that in each round exactly one node fails, as this guarantees that if no further faults happen after round r, all nodes must decide by the end of round $r+2$. Similar to the classic lower bound [7, 8] of $f+2$ on the round complexity of deterministic consensus algorithms, we will achieve this by failing, in each round, a "pivotal" node that is decisive for whether the execution would result in output 0 or output 1, so that some of the nodes perceive an execution that would end up in output 0 and others one that would produce output 1 (without further faults and for a certain behavior of the faulty nodes).

However, the simple proof of existence of such nodes from [7, 8] is insufficient for our purposes, as we also need to make sure that in each round $\Omega(t^2)$ messages are sent. To this end, we leverage the argument from [6] yielding the well-known lower bound of $\Omega(t^2)$ on the message complexity of consensus algorithms. Essentially, we argue that not only $\Omega(t^2)$ messages need to be exchanged to achieve consensus, but (i) this has to happen in round $r+1$ if we fail no node in rounds r and $r+1$ (as the algorithm needs to verify the decision it must take by the end of round $r+1$), and (ii) this can be exploited to do the "pivoting" such that indeed also in the modified execution with more faults many messages are sent in round $r+1$. While in general it can not be guaranteed that many messages must be sent right away, $(f+1)$-deciding algorithms are forced to act quickly, as in each round r, each node must be able to prove that at least $r-1$ faults occurred or decide by the end of the round. This enables to repeat the argument inductively to show that indeed $\Omega(t^2)$ messages must be sent in each of f rounds of some execution with $f \leq t/2$ faults. Bounding $f \leq t/2$ here ensures that still potentially $\Omega(t)$ additional nodes may fail, which makes it expensive (in terms of messages) for the algorithm to, if necessary, prove to all nodes that there have been further faults. □

We need to capture the properties we require from executions in order to follow the approach outlined above. We call the node v_r that is "pivotal" in round r *critical*, since failing it can delay the decision process further. The definition is phrased such that v_r must act as the arbiter that suggests an output of the execution and therefore determines the output provided faulty nodes do not interfere further. For an $(f+1)$-deciding algorithm, this must happen exactly in round r, since round $r+1$ must be used to check whether v_r failed in round r and suggested different outputs to different subsets of nodes.

Definition 3.3 (Critical Executions). *For $r \in \mathbb{N}$, a pair of executions $\mathcal{E}_0, \mathcal{E}_1$ of a binary consensus algorithm $\mathcal{A}$ is called* r-critical *if there is a* critical node v_r *such that the following properties are satisfied:*

send \ receive	F_r	W	$V \setminus (W \cup F_r)$
F_r	(faulty)	as in $\mathcal{E}_0^{(r)}$	as in $\mathcal{E}_1^{(r)}$
W	as in $\mathcal{E}_0^{(r)}$ and $\mathcal{E}_1^{(r)}$		
$V \setminus (W \cup F_r)$			

(a) Messages sent in round r of execution $\mathcal{E}(W)$.

send \ receive	F_r	W	$V \setminus (W \cup F_r)$
F_r	(faulty)	as in $\mathcal{E}_0^{(r)}$	as in $\mathcal{E}_1^{(r)}$
W	as in $\mathcal{E}_0^{(r)}$		
$V \setminus (W \cup F_r)$	as in $\mathcal{E}_1^{(r)}$		

(b) Messages sent in round $r+1$ of execution $\mathcal{E}(W)$.

Figure 1: Rounds r and $r+1$ of execution $\mathcal{E}(W)$. Nodes in $V \setminus F_r$ cannot distinguish $\mathcal{E}_0^{(r)}$, $\mathcal{E}_1^{(r)}$, and $\mathcal{E}(W)$ before round r and therefore send the same messages in round r of all three executions. If $W' = W \dot\cup \{v_{r+1}\}$, the only non-faulty node that can distinguish $\mathcal{E}(W)$ and $\mathcal{E}(W')$ in round r is v_{r+1}, and only through its messages in round $r+1$ other non-faulty nodes can distinguish these executions in round $r+1$.

- *v_r is non-faulty in both executions,*
- *the restrictions of $\mathcal{E}_0$ and $\mathcal{E}_1$ to the first $r-1$ rounds differ only at $v_r \in V$,*
- *in round r, all pairs of nodes not involving v_r exchange the same messages in both executions,*
- *no nodes fail in round $r' \geq r$ of either execution,*
- *nodes in $F_{r-1}^{\mathcal{E}_0} = F_{r-1}^{\mathcal{E}_1}$ do not send messages in rounds $r' > r$,*
- *there are $r-1$ faulty nodes in both executions,*
- *in $\mathcal{E}_0$ non-faulty nodes will output 0, and*
- *in $\mathcal{E}_1$ non-faulty nodes will output 1.*

We are going to show that for such executions, an $(f+1)$-deciding algorithm is forced to exchange many messages in round $r+1$, as otherwise some nodes would have to decide on an output without being certain that there are no other nodes taking the opposite decision in some execution. At the same time, we will be able to fail the critical node in a way that maintains that many messages are sent, yet keeps the output of the execution uncertain.

The construction proceeds inductively. The induction anchor is given by the well-known fact that for any (binary) consensus algorithm, there is a way to assign the inputs such that flipping just one of them will change the output of the algorithm [10].

LEMMA 3.4. *Let $\mathcal{A}$ be any binary consensus algorithm. Then there is a critical pair of 1-round executions of $\mathcal{A}$.*

PROOF. A 1-round execution is fully specified by the set of non-faulty nodes and their inputs, plus the messages sent by faulty nodes in round 1. For each input assignment $B : V \to \{0,1\}$ define $o(B)$ as the output in the unique execution without faults. By the validity property, $o(v \mapsto 0) = 0$ and $o(v \mapsto 1) = 1$. Hence there must be some inputs B and B' that differ only at a single node v_1, such that $o(B) = 0$ and $o(B') = 1$. Trivially, the corresponding pair of executions is indistinguishable before round 1 at all nodes except v_1, as $H_v(1)$ is, by definition, empty for all nodes $v \in V$ and $B(v) = B'(v)$ for all $V \setminus \{v_1\}$. By designating v_1 as the critical node, the definition of a critical pair is thus met by the two executions defined by B and B' (where we set $F_0 := \emptyset$ for both executions). □

In the following, we fix an $(f+1)$-deciding binary consensus algorithm $\mathcal{A}$ and denote by $\mathcal{E}_0^{(1)}$, $\mathcal{E}_1^{(1)}$ a critical pair of 1-round executions of $\mathcal{A}$. The hypothesis of the induction we want to perform is that, for some $r \in \{1, \dots, f\}$ with $f \leq t/2$, a critical pair $\mathcal{E}_0^{(r)}$, $\mathcal{E}_1^{(r)}$ of r-round executions of $\mathcal{A}$ exists satisfying that $\Omega(t^2(r-1))$ messages are sent by the non-faulty nodes in both executions. To complete the induction, it thus suffices to construct a critical pair of $(r+1)$-round executions $\mathcal{E}_0^{(r+1)}$, $\mathcal{E}_1^{(r+1)}$ and show that in round $r+1$ of these executions, $\Omega(t^2)$ messages are sent by non-faulty nodes.

The construction itself is fairly straightforward; the harder part will be to show the lower bound on the number of sent messages. We first show that we indeed can construct critical pairs of $(r+1)$-round executions from critical pairs of r-round executions.

LEMMA 3.5. *Suppose for $1 \leq r \leq t$ we are given an r-critical pair of executions $\mathcal{E}_0^{(r)}$, $\mathcal{E}_1^{(r)}$. Then an $(r+1)$-critical pair of executions exists.*

PROOF. The proof is similar in spirit to the reasoning for Lemma 3.4, where the messages sent by the critical node v_r of $\mathcal{E}_0^{(r)}$, $\mathcal{E}_1^{(r)}$ in round r take the role of the "inputs". Denote for each node $v \in V \setminus \{v_r\}$ by $m_0^{(r')}(v)$ and $m_1^{(r')}(v)$ the messages v_r sends to v in round $r' \in \mathbb{N}$ of $\mathcal{E}_0^{(r)}$ and $\mathcal{E}_1^{(r)}$, respectively.

Recall that $F_{r-1}^{\mathcal{E}_0^{(r)}} = F_{r-1}^{\mathcal{E}_1^{(r)}}$ for critical pairs; thus we may simply write F_{r-1}, F_r, etc. in the following. We set $F_r := F_{r-1} \cup \{v_r\}$, which is feasible since $|F_{r-1}| = r-1 < t$. For any set $W \subseteq V \setminus F_r$, define $\mathcal{E}(W)$ as follows. Take an r-round execution that all non-faulty nodes can distinguish from neither $\mathcal{E}_0^{(r)}$ nor $\mathcal{E}_1^{(r)}$ except by the message v_r sends to them in round r (such an execution exists by the definition of r-critical pairs). We rule that this message is $m_0^{(r)}(v)$ in case $v \in W$ and $m_1^{(r)}(v)$ if $v \in V \setminus (F_r \cup W)$ (see Figure 1a). Likewise, in round $r+1$ (see Figure 1b) it will send $m_0^{(r+1)}(v)$ respectively $m_1^{(r+1)}(v)$ to v;[4] it does not send any messages in rounds $r' > r+1$. Other faulty nodes do not send messages in rounds $r' > r$. Observe that this fully specifies the messages sent from faulty nodes to non-faulty nodes; as non-faulty nodes follow the deterministic algorithm $\mathcal{A}$, we

[4] This ensures that it does not give away to non-faulty nodes in W (resp. $V \setminus (W \cup F_r)$) that the execution is different from $\mathcal{E}_0^{(r)}$ (resp. $\mathcal{E}_1^{(r)}$).

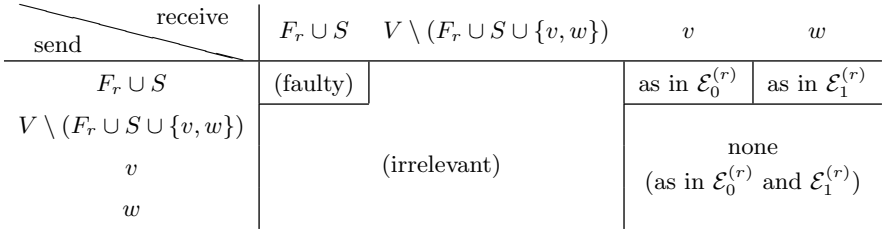

Figure 2: Round $r+1$ of the contradictive execution constructed in Lemma 3.7, which is indistinguishable from $\mathcal{E}_0^{(r)}$ and $\mathcal{E}_1^{(r)}$ before round $r+1$. If too few messages are sent in $\mathcal{E}_0^{(r)}$ and $\mathcal{E}_1^{(r)}$ in round $r+1$, we can find a small set S that might fail, permitting to construct the shown execution.

thus obtain $\mathcal{E}(W)$ as the unique extension of the r-round execution we started with in which no further nodes become faulty and faulty nodes send the messages we just specified.

Observe that, by construction, $\mathcal{E}(V \setminus F_r)$ and $\mathcal{E}(\emptyset)$ cannot be distinguished from $\mathcal{E}_0^{(r)}$ and $\mathcal{E}_1^{(r)}$, respectively, by non-faulty nodes before round $r+2$. Hence, non-faulty nodes must decide by the end of round $r+1$ in these executions since $|F_{r-1}| = r-1$ and $\mathcal{A}$ is $(f+1)$-deciding. Since the output value of $\mathcal{E}_0^{(r)}$ is 0, the same must hold true for $\mathcal{E}(V \setminus F_r)$; analogously, $\mathcal{E}(\emptyset)$ outputs 1. Consequently, there must be some $W \subset V \setminus F_r$ and $v_{r+1} \in V \setminus (W \cup F_r)$ such that $\mathcal{E}(W)$ and $\mathcal{E}(W \cup \{v_{r+1}\})$ have outputs 1 and 0, respectively. In both executions the set of faulty nodes is F_r, there are no new faults in rounds $r' \geq r+1$, and, since v_r is faulty, the only non-faulty node that can distinguish between the two executions in round r (or earlier) is v_{r+1}. Therefore, all non-faulty nodes except v_{r+1} send the same messages in round $r+1$ of both executions. The same is true for faulty nodes, with the exception of the message from v_r to v_{r+1}: all faulty nodes except v_r do not send messages, and we defined the executions such that v_r behaves identical in round $r+1$ towards all nodes but v_{r+1}. Finally, no faulty node sends any messages in rounds $r' > r+1$. Note that in the $(r+1)$-critical pair, v_{r+1} is (still) non-faulty. Overall, $\mathcal{E}(W \cup \{v_{r+1}\})$ and $\mathcal{E}(W)$ are an $(r+1)$-critical pair with critical node v_{r+1}, completing the proof. □

An observation that can be derived immediately is that the time complexity lower bound of $\min\{f+2, t+1\}$ rounds also applies to $(f+1)$-deciding algorithms. This is stronger than the result from [7, 8] which only applies if nodes must also stop sending messages when deciding, but also requires more restrictive assumptions, as [7, 8] holds for crash faults (cf. Definition 4.1). This is of particular interest as with crash faults it is possible to achieve that all nodes decide within $f+1$ rounds [24].

THEOREM 3.6. *For any consensus protocol and any f, $0 \leq f \leq t-1$, there are executions where some non-faulty node decides in or after round $f+2$.*

PROOF. By inductive application of Lemma 3.5, anchored by Lemma 3.4, we obtain an $(f+1)$-critical pair of executions $\mathcal{E}_0^{(f+1)}$, $\mathcal{E}_1^{(f+1)}$. In both executions, the same set F_f of $f < t$ nodes fails. As in Lemma 3.5, failing v_{f+1} (in round $f+1$) permits to create an execution $\mathcal{E}(W)$ (for some $\emptyset \neq W \subset V \setminus (F_f \cup \{v_{f+1}\})$) that is indistinguishable before round $f+2$ from $\mathcal{E}_0^{(f+1)}$ by nodes in W and from $\mathcal{E}_1^{(f+1)}$ by nodes in $V \setminus (W \cup F_f \cup \{v_{f+1}\})$. Hence, assuming that both in $\mathcal{E}_0^{(f+1)}$ and $\mathcal{E}_1^{(f+1)}$ all non-faulty nodes decide in or before round $f+1$, the same would be true for $\mathcal{E}(W)$. However, nodes in $W \neq \emptyset$ would decide 0 (as in $\mathcal{E}_0^{(f+1)}$) and nodes in $V \setminus (W \cup F_f \cup \{v_{f+1}\}) \neq \emptyset$ would decide 1 (as in $\mathcal{E}_1^{(f+1)}$), violating the agreement property. Hence, in one of the executions $\mathcal{E}_0^{(f+1)}$ and $\mathcal{E}_1^{(f+1)}$, which both have f faults, there must be a non-faulty node that decides in round $f+2$ or later. □

Note that a similar result was proven by Keidar and Rajsbaum [15] and Charron-Bost and Schiper [2] for uniform consensus.

The construction from Lemma 3.5 leaves some flexibility. Depending on the algorithm, there might be many choices of W and v_{r+1}. Our task is now to identify one that ensures a large number of messages exchanged. Our argument will proceed in two steps. First we will show that many messages are sent in round $r+1$ of at least one of the executions $\mathcal{E}_0^{(r)}$ and $\mathcal{E}_1^{(r)}$. Then we will use this information to identify a set W_0 of size $\lceil t/2 \rceil$ that sends many messages (say, in $\mathcal{E}_0^{(r)}$ and thus also $\mathcal{E}(V \setminus F_r)$) and show that $\mathcal{E}(W_0)$ cannot have output 0. A critical pair then can be obtained as in the proof of Lemma 3.5 by inductively adding nodes to W_0, since $\mathcal{E}(V \setminus F_r)$ outputs 0. The lower bound on the number of sent messages then follows since the nodes in W_0 must send the same messages in round $r+1$ as they do in $\mathcal{E}_0^{(r)}$, since they cannot distinguish these executions before round $r+2$.

The next lemma deals with showing that many messages are sent in $\mathcal{E}_0^{(r)}$ or $\mathcal{E}_1^{(r)}$.

LEMMA 3.7. *Suppose we are given a pair of r-critical executions $\mathcal{E}_0^{(r)}$, $\mathcal{E}_1^{(r)}$ for some $1 \leq r \leq t/2$. Then in at least one of the two executions, at least $nt/22$ messages are sent by non-faulty nodes in round $r+1$.*

PROOF. We use the notation from Lemma 3.5. Recall that whether some non-faulty node sends a message to some other node in round $r+1$ of execution $\mathcal{E}(W)$ for any $W \subseteq V \setminus F_r$ solely depends on the message it receives from node v_r in round r of $\mathcal{E}(W)$ (or the fact that it does not receive one). Hence, node $v \in V \setminus F_r$ behaves identically as in $\mathcal{E}_0^{(r)}$ (if $v \in W$) or in $\mathcal{E}_1^{(r)}$ (if $v \in V \setminus (W \cup F_r)$).

Assume for contradiction that fewer than $nt/11 < n^2/33$ messages are sent in total in both executions by non-faulty nodes. Since there are at least $|V \setminus F_r| \cdot (|V \setminus F_r| - 1)/2 > 2n^2/9$ pairs of non-faulty nodes, for more than $2n^2/9 - n^2/33 > 2n^2/11$ such pairs $\{v, w\}$ it holds that v and w exchange messages in neither $\mathcal{E}_0^{(r)}$ nor $\mathcal{E}_1^{(r)}$. Hence, they do not exchange messages in $\mathcal{E}(W)$ either, irrespectively of the

send \ receive	F_r	W_0	v	$V \setminus (F_r \cup W_0 \cup \{v\})$
F_r	(faulty)	as in $\mathcal{E}(W_0)$ and $\mathcal{E}_1^{(r)}$		
W_0	as in $\mathcal{E}(W_0)$			
v	as in $\mathcal{E}(W_0)$ and $\mathcal{E}_1^{(r)}$			
$V \setminus (F_r \cup W_0 \cup \{v\})$				

Figure 3: Round $r+1$ of execution $\mathcal{E}_0$, which is identical to round $r+1$ of $\mathcal{E}(W_0)$. See Lemma 3.5 and Figure 1 for the definition of $\mathcal{E}(W_0)$.

send \ receive	F_r	W_0	v	$V \setminus (F_r \cup W_0 \cup \{v\})$
F_r	(faulty)	as in $\mathcal{E}(W_0)$ and $\mathcal{E}_1^{(r)}$		
W_0		as in $\mathcal{E}_1^{(r)}$	as in $\mathcal{E}(W_0)$	
v	as in $\mathcal{E}(W_0)$ and $\mathcal{E}_1^{(r)}$			
$V \setminus (F_r \cup W_0 \cup \{v\})$				

Figure 4: Round $r+1$ of execution $\mathcal{E}_1$. Node v cannot distinguish $\mathcal{E}_1$ from $\mathcal{E}_1^{(r)}$ in round $r+1$. Since $\mathcal{E}_1^{(r)}$ has $r-1$ faults and outputs 1, v must thus decide on 1 at the end of round $r+1$ of $\mathcal{E}_1$.

choice of W.

We claim among these pairs there are $v, w \in V \setminus F_r$ which, summed over both nodes and executions, receive at most $t/2$ messages from non-faulty nodes in $\mathcal{E}_0^{(r)}$ nor $\mathcal{E}_1^{(r)}$. Otherwise, summing over all pairs (that do not exchange messages) and both executions, we had at least $n^2t/11$ received messages, where for each individual execution and node we have counted a message up to $|V \setminus F_r| - 1 < n$ times. This makes for a total of more than $nt/11$ messages, which by assumption is not reached. Consequently, the claim holds and we have a pair $\{v, w\}$ that is non-faulty, does not exchange messages in $\mathcal{E}(W)$, and receives at most $t/2$ messages from non-faulty nodes in $\mathcal{E}(W)$, independently of the specific choice of W.

We fix some W such that $v \in W$ and $w \notin W$. Denote by $S \subset V \setminus F_r$ the set of non-faulty nodes that sends messages to v or w in round $r+1$ of $\mathcal{E}_0^{(r)}$ or $\mathcal{E}_1^{(r)}$. By the choice of v and w, we have that $|S| \leq t/2$. Since in $\mathcal{E}(W)$ the number of faulty nodes is $r \leq t/2$, we can construct an $(r+1)$-round execution that is identical to $\mathcal{E}(W)$ before round $r+1$, while in round $r+1$ (see Figure 2) all nodes in S fail. Faulty nodes send the same messages as in round $r+1$ of $\mathcal{E}_0^{(r)}$ to v and the same messages as in round $r+1$ of $\mathcal{E}_1^{(r)}$ to w (the remaining messages are chosen arbitrarily). By construction, v and w cannot distinguish this execution from $\mathcal{E}_0^{(r)}$ and $\mathcal{E}_1^{(r)}$, respectively, before round $r+2$. Thus, as in $\mathcal{E}_0^{(r)}$ and $\mathcal{E}_1^{(r)}$ only $r-1$ nodes fail and the algorithm is $(f+1)$-deciding, v and w must decide on 0 and 1, respectively. This violates the agreement property, implying that the assumption that fewer than $nt/11$ messages are sent by non-faulty nodes in round $r+1$ of $\mathcal{E}_0^{(r)}$ and $\mathcal{E}_1^{(r)}$ together must be wrong. We conclude that indeed in one of the executions, at least $nt/22$ messages are sent in round $r+1$. □

We now move on to the last step in the proof of Theorem 3.2, which consists of showing that in the construction from Lemma 3.5, we can choose a critical pair for which many messages are sent in round $r+1$.

LEMMA 3.8. *Suppose for $1 \leq r \leq t/2$ we are given an r-critical pair of executions $\mathcal{E}_0^{(r)}$, $\mathcal{E}_1^{(r)}$. Then an $(r+1)$-critical pair of executions exists where the non-faulty nodes—excluding v_{r+1}—send at least $t^2/44$ messages in round $r+1$ of both executions.*

PROOF. Again, we use the notation from Lemma 3.5. By Lemma 3.7, in round $r+1$ of at least one of the executions $\mathcal{E}_0^{(r)}$ and $\mathcal{E}_1^{(r)}$, $nt/22$ messages are sent by non-faulty nodes. Suppose w.l.o.g. that this holds for $\mathcal{E}_0^{(r)}$ (the other case is symmetrical). Order the nodes $V \setminus F_r$ in descending order according to the number of messages they send in round $r+1$ of $\mathcal{E}_0^{(r)}$, and label them by $v(1), \ldots, v(n-r)$ according to this order. Clearly, the nodes $v(1), \ldots, v(\lceil t/2 \rceil)$ will together send at least

$$\frac{t/2}{n-r} \cdot \frac{nt}{22} > \frac{t^2}{44} \tag{1}$$

messages in round $r+1$ of $\mathcal{E}_0^{(r)}$.

Denote $W_0 := \{v(1), \ldots, v(\lceil t/2 \rceil)\}$. We claim that $\mathcal{E}(W_0)$ has output 1. Assuming otherwise, in $\mathcal{E}(W_0)$ all non-faulty nodes decide 0 by the end of round $r+2$. We will construct two executions $\mathcal{E}_0$ and $\mathcal{E}_1$ that cannot be distinguished by a node that is non-faulty in both executions, yet have conflicting outputs. Both executions are identical to $\mathcal{E}(W_0)$ before round $r+1$. In round $r+1$ of $\mathcal{E}_1$ (see Figure 4), all nodes in W_0 fail, which is feasible since $r \leq t/2$ and $|W_0| = \lceil t/2 \rceil$. In this round of $\mathcal{E}_1$, faulty nodes behave like in $\mathcal{E}(\emptyset)$ towards some fixed non-faulty node $v \in V \setminus (W_0 \cup F_r)$, but like in $\mathcal{E}(W_0)$ towards all remaining nodes. Hence, v cannot distinguish $\mathcal{E}_1$ from $\mathcal{E}(\emptyset)$ (and therefore also not from $\mathcal{E}_1^{(r)}$ with only $r-1$ faults) before round $r+2$, and must decide 1 by the end of round $r+1$. We define round $r+1$ of $\mathcal{E}_0$ to be identical to round $r+1$ of $\mathcal{E}(W_0)$ (see Figure 3); in particular, we do not fail any additional nodes in this round of $\mathcal{E}_0$.

Observe that $\mathcal{E}_0$ and $\mathcal{E}_1$ are indistinguishable from each

send \ receive	F_r	W_0	v	$V \setminus (F_r \cup W_0 \cup \{v\})$
F_r	(faulty)		(faulty)	
W_0	as in $\mathcal{E}(W_0)$			
v	(faulty)		(faulty)	as in $\mathcal{E}_1^{(r)}$
$V \setminus (F_r \cup W_0 \cup \{v\})$				

Figure 5: Round $r+2$ of execution $\mathcal{E}_0$. W_0 cannot distinguish this execution from $\mathcal{E}(W_0)$ in this round. Since $\mathcal{E}(W_0)$ has r faults, nodes in W_0 must thus decide on the same value in $\mathcal{E}_0$ and $\mathcal{E}(W_0)$ by the end of round $r+2$.

send \ receive	F_r	W_0	v	$V \setminus (F_r \cup W_0 \cup \{v\})$
F_r	(faulty)		(irrelevant)	as in $\mathcal{E}(W_0)$
W_0				
v	as in $\mathcal{E}_1^{(r)}$			
$V \setminus (F_r \cup W_0 \cup \{v\})$	as in $\mathcal{E}(W_0)$			

Figure 6: Round $r+2$ of execution $\mathcal{E}_1$. Figures 3–6 show that nodes in $V \setminus (F_r \cup W_0 \cup \{v\})$ cannot distinguish $\mathcal{E}_0$ and $\mathcal{E}_1$ before round $r+3$.

other and from $\mathcal{E}(W_0)$ at all nodes in $V \setminus (W_0 \cup \{v\} \cup F_r)$ before round $r+2$ by construction. Consequently, these nodes send the same messages in round $r+2$ of all three executions. We let the nodes in F_r send no messages in rounds $r' \geq r+2$ of executions $\mathcal{E}_0$ and $\mathcal{E}_1$, just like in $\mathcal{E}(W_0)$. In round $r+2$ of $\mathcal{E}_0$ (see Figure 5), we fail node v, and make it send the same messages as in round $r+2$ of $\mathcal{E}(W_0)$ to the nodes in W_0. In $\mathcal{E}_0$, the (non-faulty) nodes in W_0 will send the same messages as in $\mathcal{E}(W_0)$ in this round as well, since $\mathcal{E}_0$ is identical to $\mathcal{E}(W_0)$ before round $r+2$. Consequently, the nodes in W_0 cannot distinguish $\mathcal{E}_0$ from $\mathcal{E}(W_0)$ (with r faults) before round $r+3$ and must decide 0 by the end of round $r+2$ in $\mathcal{E}_0$. Towards the nodes in $V \setminus (W_0 \cup F_r \cup \{v\})$, v mimics its behavior in round $r+2$ of $\mathcal{E}_1$ (where it is non-faulty and thus computes its messages according to the algorithm). In turn, in round $r+2$ of $\mathcal{E}_1$ (see Figure 6) the nodes in W_0 mimic their behavior in execution $\mathcal{E}_0$. Hence, nodes in $V \setminus (W_0 \cup \{v\} \cup F_r)$ receive the same messages from all other nodes in round $r+2$ of both $\mathcal{E}_0$ and $\mathcal{E}_1$.

In summary, at the end of round $r+2$, we have that (i) in $\mathcal{E}_0$ the non-faulty node v decided 0, (ii) in $\mathcal{E}_1$ the non-faulty nodes in W_0 decided 1, and (iii) no node in $V \setminus (W_0 \cup \{v\} \cup F_r)$ can distinguish between $\mathcal{E}_0$ and $\mathcal{E}_1$ before round $r+3$. Therefore, in order to force an erroneous decision by some non-faulty node (in one of the executions), it is sufficient to extend $\mathcal{E}_0$ and $\mathcal{E}_1$ without further faults so that nodes in $V \setminus (W_0 \cup \{v\} \cup F_r) \neq \emptyset$ cannot distinguish between the two executions. This is done by induction on $r' \geq r+3$, where the hypothesis is that nodes in $V \setminus (W_0 \cup \{v\} \cup F_r)$ cannot distinguish $\mathcal{E}_0$ and $\mathcal{E}_1$ before round r'. The statement holds for $r' = r+3$ by the previous observations. By the hypothesis, nodes in $V \setminus (W_0 \cup \{v\} \cup F_r)$ will send the same messages in round r' of both executions, as is true for F_r which sends no messages. With respect to W_0 and v, recall that these nodes are non-faulty in only one of the two executions. In the execution where they are faulty, we thus rule that they simply send the same messages as their non-faulty counterparts in the respective other execution. It follows that nodes in $V \setminus (W_0 \cup \{v\} \cup F_r)$ receive the same messages in round r' of both executions and thus cannot distinguish between them before round $r'+1$. This concludes the induction.

As non-faulty nodes eventually decide, each node in $V \setminus (W_0 \cup \{v\} \cup F_r)$, which must behave identically in executions $\mathcal{E}_0$ and $\mathcal{E}_1$, will violate the agreement property in at least one of the executions due to (i) and (ii). It follows that the assumption that $\mathcal{E}(W_0)$ outputs 0 must be wrong. From here, we proceed as in Lemma 3.5 by adding nodes to W_0 one by one until we finally end up with an $(r+1)$-critical pair of executions $\mathcal{E}(W \cup \{v_{r+1}\})$, $\mathcal{E}(W)$ satisfying that $W_0 \subseteq W$. Finally, because the nodes in W_0 will send the same messages in round $r+1$ of $\mathcal{E}(W \cup \{v_{r+1}\})$ and $\mathcal{E}(W)$ as in round $r+1$ of $\mathcal{E}(V \setminus F_r)$ and thus also $\mathcal{E}_0^{(r)}$, Inequality (1) shows that the constructed $(r+1)$-critical pair meets the requirements of the lemma. □

Theorem 3.2 now follows by straightforward application of the derived results.

Proof of Theorem 3.2. Fix any $(f+1)$-deciding binary consensus algorithm $\mathcal{A}$. By Lemma 3.4, there is a 1-critical pair of executions of $\mathcal{A}$. We inductively apply Lemma 3.8 $f \leq t/2$ times to obtain an $(f+1)$-critical pair of executions, where in all rounds $2, \ldots, f+1$, at least $t^2/44$ messages are sent by non-faulty nodes; here we make use of the fact that when applying Lemma 3.8, in both executions of the constructed $(r+1)$-critical pair, all non-faulty nodes behave identically in round $r+1$ as they do in the r-critical pair we started from. Since in the constructed executions f nodes fail, we obtain the desired worst-case lower bound of $t^2 f/44$ on the number of messages sent by non-faulty nodes in executions with f faults. □

Taking into account the lower bound from [6], we arrive at the following result.

Corollary 3.9. *For any $(f+2)$-deciding consensus algorithm $\mathcal{A}$ that tolerates t faults and each $0 \leq f \leq t$, there are executions of $\mathcal{A}$ in which non-faulty nodes send $\Omega(t^2 f + nt)$ messages.*

4. CRASH FAULTS

In this section, we focus on the following restricted fault model.

DEFINITION 4.1 (CRASH FAULTS). *Node $v \in F_r^{\mathcal{E}} \setminus F_{r-1}^{\mathcal{E}}$* crashes *in round r of execution $\mathcal{E}$, if there is a subset $W \setminus V$ of the nodes such that the following holds.*

- *In round r, v sends the same message to each $w \in W$ as it would have if it was non-faulty.*
- *In round r, v sends no messages to nodes $w \in V \setminus W$.*
- *In rounds $r' > r$, v sends no messages.*

A consensus algorithm is resilient to t crash faults, *if it satisfies termination, agreement, and validity in all executions with at most t crashing nodes and no other faults.*

Crash faults are much easier to handle and permit $(f+1)$-deciding algorithms [2]. We will show now that $(f+1)$-deciding algorithms essentially require all non-faulty nodes to exchange messages in each round of the algorithm.

LEMMA 4.2. *Let $\mathcal{E}_0^{(r)}$, $\mathcal{E}_1^{(r)}$ be an r-critical pair of executions of an $(f+1)$-deciding algorithm $\mathcal{A}$ resilient to $t > r$ crash faults, where $t \leq n-2$. Then an $(r+1)$-critical pair of executions of $\mathcal{A}$ exists, satisfying that in both executions at least $(n-r-1)^2$ messages are sent in round $r+1$ by the nodes in $V \setminus (F_r \cup \{v_{r+1}\})$.*

PROOF. Set $F_r := F_{r-1} \cup \{v_r\}$. First, we are going to show that non-faulty nodes will have to send a lot of messages in round $r+1$, no matter when exactly v_r crashes. Afterwards it will be simple to construct an $(r+1)$-critical execution in which many messages are sent just as in Lemma 3.5.

Consider the execution $\mathcal{E}_1$ in which v_r crashes before sending any message in round r and no further faults occur. Suppose w.l.o.g. that $\mathcal{E}_1$ outputs 1 (the other case is symmetrical). No further faults happen in $\mathcal{E}_1$, so it is a valid execution as $r < t$. Moreover, all live nodes must decide 1 by the end of round $r+1$.

For each $v \in V \setminus F_r$, we define an execution $\mathcal{E}_0(v)$ as follows. $\mathcal{E}_0(v)$ is identical to $\mathcal{E}_0^{(r)}$ before round r. In round r, v_r crashes, successfully sending only its message to v. No further faults happen in $\mathcal{E}_0(v)$, thus it is a valid execution. Observe that v cannot distinguish $\mathcal{E}_0(v)$ from $\mathcal{E}_0^{(r)}$, which has $r-1$ faults, and thus must decide 0 at the end of round r of $\mathcal{E}_0(v)$. No further faults occur in $\mathcal{E}_0(v)$, implying that by the end of round $r+1$, all live nodes must decide. Since v already decided 0 and does not crash, it follows that the output of all nodes must be 0.

We claim that in all these executions, each non-faulty node sends a message to each other non-faulty node. Assume for the sake of contradiction that there is a pair of nodes $v, w \in V \setminus F_r$ such that v does not send a message to w in $\mathcal{E}_0(v)$ or $\mathcal{E}_1$. We examine first the case where v does not send a message to w in $\mathcal{E}_1$. Consider the execution that is identical to $\mathcal{E}_0(v)$ before round $r+1$, while in round $r+1$ node v crashes, sending all its messages except for the one to w; this is feasible since $r+1 \leq t$. Clearly, no node but w can distinguish this execution from $\mathcal{E}_0(v)$ before round $r+2$. Hence, as $t \leq n-2$, it holds that $V \setminus (F_r \cup \{v, w\}) \neq \emptyset$, implying that there is some node that decides 0 by the end of round $r+1$ of this execution. However, since w does not receive a message from v, just like in $\mathcal{E}_1$, it must decide 1 by the end of round $r+1$ of this execution, violating agreement. We conclude that indeed v must send a message to w in $\mathcal{E}_1$. Analogously, if v does not send a message to w in round $r+1$ of $\mathcal{E}_0(v)$, we can make it fail in round $r+1$ of $\mathcal{E}_1$ to enforce violation of the agreement property.

From here we argue analogously to Lemma 3.5. We consider executions $\mathcal{E}(W)$ that are identical to $\mathcal{E}_0^{(r)}$ and $\mathcal{E}_1$ before round r, where v_r crashes in round r such that node $w \in V \setminus F_r$ receives the message from v_r exactly if $w \in W \subseteq V \setminus F_r$. Starting from the empty set, adding nodes to W one by one will eventually lead to a critical pair, since $\mathcal{E}(\emptyset) = \mathcal{E}_1$ outputs 1 and $\mathcal{E}(V \setminus F_r)$ outputs 0. Because in all these executions, each non-faulty node must send the same messages as in $\mathcal{E}_0^{(r)}$ or $\mathcal{E}_1$, we conclude that at least $|V \setminus F_r|(|V \setminus F_r| - 1) = (n-r)(n-r-1)$ messages are sent in both executions of the critical pair. Not counting the messages sent by the critical node v_{r+1}, we have at least $(n-r-1)^2$ messages. □

Applying this lemma inductively, we arrive at the following bound.

THEOREM 4.3. *Let $\mathcal{A}$ be an $(f+1)$-deciding algorithm resilient to $t > 1$ crash faults. Then for each $f < t$ there is an execution of $\mathcal{A}$ with f faults in which $nf(n-f) + f^3/3 - O(nf) \subset \Omega(n^2 f)$ messages are sent.*

PROOF. W.l.o.g. assume that $t \leq n-2$. By Lemma 3.4, there is a 1-critical pair of executions of $\mathcal{A}$ (since this execution has no faults, the lemma applies also to the restricted fault model). We inductively apply Lemma 4.2 f times, resulting in an execution with f faults in which at least

$$\begin{aligned} & \sum_{i=1}^{f} (n-i-1)^2 \\ = \; & \frac{(n-2)(n-1)(2n-3)}{6} \\ & - \frac{(n-f-3)(n-f-2)(2(n-f)-5)}{6} \\ \in \; & \frac{6n^2 f - 6nf^2 + 2f^3 + 6n^2 - \mathcal{O}(nf)}{6} \\ \subseteq \; & nf(n-f) + \frac{f^3}{3} - \mathcal{O}(nf) \end{aligned}$$

messages are sent.[5] □

Orderly Crash Faults

Interestingly, there is a straightforward algorithm that is much more efficient if the fault model is constrained further in that nodes may choose the order in which they (attempt to) send messages in a given round.

DEFINITION 4.4 (ORDERLY CRASHES). *Assume that an algorithm $\mathcal{A}$ specifies for each round and node also in which order the node sends its messages to other nodes. Denoting for some round and a crashing node v this sequence by $v_1, \ldots, v_{n-1}$, the crash is* orderly *if the subset W of nodes*

[5] Like in the proof of Theorem 3.2, we exploit here that applying Lemma 4.2 will, in round r, change only the behavior of node v_r in comparison to the previous critical pair, and therefore still many messages are sent in round r of the new pair.

receiving messages from v is a prefix of this sequence. An algorithm is resilient to t orderly crashes*, if termination, agreement, and validity hold in any execution where there are at most t faulty nodes, all of which crash orderly.*

It should be noted that the lower bound of $\min\{f+2, t+1\}$ on the number of rounds until all nodes stop applies even if crashes are orderly [26]. In this work it is also shown that $(f+1)$-stopping becomes feasible if one further allows that nodes may send multiple messages to the same recipient in one round, as confirmation messages may be used to prove that all other nodes have received a previously sent batch of messages.

The following variant of the simple (not early-stopping) crash-tolerant algorithm given in [24] that sends $\mathcal{O}(nt)$ messages is $(f+1)$-deciding and resilient to t orderly crashes.

1. If node $i \in V$ receives a message containing value b, it decides b.
2. If node $i \in \{1, \ldots, t+1\}$ does not receive a message before round i, it decides on its input value at the beginning of round i (i.e., before sending messages).
3. If node $\{1, \ldots, t+1\}$ decides on value b at the beginning of round r or at the end of round $r-1$, it sends b to all nodes $i+1, \ldots, n$ (in this order) in round r.

THEOREM 4.5. *The above algorithm is resilient to t orderly crash faults, is $(f+1)$-deciding and $(f+2)$-stopping, and sends at most $(n-t/2-1)(t+1) < n(t+1)$ messages.*

PROOF. **Decision and Stopping:** Denote by $i \leq f+1$ the node with smallest ID that does not fail. It will decide on and send some value b to all nodes $i' > i$ at the latest in round $f+1$. Hence all live nodes decide by the end of round $f+1$ and stop by the end of round $f+2$.

Correctness: If node $i \in V$ decides b before round i, it has received b from some other node $i' < i$ that already decided b. If node i decides in round i, it decides on its input and all nodes $i' < i$ must have crashed (otherwise i would have received a message before round i). Thus, there is a unique node i_0 deciding in round i_0 on its input, all nodes $i < i_0$ crash, and all nodes $i > i_0$ decide on the same value as i_0.

Message complexity: Nodes $i \in \{1, \ldots, t+1\}$ send up to $n-i$ messages, while nodes $i \in \{t+2, \ldots, n\}$ never send a message. □

5. OPEN PROBLEMS

We conclude with a brief list of open problems regarding the trade-offs between early decision and message complexity.

- Can we have $(f+\mathcal{O}(1))$-deciding algorithms that have optimal resilience and use $\mathcal{O}(nt)$ messages?
- Are $(f+\mathcal{O}(1))$-deciding algorithms possible that are resilient to crash or omission faults and send $o(nt)$ messages for all n and t?
- Can we show strong lower bounds beyond $(f+1)$- and $(f+2)$-decision in any of the models?
- Is the message complexity of the $(f+1)$-deciding algorithm resilient to orderly crashes that we present asymptotically optimal?
- Of what use is cryptography to early-deciding algorithms, in terms of the trade-off between round and message complexity?
- Can randomized algorithms be made more efficient in runs with few faults, in terms of the trade-off between round and message complexity?

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant Nos. CCF-AF-0937274, CNS-1035199, 0939370-CCF and CCF-1217506, the AFOSR under Contract No. AFOSR Award number FA9550-13-1-0042, the Swiss National Science Foundation (SNSF), the Swiss Society of Friends of the Weizmann Institute of Science, the German Research Foundation (DFG, reference number Le 3107/1-1), the Israeli Centers of Research Excellence (I-CORE) program, (Center No. 4/11), grant 3/9778 of the Israeli Ministry of Science and Technology, and the Google Inter-university center for "Electronic Markets and Auctions". Danny Dolev is incumbent of the Berthold Badler Chair.

6. REFERENCES

[1] P. Berman, J. Garay, and K. J. Perry. Optimal Early Stopping in Distributed Consensus. In *Proc. 6th Workshop on Distributed Algorithms (WDAG)*, pages 221–237, 1992.

[2] B. Charron-Bost and A. Schiper. Uniform Consensus is Harder than Consensus. *Journal of Algorithms*, 51(1):15–37, 2004.

[3] B. S. Chlebus and D. R. Kowalski. Robust Gossiping with an Application to Consensus. *Journal of Computer and System Sciences*, 72(8):1262–1281, 2006.

[4] B. A. Coan. A Communication-Efficient Canonical Form for Fault-tolerant Distributed Protocols. In *Proc. 5th Symposium on Principles of Distributed Computing (PODC)*, pages 63–72, 1986.

[5] B. A. Coan and J. L. Welch. Modular Construction of a Byzantine Agreement Protocol with Optimal Message bit Complexity. *Information and Computation*, 97(1):61–85, 1992.

[6] D. Dolev and R. Reischuk. Bounds on Information Exchange for Byzantine Agreement. *Journal of the ACM*, 32:191–204, 1985.

[7] D. Dolev, R. Reischuk, and H. R. Strong. 'Eventual' is Earlier than 'Immediate'. In *Proc. 23rd Symposium on Foundations of Computer Science (FOCS)*, pages 196–203, 1982.

[8] D. Dolev, R. Reischuk, and H. R. Strong. Early Stopping in Byzantine Agreement. *Journal of the ACM*, 37(4):720–741, 1990.

[9] C. Dwork and Y. Moses. Knowledge and Common Knowledge in a Byzantine Environment: Crash Failures. *Information and Computation*, 88(2):156–186, 1990.

[10] M. Fischer, N. Lynch, and M. Patterson. Impossibility of Distributed Consensus with one Faulty Process. *Journal of the ACM*, 32(2):374–382, 1985.

[11] M. J. Fischer and N. A. Lynch. A Lower Bound for the Time to Assure Interactive Consistency. *Information Processing Letters*, 14:183–186, 1982.

[12] M. Fitzi and M. Hirt. Optimally Efficient Multi-valued Byzantine Agreement. In *Proc. 25th Symposium on Principles of Distributed Computing (PODC)*, pages 163–168, 2006.

[13] Z. Galil, A. Mayer, and M. Yung. Resolving Message Complexity of Byzantine Agreement and Beyond. In *Proc. 36th Symposium on Foundations of Computer Science (FOCS)*, pages 724–733, 1995.

[14] V. Hadzilacos and J. Y. Halpern. Message-optimal Protocols for Byzantine Agreement. *Mathematical Systems Theory*, 26:41–102, 1993.

[15] I. Keidar and S. Rajsbaum. A Simple Proof of the Uniform Consensus Synchronous Lower Bound. *Information Processing Letters*, 85(1):47–52, 2003.

[16] V. King and J. Saia. Scalable Byzantine Computation. *SIGACT News*, 41(3):89–104, 2010.

[17] V. King and J. Saia. Breaking the $\mathcal{O}(n^2)$ Bit Barrier: Scalable Byzantine Agreement with an Adaptive Adversary. *Journal of the ACM*, 58:18:1–18:24, 2011.

[18] G. Liang and N. H. Vaidya. Complexity of Multi-Value Byzantine Agreement. *CoRR*, abs/1006.2422, 2010.

[19] T. Mizrahi and Y. Moses. Continuous Consensus via Common Knowledge. *Distributed Computing*, 20:305–321, 2008.

[20] T. Mizrahi and Y. Moses. Continuous Consensus with Failures and Recoveries. In *Proc. 22nd Symposium on Distributed Computing (DISC)*, pages 408–422, 2008.

[21] P. R. Parvédy and M. Raynal. Optimal Early Stopping Uniform Consensus in Synchronous Systems with Process Omission Failures. In *Proc. 16th Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 302–310, 2004.

[22] A. Patra. Error-free Multi-valued Broadcast and Byzantine Agreement with Optimal Communication Complexity. In *Proc. 15th Conference on Principles of Distributed Systems (OPODIS)*, pages 34–49, 2011.

[23] K. Perry and S. Toueg. Distributed Agreement in the Presence of Processor and Communication Faults. *IEEE Transactions on Software Engineering*, SE-12(3):477–482, 1986.

[24] M. Raynal. *Fault-tolerant Agreement in Synchronous Message-passing Systems*. Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool, 2010.

[25] R. Reischuk. A New Solution for the Byzantine Generals Problem. *Information and Control*, 64(1–3):23–42, 1985.

[26] R. Zhang, Y. M. Teo, Q. Chen, and X. Wang. Lower Bounds for Achieving Synchronous Consensus with Orderly Crash Failures. In *Proc. 27th Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 61–68, 2007.