# The Complexity of Asynchronous Byzantine Consensus

Partha Dutta, Rachid Guerraoui and Marko Vukolić

Distributed Programming Laboratory, EPFL
CH-1015 Lausanne, Switzerland

**Abstract.** This paper establishes the first theorem relating resilience, round complexity and authentication in distributed computing. We give an exact measure of the time complexity of consensus algorithms that tolerate Byzantine failures and arbitrary long periods of asynchrony as in the Internet. The measure expresses the ability of processes to reach a consensus decision in a minimal number of rounds of information exchange, as a function of (a) the ability to use authentication and (b) the number of actual process failures, in those rounds, as well as of (c) the total number of failures tolerated and (d) the system configuration. The measure holds for a framework where the different roles of processes are distinguished such that we can directly derive a meaningful bound on the time complexity of implementing robust general services in practical distributed systems. To prove our theorem, we establish certain lower bounds and we give algorithms that match these bounds. The algorithms are all variants of the same generic asynchronous Byzantine consensus algorithm, which is interesting in its own right.

## 1 Introduction

### 1.1 Context

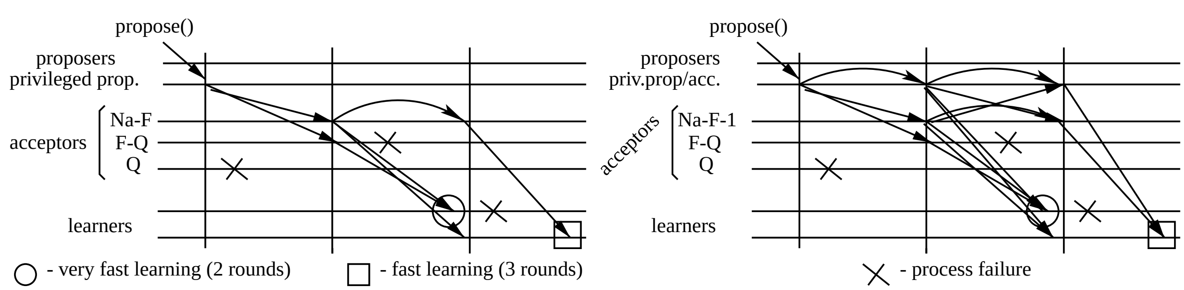
We establish a theorem on the complexity of the consensus problem in a general distributed system framework composed of three kinds of processes [21]: proposers, acceptors and learners (Fig. 1). Basically, the problem consists for the learners to decide on a common value among those proposed by the proposers, using acceptors as witnesses that help ensure the agreement. Every learner is supposed to eventually learn a value (liveness) that is the same proposed value for all learners (safety) [2]. Measuring the complexity of learning a decision in this framework automatically derives a measure of the complexity of state machine replication, a general technique to build robust distributed services [19, 31].

We study consensus algorithms that tolerate Byzantine failures. A Byzantine failure can either correspond to a crash or a malicious behavior (by default, a failure means a Byzantine failure). A process is malicious if it deviates from the algorithm assigned to it in a way that is different from simply stopping all activities (crashing). Besides process failures, the algorithms we consider also tolerate arbitrarily long periods of asynchrony, during which the relative speeds of processes and communication delays are unbounded. Such algorithms are sometimes called asynchronous [6, 21]. We assume however that the duration of the asynchronous periods and their number of occurrences are both finite, otherwise consensus is known to be impossible [14]. Processes that do not fail are called correct processes, and they can eventually communicate among each other in a timely manner. The model assumed here, called the eventually synchronous model [12], matches practical systems like the Internet which are often synchronous and sometimes asynchronous. Whereas it is important to tolerate periods of asynchrony and the largest number of faults possible, it is also important to optimize algorithms for favorable, and most frequent, situations where the system is synchronous and very few processes fail.

Clearly, it is never possible to learn a decision in one round of information exchange (we say communication round) and yet ensure agreement despite possible Byzantine failures. There are however algorithms [6] where, in certain favorable situations, a decision is learned after three communication rounds by all correct learners: we talk about *fast* learning. In fact, as conjectured in [21], and as we show in this paper, there are even slightly more favorable situations, which are still very plausible in practice, under which learning can be achieved in two communication rounds: we talk in this case about *very fast* learning and a proposer from which a value can be learned very fast is called a privileged proposer. The theorem we establish in this paper determines the exact conditions under which very fast (resp. fast) learning can be achieved. Underlying the theorem lies the notion of *favorable* (resp. *very favorable*) runs that precisely captures the favorable situations we mentioned above. Namely, a run $r$ of a consensus algorithm $A$ is said to be very favorable (resp. favorable) if: (1) $r$ is synchronous, (2) a single (correct) privileged proposer $p_l$ proposes a value in $r$ and (3) at most $Q \leq F$ (resp. more than $Q$ but at most $F$) acceptors are faulty (here, $F$ is one of the resilience thresholds define below). Basically, very fast (resp. fast) learning is achieved in very favorable (resp. favorable) runs of algorithm $A$.

Our theorem is general in that it is parameterized by (1) different resilience thresholds, (2) different system configurations, as well as (3) the ability of processes to use authentication primitives (public-key cryptography) [30] to achieve fast (resp. very fast) learning.

1. We distinguish two resilience thresholds: $M$ and $F$; $M$ denotes the maximum number of acceptor malicious failures despite which consensus safety is ensured (the number of acceptor crash-only failures does not influence safety); $F$ denotes the maximum number of acceptor failures despite which consensus liveness is ensured. Particularly interesting is the case where $M > F$: consensus safety should be preserved despite $M$ acceptor malicious failures, but liveness is guaranteed only if the number of acceptor failures is at most $F$.
2. We distinguish two system configurations: $C_1$ and $C_2$; $C_1$ is the configuration where at least one privileged proposer might not be an acceptor, or there are at least two privileged proposers (Fig. 1(a)); $C_2$ is the configuration where there is only one privileged proposer, which is also one of the acceptors (Fig. 1(b)).
3. Finally, we also distinguish the case where the processes are allowed to use authentication to achieve very fast (resp. fast) learning from the case where they are not. Note that, in both cases, we do not prevent processes from using authentication in runs that do not enable very fast or fast learning, typically non-favorable runs with proposer failures and asynchronous periods. Roughly, authentication allows the recipient of the message to validly claim to a third party that it received the message from the actual original sender of the message [30]. This ability is a major source of overhead [24,27] and hence, we would typically like to avoid using authentication for (very) fast learning.




(a) Configuration $C_1$ (b) Configuration $C_2$

**Fig. 1.** Very fast and fast learning in each configuration

### 1.2 Theorem

The theorem states that there is a consensus algorithm that:

1. Achieves very fast learning in configuration $C_1$ despite the failure of $Q$ acceptors if and only if the total number of acceptors in the system ($N_a$) is such that $N_a > 2M + F + 2Q$. In addition, the same algorithm achieves fast learning despite the failure of $F$ acceptors,
   - *with* authentication, given the same total number of acceptors, $N_a$.
   - *without* authentication, if and only if $N_a$ is also greater than $2F + M + min(M, Q)$.
2. Achieves very fast learning in configuration $C_2$ with (resp. without) authentication despite the failure of $Q$ acceptors if and only if $N_a > 2(M-1) + F + 2Q$ (resp. $N_a > max(2(M-1) + F + 2Q, 2M + F + Q)$). In addition, the same algorithm achieves fast learning despite the failure of $F$ acceptors,
   - *with* authentication, given the same total number of acceptors, $N_a$.
   - *without* authentication, if and only if $N_a$ is also greater than $2F+(M-1)+min(M-1, Q)$.

To help understand some of the parameters of the theorem, let us illustrate them through some specific interesting cases and focus on configuration $C_1$ where authentication does not impact the ability to achieve very fast learning.

1. $F = M$. Here we consider consensus algorithms that are correct when at most $F = M$ acceptors fail. At one extreme, very fast (resp. fast) learning is possible without authentication when ($Q = 0$) no acceptor fails (resp. $F$ acceptors fail) if and only if $N_a > 3F$; i.e., less than one-third of the total number of acceptors can fail. At the other extreme, very fast learning is possible without authentication when the maximum possible number of acceptors fail ($Q = F$) if and only if $N_a > 5F$; i.e., less than one-fifth of the total number of acceptors can fail. In the second case, very fast learning is possible whenever fast learning is also possible.
2. $F = 3$ and $M = 1$. Here we consider consensus algorithms that are correct when at most $F = 3$ acceptors fail, out of which at most $M = 1$ acceptors are malicious. For such algorithms, when for example $Q = 1$ acceptor fails, very fast learning is possible only if $N_a \geq 2M+F+2Q+1 = 8$. However, with $N_a = 8$ acceptors, fast learning is only possible with authentication. To achieve fast learning without authentication, $N_a \geq 2F+M+min(M, Q)+1 = 9$ acceptors are required.
3. $F = 2$, $M = 3$. Here we consider consensus algorithms that are correct when at most $F = 2$ acceptors fail and possibly only safe when at most $M = 3$ acceptors are malicious (i.e., these algorithms are correct when there are at most 2 failures, and they are safe but may not be live when there are 3 malicious failures). For such algorithms, when for example $Q = 1$ acceptor fails, very fast learning is possible only if $N_a \geq 2M + F + 2Q = 11$. At the same time, the same number of acceptors allows fast learning without authentication (because $N_a$ is greater than $2F + M + min(M, Q) = 8$).

In short, the theorem expresses, in a general and precise way, a fundamental trade-off between the resilience and complexity of asynchronous Byzantine consensus. Two sides of complexity are considered: the communication complexity (sometimes called latency), which depicts the number of rounds of information exchange before a decision is learned, as well as the authentication complexity, considered a major overhead factor in Byzantine computing [24, 27].

### 1.3 Proof Overview

The necessary parts of our theorem consist of a set of lower bounds. We prove these bounds using indistinguishability arguments that simultaneously exploit the asynchrony of the network and the Byzantine failures of the processes, in order to contradict the ability to achieve very fast (resp. fast) learning. For example, to show that very fast learning is impossible given a certain number of failures, we first construct two very favorable runs where some learner $l$ learns two distinct values very fast (i.e., in two communication rounds). Second, we exhibit two asynchronous runs with Byzantine failures that are respectively indistinguishable at $l$ from the two very favorable synchronous runs, and hence $l$ learns distinct values in the two asynchronous runs. Third, we make use of asynchrony and Byzantine failures in a way that the two asynchronous runs are indistinguishable to any learner distinct from $l$. This helps us build an eventually synchronous run in which *Agreement* is violated. Assuming that the processes can use authentication in effect restrict the range of the possible Byzantine behavior that we can exploit in the proof.

The sufficiency parts of our theorem are shown by exhibiting algorithms that match the corresponding lower bounds. Interestingly, these algorithms can all be viewed as variants from the same generic algorithm, which we call "Distributed consensus à Grande Vitesse" (DGV). DGV is parameterized by $F$, $M$ (and $Q$) as well as by the underlying configuration considered ($C_1$ vs. $C_2$). The algorithm constitutes an appealing building block to implement robust yet efficient distributed services on the Internet. DGV allows learning in two communication rounds (i.e., very fast learning) in very favorable runs, and, at the same time, gracefully degrades to allow learning in three communication rounds (i.e., fast learning) when the conditions are slightly less desirable, i.e., in favorable runs. Roughly, in DGV, if some decision value $v$ was learned at time $t$, it is guaranteed that no proposer can impose a value other than $v$ after time $t$. To achieve this, the value that acceptors can indeed accept is carefully selected on the basis of the acceptors estimates of the decision value up to that point of the execution, not to miss a value that may have been learned. In certain cases, potential disputes on which value should be accepted may arise, but these are solved by detecting the existence of the malicious acceptors that cause these disputes. DGV is composed of two parts: (1) a *Locking* module and (2) an *Election* module. In short, the *Locking* module ensures consensus safety whereas the *Election* module ensures consensus liveness under eventual synchrony assumption. The key element of DGV is its *choose()* function, within the *Locking* module, that determines which value should be accepted by an acceptor at a given point of execution. Variants of DGV are mainly obtained by varying implementations of this function.

### 1.4 Roadmap

In Section 2 we discuss related work. In Section 3 we recall the consensus problem and we define the model we consider in this paper. In Section 4 we show the necessary part of our theorem, by establishing and proving certain lower bounds, for both configurations ($C_1$, followed by $C_2$). In Section 5 we give the algorithms that match these lower bounds, as variants of our generic DGV algorithm.

## 2 Related Work

In the following, we first recall the historical context of asynchronous Byzantine consensus and its solvability bounds. Then we mention previous complexity. Finally, we compare our algorithm with related ones.

Byzantine consensus was introduced by Pease, Shostak and Lamport [26] in a synchronous model of distributed computation, where they established that two-third of correct processes is necessary and sufficient to solve the problem if processes do not use authentication. The same bound was extended in [4] to the asynchronous case, even if processes can use authentication. In the general framework of [21], which we consider in this paper, this translates into $N_a > 3F$ and $M = F$. In that framework, and for the more general case where $M \neq F$, it is not very difficult to extend the proof of [4] and show that $N_a > 2F + M$ is a necessary and sufficient condition to solve asynchronous Byzantine consensus [21].

In [8,13], it was shown that any synchronous Byzantine consensus algorithm needs $t+1$ rounds of information exchange (communication rounds) to reach consensus, where $t$ is the number of failures tolerated. The bound is given considering the case where all processes must simultaneously reach a decision. If the decision does not need to be reached simultaneously, then early decision is possible and $min(f+2, t+1)$ rounds are needed for deciding [9,17] in runs where $f \leq t$ processes are faulty. The model with asynchronous periods, called the eventually synchronous model, was first introduced by Dwork, Lynch and Stockmeyer in [12] (with language abuse, and as we mentioned in the introduction, algorithms in this model have been called asynchronous [6,21]). For asynchronous algorithms with crash failures, it was established that $f+2$ rounds are needed to achieve consensus in synchronous runs with $f$ failures [11]. All these complexity bounds were established in a restricted framework where all processes play the same role.

In [21], Lamport motivated the study of consensus complexity bounds in a general framework with distinct proposers, acceptors and learners, for the ability of this framework to better match the practical use of consensus within state machine replication protocols [19,31]. He conjectured a fundamental tight bound on the maximum resilience to achieve very fast learning in asynchronous Byzantine consensus. Recently, and concurrently with this paper, Lamport proved his conjecture for non-Byzantine failures in [22]. Our theorem generalizes that conjecture, which we thus prove for the Byzantine case as well. Our generalization goes in two directions. First, we consider the possibility of fast learning, besides very fast learning, and this can be viewed as a nice graceful degradation flavor for runs that are not favorable enough to allow very fast learning, but are still favorable enough to allow fast learning. Second, we also consider the impact of using authentication [30]. By doing so, we highlight the fact that Lamport's conjecture [21] holds only if very fast learning with authentication is precluded. In many systems, the use of authentication is far more expensive than several rounds of communication [24,27]. It is thus of primary importance to state the precise impact of authentication on the number of communication rounds needed to achieve consensus.

Certain Byzantine consensus algorithms are synchronous (e.g., [28]), or assume that a subset of the system is synchronous [7]. The first asynchronous Byzantine consensus algorithm was given by Castro and Liskov in [6]. The algorithm, called Practical Byzantine Fault Tolerance (PBFT), considered the special case where $M = F$ and $N_a = 3F + 1$. In PBFT, fast learning is achieved in synchronous runs with up to $F = M$ malicious acceptors and assuming a correct leader (i.e., what we call favorable runs in this paper). Very fast learning, which is possible in their case for $Q = 0$ was not considered. Our DGV algorithm enables very fast learning if the leader is correct, the run is synchronous and up to $Q$ acceptors fail (called very favorable runs in this paper). However, if $Q'$ acceptors fail, where $Q < Q' \leq F$, DGV degrades gracefully and features the same complexity as PBFT (i.e., fast learning).

A deconstruction of the Paxos algorithm, similar to the decomposition of DGV we give in this paper, was given in [3] for the non-Byzantine case. Our decomposition makes it possible to have a reusable *Locking* module (capturing consensus safety properties), that can be combined with different *Election* algorithms (providing consensus liveness), in the same vein as [10]. For instance, our *Election* module can easily be shifted to the level of proposers or implemented by a deterministic scheduler. In comparison, in PBFT, the techniques used for ensuring safety were incorporated into the liveness providing part (the leader change algorithm). By introducing a pair of additional messages in certain (non-favorable) runs of DGV which, by the way, do not critically degrade the performance of the algorithm in these runs, we make the safety providing part of the algorithm (i.e., *Locking* module) independent from the liveness providing part of the algorithm (i.e., *Election* module). In practical implementations, one might of course consider removing these messages.

A few asynchronous Byzantine consensus algorithms enabling very fast learning have been recently proposed. Namely, these are Kursawe's optimistic Byzantine Agreement [18], Martin and

Alvisi's FaB Paxos (Fast Byzantine Paxos) [25], the oracle-based protocol by Friedman, Mostefaoui and Raynal [15], and Lamport's algorithm [23]. Kursawe's algorithm considers the specific case of $M = F$ and $N_a = 3F + 1$, and enables very fast learning when $Q = 0$. It does not allow fast learning if some acceptor actually fails (which is feasible in this case). Developed concurrently with this paper, Martin and Alvisi's FaB Paxos algorithm is simple and elegant: it considers configuration $C_1$ and the case where $M = F = Q$ (therefore assuming $N_a = 5F+1$ acceptors), while enabling very fast learning despite $F$ failures. This algorithm does not match the lower bound in configuration $C_2$, nor does it adapt to the general case where $M \neq F \neq Q$, in configuration $C_1$. The oracle-based randomized protocol by Friedman et al. considers the case where $M = F$ and $Q = 0$ in configuration $C_1$, and achieves very fast learning with $N_a = 5F + 1$ acceptors (more than required by our lower bound). Lamport's algorithm achieves very fast learning in configuration $C_1$.

To summarize, while certain algorithms, for some special values of $Q$, for the special case where $M = F$ and in configuration $C_1$ were suggested in the literature, our DGV algorithm is the first generic one with respect to $M$, $F$ and $Q$, that delivers optimal performance. Furthermore, even for the special case where $M = F$, we are not aware of any solution that handles the case where $Q \neq 0$ in configuration $C_2$. In addition, no algorithm combines very fast learning and fast learning: achieving both properties is not trivial, especially when handling the general values of $M$, $F$ and $Q$ and precluding authentication.

## 3 Preliminaries

In this paper we address the consensus problem, as defined in [21], in a distributed system composed of three sets of processes: (1) *proposers* $= \{p_1, p_2, ..., p_{N_p}\}$, (2) *acceptors* $= \{a_1, a_2, ..., a_{N_a}\}$, and (3) *learners* $= \{l_1, l_2, ..., l_{N_l}\}$ [20,21]. In this problem, every proposer starts with a proposal value. Proposers may never propose a value, or may propose several times. Learners need to learn the same proposal value. Acceptors act as witnesses to help learners agree. On proposing a value, a proposer communicates with acceptors, and learners learn a value on receiving appropriate messages from the acceptors. More precisely, in every run of a consensus algorithm, only proposers propose values and learners learn values, such that the following properties hold:

- (Validity:) If a learner learns a value $v$, then some proposer proposes $v$[1];
- (Agreement:) No two learners learn different values;
- (Termination:) If a correct proposer proposes a value, then eventually, every correct learner learns a value.

Processes may fail by arbitrarily deviating from the algorithm assigned to them. When a process fails by crashing, i.e., simply stop its execution, we say that it has crashed, and if it deviates from the algorithm in a way different from crashing, we say that it is *malicious*. We consider consensus algorithms that provide safety properties, i.e., *Validity* and *Agreement*, despite at most $M$ malicious acceptors. Safety properties of consensus are preserved regardless of the number of acceptor that crashed, as long as the number of malicious acceptors is at most $M$. However, we consider consenus algorithms that provide liveness, i.e., *Termination*, only if the total number of acceptor failures is at most $F$. In the case where $M \geq F$, *Validity* and *Agreement* have to be preserved in all runs in which at most $M$ acceptors fail. However, *Termination* might be violated and it is guaranteed only if the number of actual acceptor failures is at most $F$. Finally, any number of learners might fail by crashing. As in [21], we assume that learners are not malicious.

[1] Here we adopt language abuse for presentation simplicity. In fact, it is impossible to ensure that a malicious proposer, on proposing a value, will not pretend that it has proposed a different value. A more precise definition of *Validity* would be: if a learner $l$ learns a value $v$ in run $r$, then there is a run $r'$ (possibly different from $r$) such that some proposer proposes $v$ in $r'$, and $l$ cannot distinguish $r$ from $r'$.

Every pair of processes is connected by a bi-directional channel that may duplicate, delay or lose messages, or may deliver them out of order. However, channels do not alter messages. We assume that every message $m$ that is sent is unique and has a $m.sender$ field that is supposed to contain a unique identifier of the sending process. We assume a computationally bounded adversary as well as standard cryptographic techniques in the design of Byzantine consensus algorithms [6]. We consider public-key cryptography [30] (PKC), message authentication codes [32] and message digests [29], where $D(m)$ denotes a digest of the message $m$ and $\langle m \rangle_{\sigma_p}$ denotes $m$, accompanied by $D(m)$ digitally signed by process $p$. It is commonly admitted that PKC is usually considered pretty expensive [24, 27]. As a consequence, and as pointed out in the introduction, we distinguish the case where processes can always use PKC (we say use authentication), including in favorable (or very favorable) runs to achieve fast (or very fast) learning, from the case where the processes can only use authentication when fast (or very fast) learning is not possible.

To circumvent the impossibility of fault-tolerant consensus in an asynchronous system [14], we make the following eventual synchrony assumptions [12]: in any run, there is a bound $\Delta_c$ and a time $GST$ (*Global Stabilization Time*), such that any message sent by a correct process to a correct process at time $t' \geq GST$ is received by time $t' + \Delta_c$. $\Delta_c$ and $GST$ do not need to be known by the processes. We also assume an upper bound $\Delta_{auth}$ on the local computation related to PKC. We assume all other local computations to require negligible time.

Assume that every process starts consensus with some estimate of $\Delta_c$ and $\Delta_{auth}$ (the bounds on message transmission delay and local computations). We say that a run of the consensus algorithm is *synchronous* if: (1) all correct processes have the same estimates of $\Delta_c$ and $\Delta_{auth}$, say $\delta_c$ and $\delta_{auth}$, respectively, (2) $\delta_c \geq \Delta_c$ and $\delta_{auth} \geq \Delta_{auth}$, and (3) $GST = 0$. Roughly speaking, in a synchronous run, no correct process times out waiting for messages from another correct process.

## 4 Lower Bounds

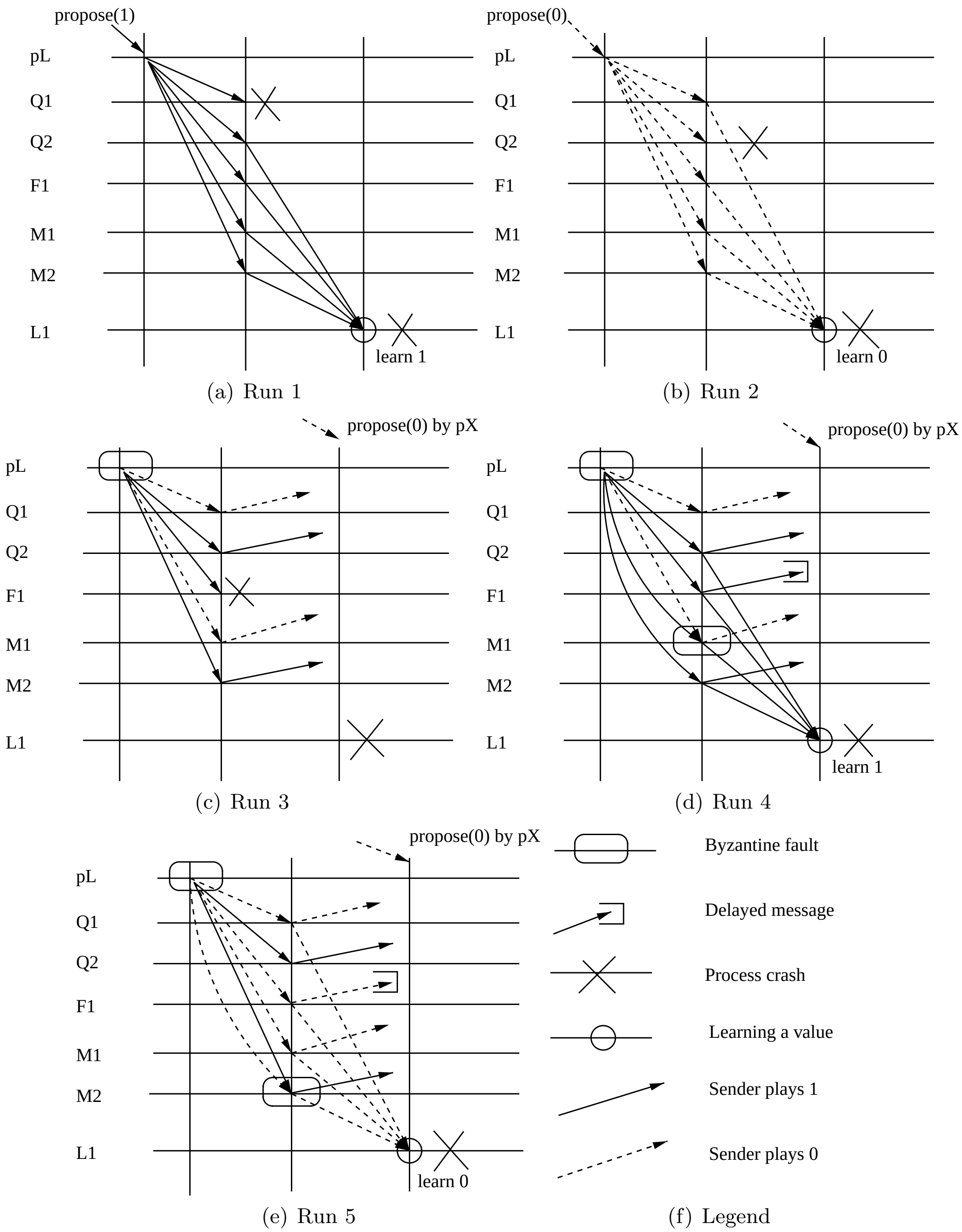
To precisely state the lower bounds underlying our theorem, we assume full information protocols in a *round-by-round* eventually synchronous model [16]. In each round, the processes send messages to all processes, receive messages in that round, update their states and move to the next round, such that the following properties hold. Denote by $alive(k)$ the set of processes that complete round $k$. There is a round $k$ such that for every round $k' \geq k$, every message sent by a correct process in $alive(k')$ to another correct process in $alive(k')$ is delivered in round $k'$. A synchronous run is then simply a run in which $k = 1$. In our lower bounds, we assume that there are always at least two proposers and at least two learners.

First we prove the lower bounds for configuration $C_1$. Recall that the system is in configuration $C_1$ if: (a) there is a single privileged proposer that is not an acceptor (configuration $C_{1a}$), or (b) there is more than one privileged proposer (regardless of whether they are also acceptors or not - configuration $C_{1b}$). Then we will consider configuration $C_2$, where there is a single privileged proposer, that is also an acceptor.

### 4.1 Configuration $C_1$

Consider the case with a single privileged proposer that is not an acceptor (configuration $C_{1a}$).

**Proposition L.1.** Let $A$ be any algorithm and $p_l$ the only privileged proposer, which is furthermore not an acceptor. If in every very favorable run of $A$ every correct learner learns a value by round 2 despite the failure of $Q$ acceptors, then $N_a > 2Q + F + 2M$.




(a) Run 1 (b) Run 2 (c) Run 3 (d) Run 4 (e) Run 5 (f) Legend

**Fig. 2.** Illustration of proof L.1: lower bound on very fast learning, configuration $C_{1a}$ - case with a single privileged proposer that is not an acceptor

**Proof L.1.** Suppose by contradiction that $N_a \leq 2Q + F + 2M$. We divide the set of acceptors into five sets, $Q_1$, $Q_2$, $F_1$, $M_1$ and $M_2$, where the first two sets are of size at most $Q$, the third set is of size at most $F$, and the last two sets are of size at most $M$, respectively. Without loss of generality we assume that each of these five set consists of only one process. If a set has more than one process, we simply modify the runs so that all processes inside a set receive the same set of messages, and if they fail, they fail at the same time, in the same way; the proof also holds if any of the bounds $Q$, $F$ or $M$ is 0. Assume there are two learners $l_1$ and $l_2$, and there are two proposers: the privileged proposer $p_l$, and proposer $p_x$.

Suppose $p_l$ is correct and proposes a value at the beginning of round 1. If a proposal value is learned in round 2, then the only possible communication pattern is the following (remember that on proposing a value, the proposer communicate with acceptors, and learners learn a message on receiving appropriate messages from the acceptors): (Round 1) proposer $p_l$ sends messages to all acceptors; (Round 2) every acceptor forwards the message received in the first round to every process. Learners on receiving a sufficient number of messages from acceptors learn a value.

We only consider the cases where $p_l$ proposes 0 or 1 at the beginning or round 1 (as this is sufficient to prove the lower bound). Let $m1$ and $m0$ be the authenticated messages, sent by $p_l$ in round 1, when $p_l$ is correct and proposes 1 or 0, respectively. We say that an acceptor $a_i$ *plays 1* (resp. 0) to some acceptor or learner $q_j$ in round $k$ of some run $r$, if $q_j$ cannot distinguish at round $k$, the run $r$ from some run in which (1) $a_i$ has received $m1$ (resp. $m0$) from $p_l$ in the first round, and (2) $a_i$ is correct. It is important to note that, due to the cryptographic assumptions we make, $a_i$ can play 1 (or 0) only if $a_i$ has received $m1$ (resp. $m0$) from $p_l$. (If $p_l$ is faulty then $p_l$ may send $m1$ to $a_i$ even if $p_l$ proposed 0, and thus, $a_i$ may play 1.)

A *very favorable partial run* is a prefix of a very favorable run. From our assumption, in every very favorable run, the correct learners learn the proposal value (of $p_l$) by round 2. Consider the following two very favorable partial runs, R1 and R2 (The message patterns of the first rounds of these runs are illustrated in Figure 2).

**R1:** All processes except $l_1$ and $Q_1$ are correct. Proposer $p_l$ proposes 1 in round 1, $Q_1$ crashes before sending any message in round 2, and learner $l_1$ receives round 2 messages from all acceptors except $Q_1$. From our assumption on $A$, $l_1$ learns 1 at the end of round 2, and then, $l_1$ crashes before sending any message in round 3.

**R2:** All processes except $l_1$ and $Q_2$ are correct. Proposer $p_l$ proposes 0 in round 1, $Q_2$ crashes before sending any message in round 2, and learner $l_1$ receives round 2 messages from all acceptors except $Q_2$. From our assumption on $A$, $l_1$ learns 0 at the end of round 2, and then, $l_1$ crashes before sending any message in round 3.

We now construct three runs that are not very favorable.

**R3:** All processes are correct *except* (1) the proposer $p_l$, which is malicious, (2) acceptor $F_1$, which crashes before sending any message in round 2, and (3) $l_1$ which crashes before sending any message in round 3. In round 1, proposer $p_l$ sends $m0$ to the acceptors $Q_1$ and $M_1$, and $m1$ to the rest of the acceptors. Acceptor $F_1$ crashes such that no process receives round 2 message from $F_1$, and $l_1$ crashes such that no process receives any round 3 message from $l_1$. From round 2, acceptors $Q_1$ and $M_1$ play 0 to all processes, and $Q_2$ and $M_2$ play 1 to all processes. In round 3, correct proposer $p_x$ proposes 0. Since a correct proposer proposes, eventually correct learner $l_2$ learns some value $v \in \{0, 1\}$, say at round $K$.

**R4:** All processes are correct except (1) the proposer $p_l$ and the acceptor $M_1$, which are malicious, and (2) learner $l_1$ which crashes before sending any message in round 3. In round 1, proposer $p_l$ sends $m0$ to acceptor $Q_1$, both $m1$ and $m0$ to $M_1$, and $m1$ to the rest of the acceptors. In round 2 and higher rounds, acceptor $Q_1$ plays 0 to all processes, acceptors $Q_2$, $F_1$ and $M_2$ play 1 to all processes. However, malicious acceptor $M_1$ plays 1 to learner $l_1$ and plays 0 to all other processes. ($M_1$ can do so because it has received both $m1$ and $m0$.) Learner $l_1$ receives round 2 message from all acceptors except $Q_1$. Clearly, at the end of round 2, $l_1$ cannot distinguish R4 from R1 (because it receives the same set of messages from the acceptors in round 2 of both runs), and hence, learns 1. Then learner $l_1$ crashes before sending any message in round 3. In round 3, correct proposer $p_x$ proposes 0. Up to round $K$, all non-crashed (i.e., processes that are correct or malicious) processes receive messages from all other non-crashed processes distinct from $F_1$ (i.e., all messages sent by $F_1$ are lost up to round $K$). At the end of round $K$, no correct process distinct from $F_1$ can distinguish R4 from R3 (because every non-crashed acceptor different from $F_1$ plays identical values in both runs), and hence, learner $l_2$ learns $v \in \{0, 1\}$ at round $K$. All non-crashed processes receive messages from all other non-crashed processes (including $F_1$) in rounds higher than $K$.

**R5:** All processes are correct except (1) the proposer $p_l$ and the acceptor $M_2$, which are malicious, and (2) learner $l_1$ which crashes before sending any message in round 3. In round 1, proposer $p_l$ sends $m1$ to acceptor $Q_2$, both $m1$ and $m0$ to $M_2$, and $m0$ to the rest of the acceptors. In round 2 and higher rounds, acceptor $Q_2$ plays 1 to all processes, acceptors $Q_1$, $F_1$ and $M_1$ play 0 to all processes. However, malicious acceptor $M_2$ plays 0 to learner $l_1$ and plays 1 to all other processes. ($M_2$ can do so because it has received both $m1$ and $m0$.) Learner $l_1$ receives round 2 message from all acceptors except $Q_2$. Clearly, at the end of round 2, $l_1$ cannot distinguish R5 from R2 (because it receives the same set of messages from the acceptors in round 2 of both runs), and hence, learns 0. Then learner $l_1$ crashes before sending any message in round 3. In round 3, correct proposer $p_x$ proposes 0. Up to round $K$, all non-crashed processes receive messages from all other non-crashed processes distinct from $F_1$ (i.e., all messages sent by $F_1$ are lost up to round $K$.) At the end of round $K$, no correct process distinct from $F_1$ can distinguish R5 from R3 (because every non-crashed acceptor different from $F_1$ plays identical values in both runs), and hence, learner $l_2$ learns $v \in \{0, 1\}$ at round $K$. All non-crashed processes receive messages from all other non-crashed processes (including $F_1$) in rounds higher than $K$.

Clearly, either R4 or R5 violates consensus *Agreement*: $l_2$ decides $v$ in both runs, but $l_1$ decides 1 in R4 and 0 in R5. However, in both runs, at most $M$ acceptors are faulty: a contradiction with the requirement that $A$ does not violate *Validity* and *Agreement* if $M$ processes are faulty. □

**Remarks (L.1).** In round 3 of R3, the proposal by $p_x$ is required to ensure that $l_2$ decides in R3. If $p_x$ does not propose then the only (possible) proposal in the run is by the malicious proposer $p_l$, and hence, the *Termination* property does not require any learner to decide. Secondly, for ease of presentation we state in the proof that $p_l$ sends two messages ($m1$ and $m0$) to $M_1$ in round 1 of R4. In fact, $p_l$ may sends a single message such that $M_1$ can recover both $m1$ and $m0$ from it. Since both $p_l$ and $M_1$ are malicious in R4, they collude to achieve this. (A similar argument holds for messages from $p_l$ to $M_2$ in round 1 of R5.)

Now we sketch the proof of the same bound in the case of configuration $C_{1b}$, i.e. where there are two (or more) privileged proposers (that do not necessarily have to be malicious). The proof is very similar to proof L.1. We will refer to this proof as to proof L.2.

**Proposition L.2.** Let $A$ be any algorithm, and $p_v$ and $p_w$ two privileged proposers. If, in every very favorable run of $A$, in which a single privileged proposer proposes, every correct learner learns a value by round 2, then $N_a > 2Q + F + 2M$. (To strengthen this proposition, we assume that privileged proposers may fail only by crashing.)

**Proof L.2 (sketch).** The proof of this part of the theorem is a trivial modification of proof L.1, obtained as follows. Each run in proof L.1 (R1 to R5) is modified to get five new runs (R1' to R5'). To get run Ri' from Ri ($1 \leq i \leq 5$), in round 1, we remove the propose by proposer $p_l$ and add propose(1) by $p_v$ and propose(0) by $p_w$, and we define $m1'$ to be the message sent by $p_v$ (on proposing 1) and $m0'$ to be the message sent by $p_w$ (on proposing 0). In round 1 of Ri', an acceptors receives $m1'$ if it receives $m1$ in Ri (and similarly for $m0'$). (Thus $M_1$ receives messages from both $p_v$ and $p_w$ in round 1 of R4.) Acceptors play 1 on receiving $m1'$ and play 0 on receiving $m0'$. The rest of the proof remains the same.[2] □

Now we prove the rest of part 1 of the theorem. This establishes the bound on the number of acceptors required for achieving fast learning without authentication combined with very fast learning (with or without authentication) in configuration $C_{1a}$ (the proof can be migrated to configuration $C_{1b}$ in the similar way proof L.2 is derived from proof L.1). We refer to this proof as proof L.3. Clearly, any algorithm that achieves fast learning without authentication combined with very fast learning has a *restricted authentication pattern*:

1. The messages sent from the privileged proposer to acceptors in round 1, and the messages sent from the acceptors to the learners in round 2, may or may not be authenticated, and
2. The messages exchanged among acceptors in round 2, and the messages sent from acceptors to the learners in round 3, are not authenticated.

In this proof we use indistinguishability arguments, that exploit the fact that a malicious process can claim that it received a different value from a correct process than the one the correct process actually sent. This is possible, as we assume that the messages that are used for fast, but not for very fast learning, are not authenticated. In addition, we exploit the asynchrony of the network and the fact that malicious processes can cooperate.

**Proposition L.3.** Let $A$ be any algorithm and $p_l$ the only privileged proposer, which is a furthermore an acceptor, such that, in every very favorable (resp. favorable) run of $A$ every correct learner learns a value by round 2 (resp. 3). In addition, suppose $A$ satisfies the restricted authentication pattern. Then, $N_a > 2F + M + min(M, Q)$.

**Proof L.3.** First, we consider the case where $M \geq Q$. Suppose, by contradiction, that $N_a \leq 2F + M + Q$. We divide the set of acceptors into five sets, $F_1$, $F_2$, $MQ_1$, $Q_1$ and $Q_2$, where the first two sets are of size at most $F$, the third set is of size at most $M - Q$, and the last two sets are of size at most $Q$. Without loss of generality, we assume that each of these five sets consists of only one process. (If a set has more than one process, we just modify the runs so that all processes inside a set receive the same set of messages, and if they fail, they fail at the same time, in the same way. The proof also holds if any of $Q$, $F$ or $M - Q$ is 0.) We assume at least two learners, $l_1$ and $l_2$, and two proposers: the potentially malicious privileged proposer $p_l$, and the proposer $p_x$.

We only consider the cases where $p_l$ proposes 0 or 1 (as this is sufficient to prove the lower bound). Let $m1$ and $m0$ be the authenticated messages sent by $p_l$ in round 1, when $p_l$ is correct and proposes 1 or 0, respectively. We say that an acceptor $a_i$ *plays* 1 (resp. (0) to some process $a_j$ in round 2 of some run $r$ if $a_j$ cannot distinguish, at round 2, run $r$ from some run in which

[2] We can however slightly simplify the proof L.2: proposer $p_x$ may be removed. Recall that, in $R3$, proposal by correct proposer $p_x$ was introduced to ensure that $l_2$ eventually learns a value. However, in $R3'$ both $p_v$ and $p_w$ are correct and proposes a value, and hence, even without the proposal of $p_x$, $l_2$ is required to learn a value.

(1) $a_i$ has received $m1$ (resp. $m0$) from $p_l$ in the first round, and (2) $a_i$ is correct. Furthermore, we say that an acceptor $a_i$ *plays* a tuple $(f_1, f_2, mq_1, q_1, q_2)$ to some process $a_j$ in round 3 of some run $r$ if $a_j$ cannot distinguish, at round 3, the run $r$ from some run in which (1) $a_i$ has received the value $f_1$ from $F_1$, $f_2$ from $F_2$, $mq_1$ from $MQ_1$, $q_1$ from $Q_1$ and $q_2$ from $Q_2$ in round 2, and (2) $a_i$ is correct. Here, $a_i$ receives a value $x_j$ (0 or 1) from $X_j$ means that $X_j$ played $x_j$ to $a_i$ in round 2 and $a_i$ received this message during the round 2. If correct acceptor $a_i$ does not receive any message from $X_j$ in round 2, $a_i$ plays $'-'$ in place of $x_j$ in round 3.

A *very favorable partial* run is a prefix of a very favorable run. Similarly, a *favorable partial* run is a prefix of a favorable run. From our assumption, in every very favorable run (i.e., in the synchronous run in which up to $Q$ acceptors fail and in which a single correct privileged proposer $p_l$ proposes), the correct learners learn the proposal value (of $p_l$) by round 2. Furthermore, in every favorable run (i.e., in the synchronous run in which up to $Q'$, $Q < Q' \leq F$, acceptors fail and in which the single correct privileged proposer $p_l$ proposes), the correct learners learn the proposal value (of $p_l$) by round 3. Keeping in mind that $A$ satisfies restricted authentication pattern, consider the following runs: a favorable partial run R1 and a very favorable partial run R2. (The patterns of the messages exchanged in the initial rounds of runs are depicted in Figure 3.)

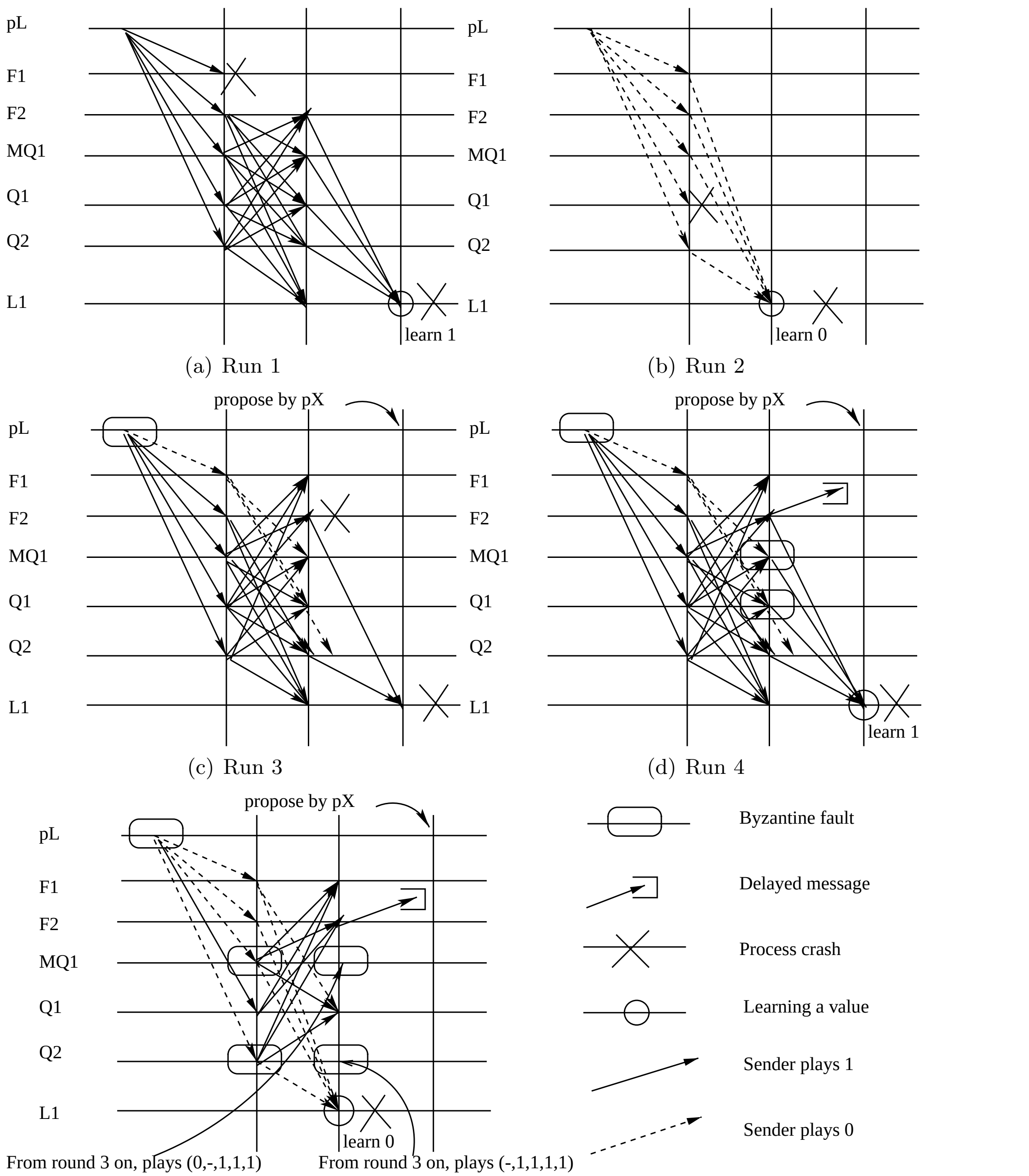
**R1:** All processes, except $F_1$ and $l_1$, are correct. Proposer $p_l$ proposes 1 in round 1, $F_1$ crashes before sending any message in round 2, and learner $l_1$ receives round 3 messages from all acceptors, except from $F_1$. From our assumption on $A$, $l_1$ learns 1 at the end of round 3, and then $l_1$ crashes before sending any message in round 4.

**R2:** All processes, except $l_1$ and $Q_1$, are correct. Proposer $p_l$ proposes 0 in round 1, $Q_1$ crashes before sending any message in round 2, and learner $l_1$ receives round 2 messages from all acceptors, except from $Q_1$. From our assumption on $A$, $l_1$ learns 0 at the end of round 2, and then, $l_1$ crashes before sending any message in round 3.

We now construct three non-favorable runs.

**R3:** All processes are correct *except* (1) the proposer $p_l$, which is malicious, (2) acceptor $F_2$, which crashes after sending the round 3 message to $l_1$, and (3) $l_1$ which crashes before sending any message in round 4. In round 1, the proposer sends $m1$ to the acceptors $F_2$, $MQ_1$, $Q_1$ and $Q_2$, and $m0$ to $F_1$. In round 2, messages sent from $F_2$ to $F_1$, $MQ_1$ and $Q_1$ and messages sent from $F_1$ to $F_2$ are lost (this is possible as $F_2$ crashes). All other messages of round 2 are delivered by the end of round 2, except for the message sent from $F_1$ to $Q_2$, that is delivered in round 3. Note that, in round 3, $F_1$ plays $(0, -, 1, 1, 1)$, $F_2$ plays $(-, 1, 1, 1, 1)$, $MQ_1$ plays (0,-,1,1,1), $Q_1$ plays $(0, -, 1, 1, 1)$ and $Q_2$ plays (-,1,1,1,1). Note that, at the end of round 2, $F_2$ and $Q_2$ cannot distinguish R3 from R1, so, in round 3 they send the same messages as in round 3 of run R1, including those they send to $l_1$. Learner $l_1$ crashes such that no process receive any round 4 message from $l_1$. In round 4, the correct proposer $p_x$ proposes 0. Since a correct proposer proposes, eventually a correct learner $l_2$ learns some value $v \in \{0, 1\}$, say at round $K$.

**R4:** All processes are correct except (1) the proposer $p_l$ and the acceptors $MQ_1$ and $Q_1$, which are malicious, and (2) learner $l_1$ which crashes before sending any message in round 4. In round 1, proposer $p_l$ sends $m0$ to acceptor $F_1$, and $m1$ to $F_2$, $MQ_1$, $Q_1$ and $Q_2$. All messages in round 2 are delivered as in R3. From round 3 up to round $K$, all non-crashed processes receive messages from all other non-crashed processes distinct from $F_2$. Only round 3 message from $F_2$ to $l_1$ is de-




(a) Run 1 (b) Run 2 (c) Run 3 (d) Run 4 (e) Run 5 (f) Legend

**Fig. 3.** Illustration of proof L.3: lower bound on combining very fast learning (with authentication) and fast learning without authentication - configuration $C_{1a}$

livered, and the delivery of all other messages sent by $F_2$ up to round $K$, is delayed until round $K+1$. Furthermore, in round 3, malicious acceptors $MQ_1$ and $Q_1$ play $(-,1,1,1,1)$ instead of $(0,-,1,1,1)$ (as if they had received the same round 2 messages as in R1) to $l_1$ (this is possible as messages exchanged among acceptors in round 2 are not authenticated), and to all other processes play according to the algorithm from that point on. Learner $l_1$ upon receiving round 3 messages from $F_2$, $(M-Q)_1$, $Q_1$ and $Q_2$, learns 1, as $l_1$ cannot distinguish R4 from R1 (because $l_1$ receives the same set of messages from the acceptors in round 3 of both runs). Then learner $l_1$ crashes before sending any message in round 4. In round 4, the correct proposer $p_x$ proposes 0. Up to round $K$, all non-crashed processes receive messages from all other non-crashed processes distinct from $F_2$ (i.e., all messages sent by $F_2$ up to round $K$ are delayed until round $K+1$). At the end of round $K$, no correct process can distinguish R4 from R3 (because every acceptor different from $F_2$ plays identical values in both runs). Hence, learner $l_2$ learns $v \in \{0,1\}$ at round $K$. All correct processes receive messages from all other correct processes (including $F_2$) in rounds higher than $K$.

**R5:** All processes are correct except (1) the proposer $p_l$ and the acceptors $MQ_1$ and $Q_2$, which are malicious, and (2) the learner $l_1$ which crashes before sending any message in round 3. In round 1, proposer $p_l$ sends $m1$ to acceptor $Q_1$, $m0$ to all other correct acceptors and both $m0$ and $m1$ to $MQ_1$ and $Q_2$. In round 2, acceptor $Q_1$ plays 1 to all processes, acceptors $F_1$ and $F_2$ play 0 to all processes. However, the malicious acceptors $MQ_1$ and $Q_2$ play 0 to learner $l_1$ and play 1 to all other processes. All round 2 messages are delivered as in R3. Furthermore, in round 3, $Q_2$ plays $(-,1,1,1,1)$ and $MQ_1$ plays $(0,-,1,1,1)$ (as per algorithm).[3] Learner $l_1$ receives round 2 message from all acceptors except $Q_1$. Clearly, at the end of round 2, $l_1$ cannot distinguish R5 from R2 (because $l_1$ receives the same set of messages from the acceptors in round 2 of both runs), and hence, learns 0. Then learner $l_1$ crashes before sending any message in round 3. In round 4, the correct proposer $p_x$ proposes 0. From round 3 up to round $K$, all non-crashed processes receive messages from all other non-crashed processes distinct from $F_2$ (i.e., all messages sent by $F_2$ up to round $K$ are delayed until round $K+1$.) At the end of round $K$, no correct process can distinguish R5 from R3 (because every acceptor different from $F_1$ plays identical values in both runs), and hence, learner $l_2$ learns $v \in \{0,1\}$ at round $K$. All non-crashed processes receive messages from all other non-crashed processes (including $F_2$) in rounds higher than $K$.

Clearly, either R4 or R5 violates *Agreement*: $l_2$ decides $v$ in both runs, but $l_1$ decides 1 in R4 and 0 in R5. However, in both runs, at most M acceptors are malicious: a contradiction with the requirement that $A$ does not violate *Validity* and *Agreement* if $M$ acceptors are malicious.

In the case where $Q > M$, again we divide the set of acceptors, this time into four sets $F_1$ and $F_2$, of size at most $F$, and $M_1$ and $M_2$, of size at most $M$. To prove the lower bound in this case, we can use similar runs we used in case where $M \geq Q$, where acceptor $M_1$ plays the role of acceptor $Q_1$ and $M_2$ the role of $Q_2$, and where acceptor $MQ_1$ does not exist. □

Now, we prove part 2 of the theorem, for configuration $C_2$.

### 4.2 Configuration $C_2$

First, we sketch the proof of the lower bound on the number of acceptors required for very fast learning, even with authentication, in configuration $C_2$. We refer to this proof as proof L.4.

[3] The absence of authentication is exploited also in this point of the proof, where $Q_2$ is allowed to play $(-,1,1,1,1)$ in round 3, although $Q_2$ falsely claims that it received 1 from a correct acceptor $F_2$ in round 2. If the messages exchanged among acceptors were authenticated in round 2 this would not be possible.

**Proposition L.4.** Let $A$ be any algorithm and $p_l$ the only privileged proposer, which is, furthermore, also an acceptor. If, in every very favorable run of $A$ every correct learner learns a value by round 2, then $N_a > 2Q + F + 2M - 2$.

**Proof L.4 (sketch).** The proof is again a simple modification of proof L.1. However, if we try to directly apply that proof, since $p_l$ is now an acceptor, in run R4 (and R5) $M + 1$ acceptors are faulty, and in run R3 $F$ acceptors are crash-stop faulty and one acceptor is malicious. Hence, from the property of $A$, *Agreement* need not hold in $R4$ and $R5$, and *Termination* need not hold in $R3$. Consequently, we cannot show the desired contradiction. Thus we modify proof L.1, such that $M_1$ and $M_2$ have only $M - 1$ acceptors each, and $F_1$ has $F - 1$ acceptors.

Suppose by contradiction that $N_a \leq 2Q+F+2M-2$. Then, we can divide the set of acceptors, that are distinct from $p_l$, into five sets, $Q_1$, $Q_2$, $F_1$, $M_1$ and $M_2$, where the first two sets are of size at most $Q$, the third set is of size at most $F - 1$, and the last two sets are of size at most $M-1$, respectively. In the runs, $p_l$ acts as proposer, as well as an acceptor (sending messages to the learner and other acceptors from round 2). For each run, in round 1, $p_l$ receives the same message as processes in $M_1$, and in higher rounds, plays the same value as the processes in $M_1$. We continue as in the proof of L.1 to obtain a contradiction. The diagrams depicting the runs are presented in Figure 4. Notice that, since $M_1$ and $M_2$ are each of size at most $M - 1$, in runs R24 and R25 run at most $M$ acceptors are faulty, and in run R23 at most $F$ acceptors are faulty ($p_l$ being the additional faulty acceptor). $\square$

Now, we prove another bound on the possibility of very fast learning in configuration $C_2$, with a restriction that authentication cannot be used for very fast learning. We refer to this proof as proof L.5.

**Proposition L.5.** Let $A$ be any algorithm and $p_l$ the only privileged proposer, which is, furthermore, also an acceptor. If, in every very favorable run of $A$ every correct learner learns by round 2 without using authentication, then $N_a > 2Q + F + 2M - 2$.

**Proof L.5.** Suppose by contradiction that $N_a \leq Q + F + 2M$. Then, we can divide the set of acceptors, that are distinct from $p_l$, into four sets, $Q_1$, $F_1$, $M_1$ and $M_2$, where the first set is of size at most $Q$, the second set is of size at most $F-1$, and the last two sets is of size $M$. In the following we say that an acceptor $a_i$ *plays 2* in a run if no process different from $a_i$ can distinguish this run from some run in which $a_i$ does not receive any message in round 1. We now construct five partial runs to derive a contradiction. The runs are depicted in Figure 5; we give short descriptions below.

**R31 and R32:** Runs R1 and R2 are very favorable partial runs in which correct proposer $p_l$ proposes 1 and 0 respectively, and $l_1$ receives messages from all acceptors except $Q_1$ in round 2. From the property of $A$, it follows that $l_1$ learns 1 and 0, respectively, at the end of round 2. Subsequently, $l_1$ crashes before sending any message in round 3.

**R33:** In run 3, every process except $p_l$, $l_1$ and $F_1$ is correct. Malicious proposer $p_l$ sends $m1$ to $M_2$, $m0$ to $M_1$, and does not send messages to $F_1$ and $Q_1$. $F_1$ and $p_l$ crash before sending any message in round 2. (Note that at most $F$ acceptors crash.) From round 2, $M_1$ plays 0, $M_2$ plays 1, and $Q_1$ plays 2. Proposer $p_x$ proposes 0 in round 3. From the *Termination* property of $A$, it follows that learner $l_2$ decides some value $v \in \{0, 1\}$, say at round $K$.

**R34:** Every process except $M_1$ and $l_1$ are correct. Proposer $p_l$ proposes 1 and sends $m1$ to all acceptors, of which, the message to $Q_1$ is lost. From round 2 onwards, $p_l$, $M_2$ and $F_1$ play 1 to all processes, $Q_1$ plays 2 to all processes, and malicious acceptor $M_1$ plays 1 to $l_1$ and 0 to all other

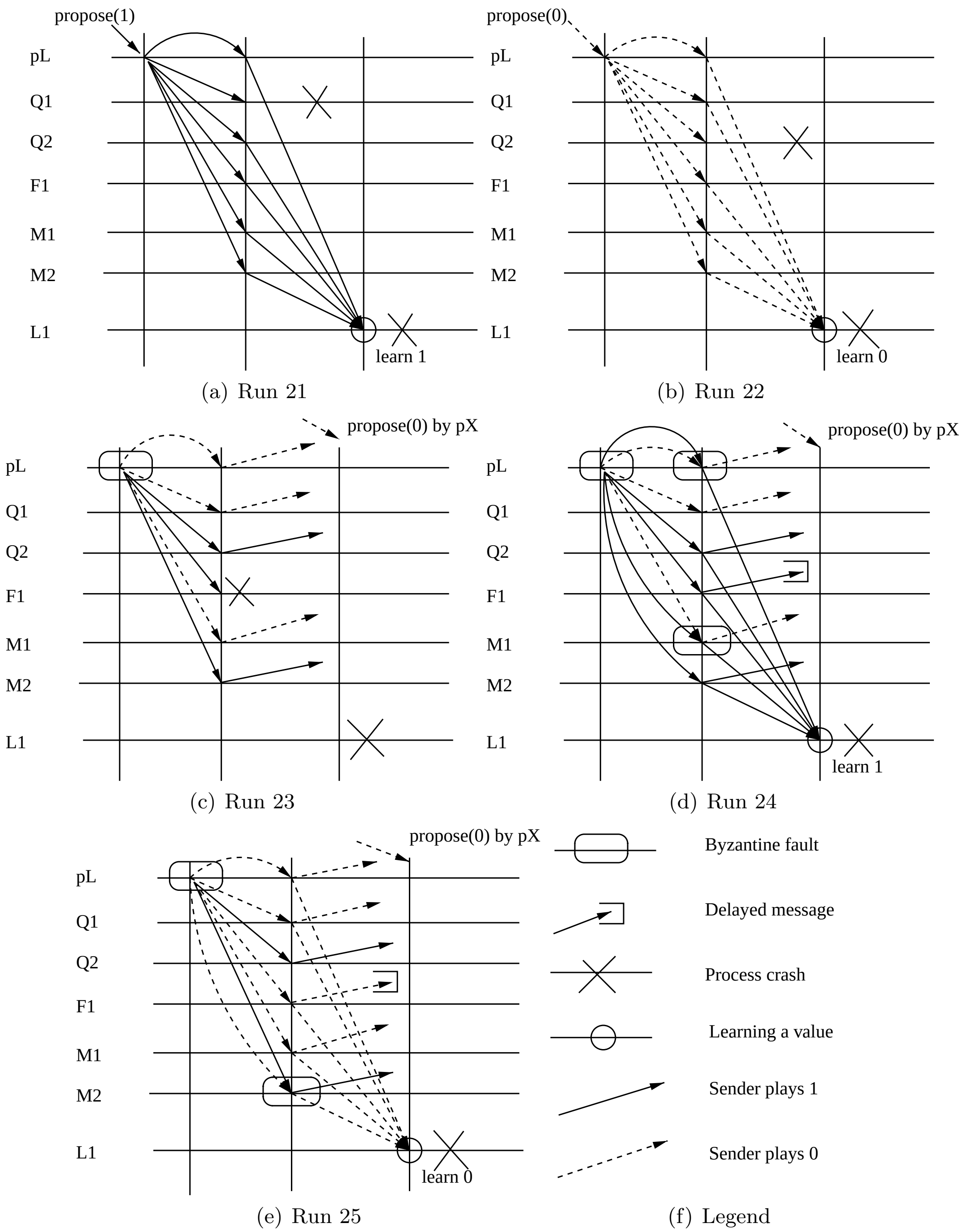



**Fig. 4.** Illustration of proof L.4: lower bound on the possibility of very fast learning in configuration $C_2$

processes. At the end of round 2, $l_1$ cannot distinguish R34 from R31, and hence, learns 1, and then crashes before sending any message in round 3. Proposer $p_x$ proposes 0 in round 3. However, from round 3 to round K, all messages send by $p_l$ and $F_1$ are lost. At the end of round $K$, learner $l_2$ cannot distinguish R34 from R33, and hence, learns $v$.

**R35:** This run is similar to R34, except that $p_l$ proposes 0, and instead of $M_1$, $M_2$ is malicious: it plays 0 to $l_1$ and plays 1 to all other processes. Learner $l_1$ cannot distinguish R35 from R32, and hence learns 0, and then crashes. As in R34, at the end of round $K$, $l_2$ cannot distinguish R35 from R33 and decides $v$.

Clearly, either R34 or R35 violates consensus *Agreement*: $l_2$ decides $v$ in both runs, but $l_1$ decides 1 in R34 and 0 in R35. However, in both runs, at most M acceptors are faulty: a contradiction with the requirement that $A$ does not violate *Validity* and *Agreement* if $M$ processes are faulty. □

It is easy to see that runs R34 and R35 in proof L.5 cannot be constructed when authentication is used in the first communication round (for very fast learning): M1 cannot play 1 as well as play 0 on receiving only message $m1$ from correct proposer $p_l$ (and similarly for M2 in R35). To circumvent this problem in the presence of authentication, we need to assume that $p_l$ is malicious in R34 and R35, and hence (to maintain the upper bound $M$ on the number of malicious acceptors), $M1$ and $M2$ each contains $M-1$ acceptors. This gives us a lower bound of $Q+F+2M-2$ on $N_a$. However, this bound is strictly weaker than the bound $N_a > 2M + F + 2Q - 2$, shown by proof L.4.

Finally we show how to modify proof L.3 to prove the rest of part 2 of the theorem, i.e., to prove the bound on the number of acceptors required for achieving fast learning without authentication combined with very fast learning (with or without authentication) in configuration $C_2$. We refer to this proof to proof L.6.

**Proposition L.6.** Let $A$ be any algorithm and $p_l$ the only privileged proposer, which is, furthermore, also an acceptor, such that, in every very favorable (resp. favorable) run of $A$ every correct learner learns a value by round 2 (resp.3). In addition, let $m$ be the message that is used for learning in 3 rounds (fast learning), but $m$ is not used for learning in 2 rounds (very fast learning). Suppose also that in $A$ no such a message $m$ is authenticated. Then, $N_a > 2F+M-1+min(M-1, Q)$.

**Proof L.6 (sketch).** Basically, the proof relies on proof L.3 and we show how it can be reused in configuration $C_2$, in a similar way we reused proof L.1 in proof L.4. Namely, we use the runs similar to runs R1-R5 of proof L.3; we only change the size of acceptors sets. Again, we distinguish two cases: (1) the case where $M \geq Q + 1$, and (2) the case where $M < Q + 1$. In the case (1), where $M \geq Q + 1$, we adapt proof L.3 in the following way: the size of the set $F_2$ is now at most $F - 1$ (instead of at most $F$) and the size of the set $MQ_1$ is now at most $M - Q - 1$ (instead of at most $M - Q$). Finally, one additional proposer/acceptor plays the roles of both the privileged proposer $p_l$ and the acceptor that belongs to the set $F_2$ in proof L.3.

In the case (2), where $Q + 1 > M$, again we divide the set of acceptors, this time into four sets: $F_1$, of size at most $F$, $F_2$, of size at most $F - 1$, and $M_1$ and $M_2$, of size at most $M - 1$. To prove the lower bound in this case, we can use similar runs we used in proof L.3, where acceptor $M_1$ plays the role of acceptor $Q_1$ and $M_2$ the role of $Q_2$, and where acceptor $MQ_1$ does not exist. In addition, as in the case (1), where $M \geq Q + 1$, one additional proposer/acceptor plays the roles of both the privileged proposer $p_l$ and the acceptor that belongs to the set $F_2$ in proof L.3. □

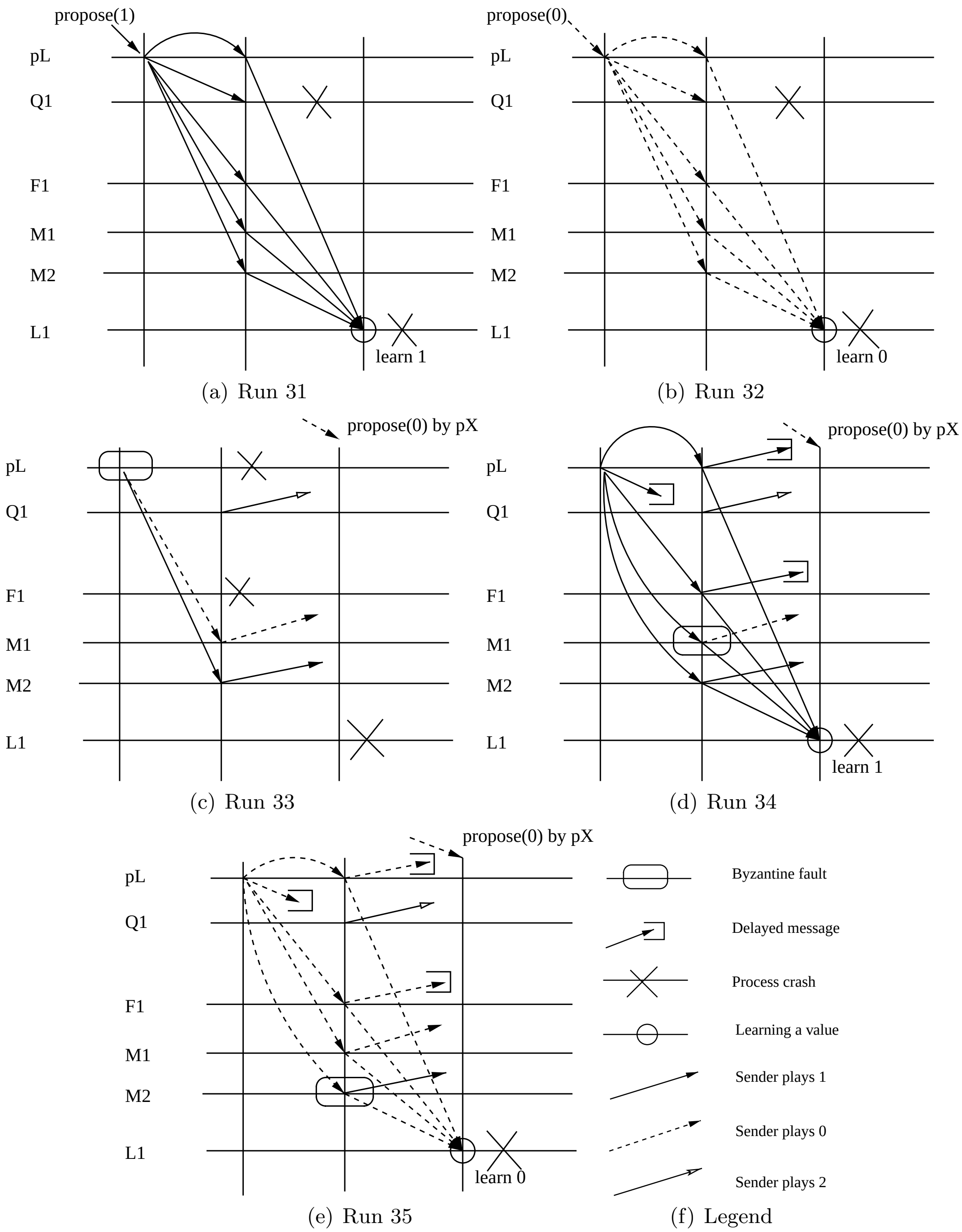



**Fig. 5.** Illustration of proof L.5: lower bound on the possibility of very fast learning without authentication in configuration $C_2$

## 5 The DGV Algorithm

To complete the proof of our theorem we describe here algorithms that match our lower bounds. Our algorithms are all variants of the same Byzantine consensus algorithm.

In the following, we first detail one variant of the algorithm, denoted by $DGV_{Alg.1}$, that matches the lower bounds of part 1 of the theorem, for configuration $C_1$ in the case of a single privileged proposer. In fact, the variant we consider here also matches the interesting case of configuration $C_2$, where all proposers are acceptors and authentication is not used for very fast learning. Namely, we show that:

**Proposition Alg.1.** There is a consensus algorithm $A$, with a single privileged proposer $p_l$, such that: if $p_l$ is not an acceptor (resp. if all proposers are acceptors), then in every very favorable run of $A$ every correct learner learns a value by round 2 without using authentication despite the failure of $Q$ acceptors whenever $N_a > max(2M+Q+2F)$ (resp. $N_a > max(2(M-1)+Q+2F, 2M+Q+F)$). This matches the bound established by proposition L.1 (resp. the bound established by combining propositions L.4 and L.5) from Section 4.

In addition, in every favorable run of $A$, every correct learner learns a value by round 3 despite the failure of $F$ acceptors:

(a) using authentication when $N_a \leq 2F + M + min(M, Q)$ (resp. $N_a \leq 2F + (M-1) + min(M-1, Q)$),

(b) without using authentication when $N_a > 2F + M + min(M, Q)$ (resp. $N_a > 2F + (M-1) + min(M-1, Q)$); (b) matches the bound established by proposition L.3 (resp. L.6) from Section 4.

As we discuss in the following, it is not difficult to modify $DGV_{Alg.1}$ to obtain an algorithm $DGV_{Alg.2}$ that achieves the following:

**Proposition Alg.2.** There is a consensus algorithm $A$, with more than one privileged proposer, such that, in every very favorable run of $A$ in which a single privileged proposer $p_l$ proposes, every correct learner learns a value by round 2 without using authentication despite the failure of $Q$ acceptors whenever $N_a > max(2M + Q + 2F)$. This matches the bound established by proposition L.2 from Section 4.

In addition, in every favorable run of $A$, every correct learner learns a value by round 3 despite the failure of $F$ acceptors:

(a) using authentication when $N_a \leq 2F + M + min(M, Q)$,

(b) without using authentication when $N_a > 2F + M + min(M, Q)$.

We prove proposition Alg.1 (and show how proposition Alg.2 can be derived) by proving the correctness of $DGV_{Alg.1}$ in Section 5.6. Finally, in Section 5.7, we describe DGV variants that match the lower bounds from part 2 of our theorem, in configuration $C_2$ in general, and highlight how DGV can be efficiently adapted to the special case where $Q = F$.

### 5.1 Overview

DGV is composed of two parts: (1) a *Locking* module and (2) an *Election* module. In short, the *Locking* module ensures consensus safety whereas the *Election* module ensures consensus liveness under eventual synchrony assumption. The key element of DGV is its *choose()* function, within the *Locking* module, that determines which value should be accepted by an acceptor at a given point in time. The pseudocodes of *Locking* and *Election* modules are given in Section 5.5, in Figures 8 and 9, respectively.

The algorithm proceeds in a sequence of *views* (Fig. 6). A view is a time frame in which some pre-determined proposer is the *leader*. A leader is the only proposer whose messages are considered by the acceptors within a view. DGV is based on the rotating coordinator paradigm [12], where the leader of the view number $w$ is $p_k$, for $k = w \bmod N_p$. The algorithm starts in the *initial view*, $InitView$, which is a constant known to all processes (e.g., $InitView = 0$). Privileged proposer $p_{Init}$ (where $Init = 0$ for $Initview = 0$), is the leader of $InitView$.

A view leader executes the *Locking* module of DGV, which consists of two phases: *READ* and *WRITE* phase. Basically, the *READ* phase makes sure that, if any value was learned by some learner in some previous view, it will be proposed in the new view. This is determined by the key part of the *READ* phase, the *choose()* function. Since the algorithm starts in $InitView$, if $p_{Init}$ proposes in the $InitView$, $p_{Init}$ skips the *READ* phase and executes only the *WRITE* phase. In the *WRITE* phase, the leader tries to impose to learners its estimate of the decision value, with the intermediation of acceptors. The *WRITE* phase allows very fast learning in very favorable runs and at the same time provides graceful degradation, to allow fast learning in favorable runs. In other words, if a correct privileged proposer $p_{Init}$ proposes at the very beginning of the algorithm, it achieves very fast (resp. fast) learning in a very favorable (resp. favorable) run. If there is more than one privileged proposer (proposition Alg.2), it is not difficult to obtain a variant of DGV where we allow any privileged proposer ($p_x$) to achieve (very) fast learning, provided that $p_x$ is the only proposer that actually proposes a value in a (very) favorable run. This is done by setting $InitView$ to $-1$ and by allowing acceptors to accept a value from any privileged proposer $p_x$ in $InitView$.

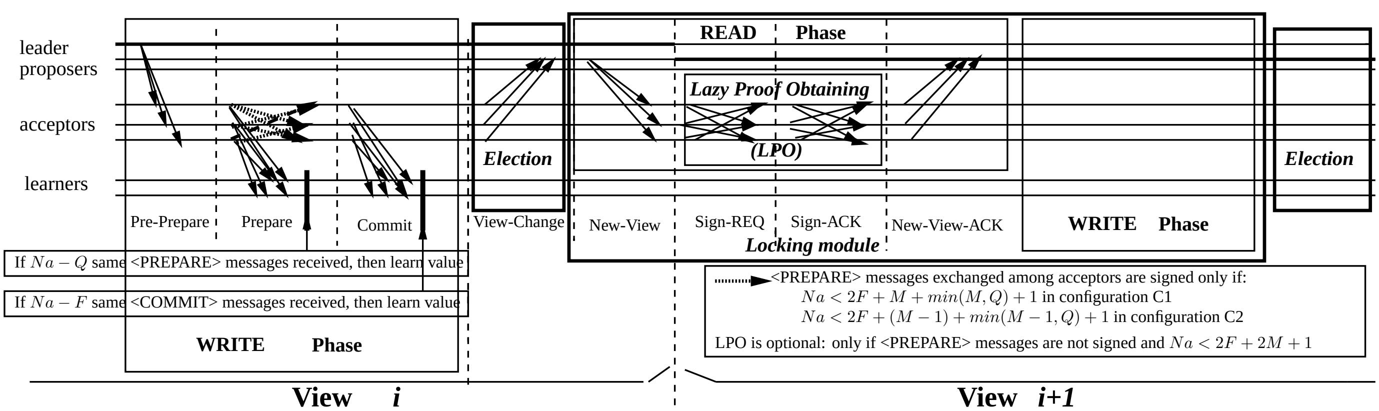



**Fig. 6.** Communication pattern and structure of DGV

A view leader that is suspected of not making any progress is changed on the basis of timeouts within the *Election* module of the algorithm. As soon as the acceptors initialize the algorithm, they start a timer that is permanently stopped as soon as they hear from at least one correct learner that it had learned a value. Otherwise, upon expiration of the timer, the acceptor suspects the leader. If $\lfloor (N_a + M)/2 + 1 \rfloor$ correct acceptors suspect the current leader, the leader is (eventually) changed. The set of $\lfloor (N_a + M)/2 + 1 \rfloor$ acceptors is a *non-malicious majority* set, i.e., every set of size $\lfloor (N_a + M)/2 + 1 \rfloor$, for every run $r$, always contains a majority of *non-malicious* acceptors in $r$. [4].

In the following, we describe the *WRITE* phase of the *Locking* module.

[4] We prove this statement in Section 5.6 (Lemma 2).

### 5.2 $\boldsymbol{WRITE}$ Phase

In the first communication round, the leader sends the *PRE-PREPARE* message to all acceptors, including the proposal value $v$ and the view number $w$, together with the $WriteProof$, set of authenticated messages that certifies the proposal value $v$. We come back to the generation of this set in Section 5.4. For the time being, it is enough to assume that the acceptors can check the validity of the $WriteProof$. In $InitView$, $WriteProof = nil$.

Every acceptor $a_i$, if it is in view $w$, upon reception of the *PRE-PREPARE* message from the leader with the valid $WriteProof$, adds the *PRE-PREPARE* message to its set $K_{a_i}$ (for simplicity, we say that: (a) $a_i$ pre-prepares $v$ in $w$ and (b) $K_{a_i} := (v, w)$). Acceptors pre-prepare a value at most once in the particular view. Then, acceptors begin the second communication round by echoing the *PRE-PREPARE* message to learners, within a *PREPARE* message, with the same value and the view number. The $Writeproof$ set does not have to be echoed. Furthermore, acceptors send the *PREPARE* message, to all other acceptors. If $N_a \leq 2F+M+min(M,Q)$ in the case of configuration $C_1$ or $N_a \leq 2F+(M-1)+min(M-1,Q)$ in the case of configuration $C_2$, the *PREPARE* messages exchanged among the acceptors are authenticated.

Upon reception of $N_a - Q$ *PREPARE* messages from different acceptors, with the same value $v$ and view number $w$, a learner learns $v$. Upon reception of $N_a - F$ *PREPARE* messages from different acceptors, with the same value $v$ and view number $w$, that furthermore match the value and the view number in $K_{a_i}$ and the current view of the acceptor, acceptor $a_i$ adds these *PREPARE* messages to its set $P_{a_i}$. For simplicity, we say that: (a) $a_i$ prepares $v$ in $w$ and (b) $K_{a_i} := (v, w)$ (when we say that $a_i$ accepts $v$ in $w$, we mean that $a_i$ pre-prepares or prepares $v$ in $w$). Then, $a_i$ sends a *COMMIT* message (third communication round) containing $v$ and $w$ to all learners. Upon reception of $N_a - F$ *COMMIT* messages with the same $v$ and view number $w$ from different acceptors, learner learns $v$, unless it had already learned a value.

### 5.3 Changing Leader

Upon initialization, acceptors trigger the timer $SuspectTimeout$, that is initially equal to some value $InitTimeout$, conveniently chosen with respect to the estimates of $\Delta_c$ and $\Delta_{auth}$. If $SuspectTimeout$ expires, the acceptor suspects the current leader. If a sufficient number of acceptors suspect the current leader, then the leader is changed. Basically, the leader of the view is changed if it is faulty, or if the run is not synchronous. This is done within the *Election* module of DGV.

When an acceptor suspects the leader, it sends the signed *VIEW-CHANGE* message to the leader of the next view, doubles the $SuspectTimeout$ and triggers it again. If the new leader is not elected until the expiration of $SuspectTimeout$, the acceptors send signed *VIEW-CHANGE* messages to the next leader, and so on. When some proposer $p_j$ receives $\lfloor (N_a + M)/2 + 1 \rfloor$ *VIEW-CHANGE* messages from different acceptors, with valid signatures and the same view number $w$, such that $w \bmod N_p = j$, $p_j$ becomes the leader (we say $p_j$ is elected). A leader uses a set of received signed *VIEW-CHANGE* messages as the view proof ($ViewProof_w$), the proof that it is a legitimate leader of the view $w$. The new leader sends to all acceptors the *NEW-VIEW* message containing the view number and the view proof. Upon reception of a valid *NEW-VIEW* message for a higher view, an acceptor updates its view number and view proof, and updates the value for future timeouts (line 17, Fig. 9). The values for $SuspectTimeout$ are chosen in such way that all acceptors trigger the same timeout value after sending a *VIEW-CHANGE* message for a particular view number.

$SuspectTimeout$ is stopped when the acceptor receives the confirmation from some learner that it learned a value. When a learner learns a value $v$, it sends (periodically) the signed *DECISION* message that contains a value $v$ to all acceptors and learners (for presentation simplicity, we use authenticated *DECISION* messages from learners, to enable acceptors to halt *Locking* and *Election* modules. The authentication can be avoided in this case by using a variation of consistent broadcast of [4], as we show in Appendix 5.7). When an acceptor receives a *DECISION* message from some learner, it stops permanently the *SuspectTimeout* and halts *Locking* and *Election* modules. Learners that do not learn a value for some time, start to periodically query acceptors for the *DECISION* message. Upon reception of such a query, an acceptor, if it has received a signed *DECISION* message from some learner, forwards the *DECISION* message to all learners. Upon reception of a correctly signed *DECISION* message that contains a value $v$, a learner learns $v$ if it did not already learn a value. Note that a learner can learn a value on the basis of a *DECISION* message that is correctly signed by some learner, because learners are assumed not to be malicious.

### 5.4 *READ* Phase and *choose*() function

Upon being elected, a new leader of view $w$, $p_w$, sends a $\langle NEW-VIEW, w, ViewProof_w\rangle$ message to all acceptors, where $ViewProof_w$ is the proof, based on authenticated messages, that $p_w$ is a legitimate, elected leader of $w$. Upon reception of the *NEW-VIEW* message for view $w$, sent by $p_w$, an acceptor $a_i$, if it is in $w_{a_i} \geq w$, replies to $p_w$ with the signed *NEW-VIEW-NACK* message that includes the valid proof, $ViewProof_{w_{a_i}}$, of the fact that it is in $w_{a_i} \geq w$, (where $ViewProof_{w_{a_i}}$, is view proof $a_i$ received from the leader of $w_{a_i}$).

Else, $a_i$ updates its view number ($w_{a_i}$) to $w$, and its view proof ($ViewProof_{w_{a_i}}$) to $ViewProof_w$. If $N_a-M-2F > M$ or *PREPARE* messages in the *WRITE phase* are authenticated, then $a_i$ replies with the signed *NEW-VIEW-ACK* message, containing its sets $K_{a_i}$ and $P_{a_i}$. Else, if $N_a-M-2F \leq M$ and authentication is not used in *WRITE phase*, $a_i$ sends a $\langle SIGN-REQ, v_{a_i}, w_{a_i}\rangle$ message to the set of acceptors from which $a_i$ received $\langle PREPARE, v_{a_i}, w_{a_i}\rangle$ messages from the set $P_{a_i}$ (when $N_a - M - 2F \leq M$, acceptors have to keep track of *PREPARE* messages they have sent). Upon reception of *SIGN-REQ* message, acceptors respond with a signed *SIGN-ACK* message that contains a signed *PREPARE* message corresponding to request *SIGN-REQ* if they have sent that *PREPARE* message. As liveness has to be guaranteed only if at most $F$ acceptors fail, in this case, a non-malicious acceptor $a_i$ is guaranteed to obtain $N_a - 2F \geq M + 1$ *SIGN-ACK* messages. The acceptor $a_i$ includes the $N_a - 2F$ received signed *SIGN-ACK* messages in the *NEW-VIEW-ACK* message that it sends to the leader of the new view. The pair $(v_{a_i}, w_{a_i})$ reported by $P_{a_i}$ in the *NEW-VIEW-ACK* message sent by $a_i$ is considered valid by the leader of the new view, only if it is accompanied with a matching, valid set of $N_a - 2F$ signed *SIGN-ACK* messages. This technique is a generalization of what is known as a "lazy" proof obtaining (LPO) technique [5].

Upon reception of $N_a - F$ valid *NEW-VIEW-(N)ACK* messages, if there is any valid *NEW-VIEW-NACK* message, the leader updates its view number and aborts its current proposal. If the leader did not receive any *NEW-VIEW-NACK* message, adds $N_a - F$ received *NEW-VIEW-ACK* messages to the set $WriteProof$. We define the *candidate* values of the $WriteProof$ in the following way:

**Definition 1 (Candidate values).** *We say that a value $v$ is* Candidate-2 *or* Candidate-3 *value in the set $WriteProof$, with the* cardinalities *$S_v^2$ and $S_v^3$, respectively, if:*

- *(Candidate-2) $S_v^2 \geq N_a - Q - M - F$ different $K_{a_i}$ sets of* NEW-VIEW-ACK *messages in the $WriteProof$ contain the value $v$ ($K_{a_i} = (v, *)$).*
- *(Candidate-3) $S_v^3 \geq N_a - 2F - M$ different valid $P_{a_i}$ sets of* NEW-VIEW-ACK *messages in the $WriteProof$ contain the value $v$, associated*[5] *with the same view number $w$ ($P_{a_i} = (v, w)$).*

[5] We say that the view number $w$ and the value $v$ are *associated* if there is some set $K_*$ or $P_*$, such that $K_{a_i} = (v, w)$ or $P_{a_i} = (v, w)$. Note that one value can be associated with multiple view numbers and vice versa.

Finally, a new leader $p_w$ chooses the value that it is going to propose to acceptors using the *choose()* function, which we give in Figure 7.

---

**choose**$(v, WriteProof)$ **returns**$(v, view)$ **is** {
$view_2, view_3 := -1$; $v_2, v_3 := nil$; $flag := true$
**sort** all (if any) candidate-3 values by their associated view no.; let $w_3$ be the highest among those view no.
**if** $\exists$ a single candidate-3 value $v_3'$ associated with $w_3$ **then** $v_3 := v_3'$; $view_3 := w_3$
**elseif** $\exists$ more than one such a value **then** $flag := false$; **abort**
**endif**
**if** there is a single candidate-2 value $v'$ **then** $v_2 := v'$;
**elseif** there are two candidate-2 values $v'$ and $v''$ **then**
order sets $K_*'$ and $K_*''$, that contain $v'$ and $v''$, respectively, by descending view numbers
let $view'$ and $view''$ be the view numbers of $M+1^{st}$ highest view number associated to $v'$ and $v''$, respectively.
**if** $view' > view''$ **then** $v_2 := v'$ **elseif** $view'' > view'$ **then** $v_2 := v''$;
**else** % *view'=view''*
**if** *NEW-VIEW-ACK* sent by leader of view $view' = view''$ is in $Writeproof$ **then abort**
**elseif** $S^2_{v'} \geq N_a - Q - F - M + 1$ **then** $v_2 := v'$ **elseif** $S^2_{v''} \geq N_a - Q - F - M + 1$ **then** $v_2 := v''$ **endif**
**endif**
**endif**
**endif**
**if** $v_2 \neq nil$ **then** $view_2 := M + 1^{st}$ highest view number associated to $v_2$ in $K_*$ sets **endif**
**if** $view_2 > view_3$ **then return**$(v_2, view_2)$ **elseif** $view_3 > view_2$ **return**$(v_3, view_3)$ **else**
**if** $view_2 = view_3 \neq -1$ **and** ($v_2 = v_3$ **or** ($v_2 \neq v_3$ **and** $S^3_{v_3} > M$ ) **or** *PREPARE* messages authenticated)
**then return**$(v_3, view_3)$
**elseif** $view_2 = view_3 \neq -1$ **and** $v_2 \neq v_3$ **and** $S^3_{v_3} \leq M$ **and** *PREPARE* messages not authenticated **then**
**if** system configuration is $C_1$ **then** $flag := false$; **abort**
**else** % system configuration is the case of $C_2$ (all proposers are also acceptors)
**if** *NEW-VIEW-ACK* sent by leader of view $view_2 = view_3$ is in $Writeproof$ **then abort**
**else case**
$((S^3_{v_3} \geq N_a - M - 2F + 1$ **and** $S^2_{v_2} < N_a - Q - M - F + 1)$ **or** $S^3_{v_3} = M)$: **return**$(v_3, view_3)$
$(S^3_{v_3} < N_a - M - 2F + 1$ **and** $S^2_{v_2} \geq N_a - Q - M - F + 1)$: **return**$(v_2, view_2)$
$(M > S^3_{v_3} \geq N_a - M - 2F + 1$ **and** $S^2_{v_2} \geq N_a - Q - M - F + 1)$: $flag := false$; **abort**
**endif**
**endif**
**endif**
**endif**
**return**$(v, \perp)$
}

---

**Fig. 7.** *Choose()* function

The function *choose()* has two input parameters: (1) $v$, the initial proposal value of $p_w$ and the $Writeproof$, set of the $N_a - F$ valid *NEW-VIEW-ACK* messages for view $w$. The main idea behind *choose()* is that, if a value $v_2$ (resp. $v_3$) was learned by some learner in some previous view $w_2$ (resp. $w_3$) upon reception of $N_a - Q$ (resp. $N_a - F$) $\langle PREPARE, v_2, w_2\rangle$ (resp. $\langle COMMIT, v_2, w_2\rangle$) messages, then $v_2$ (resp. $v_3$) will certainly be the candidate-2 (resp. candidate-3) value in $Writeproof$ of view $w$ (and every subsequent view). This is true as out of $N_a - Q$ (resp. $N_a - F$) acceptors that sent the same *PREPARE* (resp. *COMMIT*) message to learners, there are at least $N_a - Q - M - F$ (resp. $N_a - 2F - M$) non-malicious acceptors whose *NEW-VIEW-ACK* messages will be part of the $Writeproof$.

However, it may happen that there are multiple candidate values in the $Writeproof$. We say that the candidate-2 value $v_2$ is associated with a view number $view_2$, where $view_2$ is the $M + 1^{st}$ highest view number associated to $v_2$ in $K_{a_i}$ sets of the *NEW-VIEW-ACK* messages that belong to the $Writeproof$. In addition, we say that the candidate-3 value $v_3$ is associated with a view number $view_3$, if for at least $N_a - 2F - M$ valid $P_{a_i}$ sets of the *NEW-VIEW-ACK* messages that belong to the $Writeproof$ $P_{a_i} = (v_3, view_3)$. If there is more than one candidate value in the $Writeproof$ happens, a candidate value with the highest associated view number will be selected. If there are multiple candidate values associated with the same (highest) view number, *choose*() is finely tuned to always return a value that was learned in some previous view (if any), rather than some other candidate value. For details on how this is obtained, we refer the reader to the correctness proof of this DGV variant given in Section 5.6. In any case, if some value $v$ was learned by some learner in some previous view, *choose()* will never return a value different than $v$ (to ensure *Agreement*). Informally, when there is a dispute between two (or even more) candidate values with the same associated view number $w$, where one of the candidate values was actually learned in some previous view, either: (a) a leader of $w$ was malicious (in case this proposer is also an acceptor), or (b) the $Writeproof$ contains malicious acceptors. In case (a), if the $Writeproof$ contains the message from the leader of $w$, *choose*() aborts. If this is not the case, we exploit the fact that one malicious acceptor (leader of view $w$) is out of the $Writeproof$ so we adapt our calculations with respect to this (e.g., a candidate-2 value that was actually learned will have a cardinality of at least $N_a - Q - M - F + 1$ in the $Writeproof$, see lines 13-14, Fig. 7). In case (b), where malicious acceptor is not necessarily the leader of the disputed view $w$, *choose*() aborts again.

When $choose(v, Writeproof)$ aborts (lines 5, 13, 23, 25 and 29, Fig. 7), we are sure that the $Writeproof$ contains at least one malicious acceptor. Therefore, a new leader can wait for one additional *NEW-VIEW-ACK* message ($N_a - F + 1^{st}$), when it invokes *choose*() on every possible valid $Writeproof$, i.e. on every subset of received *NEW-VIEW-ACK* messages of size $N_a - F$. If choose aborts on every such subset, new leader waits for another *NEW-VIEW-ACK* message and so on. Termination is guaranteed in presence of $N_a - F$ correct acceptors as *choose*() never aborts when the $Writeproof$ consists of *NEW-VIEW-ACK* messages sent only by the correct acceptors.

Upon finding a $Writeproof$ for which *choose*() returns a value $v$, the new leader sends the *PRE-PREPARE* message to all acceptors, in the same way as the leader of $InitView$, except that this time $WriteProof \neq nil$. An acceptor checks the $Writeproof$ (as mentioned in Section 5.2) using the same *choose*() function and accepts the *PRE-PREPARE* message if the proposed value $v$ can be extracted from the $WriteProof$. Then the *WRITE* phase continues as described in Section 5.2.

### 5.5 Modularizing DGV

We distinguish two main parts of the DGV algorithm. One is the *Locking* part of the algorithm, described in Figure 8, which consists of the *READ* and the *WRITE* phase. This part of the algorithm captures the *Safety* properties of the algorithm - *Validity* and *Agreement*. The two phases of the *Locking* part are explained in Section 5.

Note that at lines 34 and 37 in the *Locking* module (Fig. 8), in the *WRITE* phase, acceptors and learners can wait for $\lfloor (N_a + M)/2 + 1 \rfloor$ instead of $N_a - F$ *PREPARE* and *COMMIT* messages, respectively, when the *PREPARE* messages exchanged among acceptors in the *WRITE* phase are authenticated (line 32, Fig. 8. The set of $\lfloor (N_a + M)/2 + 1 \rfloor$ acceptors is a *non-malicious majority* set, i.e., every set of size $\lfloor (N_a + M)/2 + 1 \rfloor$, for every run $r$, always contains a majority of *non-malicious* acceptors in $r$. This optimization of DGV makes it possible to have, in the described case, fast learning in the synchronous run in which a privileged proposer is correct, despite the failure of $N_a - \lfloor (N_a + M)/2 + 1 \rfloor \geq F$ acceptors.

The second part of the algorithm is the *Election* module, which is described in Figure 9. The *Election* module, under the assumption of an eventually synchronous system, provides liveness. This part of the algorithm elects new leaders on the basis of timeouts.

In Figure 10 we give the simple wrap-up algorithm. Upon entering a view in which proposer $p_j$ is a leader (this is done within a *Election* module of the algorithm, $p_j$ executes the *Locking* module

**at every proposer** $p_j$**:**
**propose**$(v, w, ViewProof_w)$ **is**
$WriteProof := nil$
**if** $(w \neq Initview)$ **then**
% *READ phase*
send to all acceptors $\langle NEW-VIEW, w, ViewProof_w \rangle$
**wait until** reception of $N - F$ valid signed
$\langle NEW-VIEW-(N)ACK, w, K_{a_i}, P_{a_i}, proof_{P_{a_i}}, ViewProof_{w_{a_i}}, w_{a_i} \rangle$ messages,
where $\forall K_{a_i} = (*, view < w)$ and $\forall P_{a_i} = (*, view < w)$
$WriteProof :=$ set of $N_a - F$ received $\langle NEW-VIEW-(N)ACK \rangle$ messages
**if** received any valid $\langle NEW-VIEW-NACK \rangle$ message **then**
$w :=$ highest valid $w_{a_i}$ from $WriteProof$
$ViewProof_w := ViewProof_{w_{a_i}}$ corresponding to view $w$
**return**
**else choose**$(v, WriteProof)$ **endif**
**end**

% *WRITE phase*
send to all acceptors $\langle PRE-PREPARE, v, w, WriteProof \rangle$

**at every acceptor** $a_j$**:**
% *READ phase*
**upon** reception of $\langle NEW-VIEW, w, ViewProof_w \rangle$ from $p_i$
**if** $(w_{a_j} < w)$ **and** $(ViewProof_w$ matches $w)$ **then**
$w_{a_j} := w$; $ViewProof_{view_{a_j}} := ViewProof_w$
$proof_{P_{a_j}} := nil$
**if** $(2F + M + min(M, Q) < N_a \leq 2F + 2M$ in configuration $C_1)$ **or**
$(2F + M - 1 + min(M-1, Q) < N_a \leq 2F + 2M$ in configuration $C_2)$ **then** % *LPO*
send $\langle SIGN-REQ, P_{a_j}.v, P_{a_j}.w \rangle$ to all acceptors whose $\langle PREPARE, v, w \rangle$ message is in $P_{a_j}$
**upon** reception of $N_a - 2F$ signed $\langle SIGN-ACK \rangle$ messages that correspond to sent $\langle SIGN-REQ \rangle$
$proof_{P_{a_j}} :=$ set of received signed $\langle SIGN-ACK \rangle$ messages
**endif**
send signed $\langle NEW-VIEW-ACK, w, K_{a_j}, P_{a_j}, proof_{P_{a_j}}, nil, nil \rangle$ to $p_i$
**else**
send signed $\langle NEW-VIEW-NACK, w, nil, nil, nil, proof_{a_j}, w_{a_j} \rangle$ to $p_i$
**upon** reception of $\langle SIGN-REQ, v, w \rangle$ from $a_i$ % *LPO*
**if** $\langle PREPARE, v, w \rangle$ already sent **then** send $\langle SIGN-ACK \langle PREPARE, v, w \rangle_{\sigma_{a_j}} \rangle$ to $a_i$ **endif**
% *WRITE phase*
**upon** reception of $\langle PRE-PREPARE, v, w, WriteProof, fresh \rangle$ from $p_i$, with a valid $WriteProof_w$
**if** $(w_{a_j} = w)$ **and** $(w$ **mod** $N_p = i)$ **and** $\langle PRE-PREPARE, *, w, *, * \rangle$ received for the $1^{st}$ time **and**
$((w_{a_j} = InitView)$ **or** $(v$ matches **choose**$(v, WriteProof)))$ **then**
$K_{a_j} :=$ received $\langle PRE-PREPARE \rangle$ message $\{K_{a_j} := (v, w)\}$
$m := \langle PREPARE, v, w \rangle$
send $m$ to all learners
**if** $(N_a \leq 2F + M + min(M, Q)$ in configuration $C_1)$ **or**
$(N_a \leq 2F + M - 1 + min(M-1, Q)$ in configuration $C_2)$ **then** $m := \langle m, \langle m \rangle_{\sigma_{a_j}} \rangle$
send $m$ to all acceptors
**upon** reception of $N_a - F$ signed $\langle \langle PREPARE, v, w \rangle, \ldots \rangle$ matching $K_{a_j}$, $w = w_{a_j}$
$P_{a_j} :=$ set of received $\langle PREPARE \rangle$ messages $\{P_{a_j} := (v, w)\}$
send to all learners $\langle COMMIT, v, w \rangle$

**at every learner** $l_j$**:**
**upon** reception of $N_a - Q$ $\langle PREPARE, v, w \rangle$ **or** $N_a - F$ $\langle COMMIT, v, w \rangle$ with the same $v, w$
**if** $l_j$ has not yet learned a value **then learn**$(v)$ **endif**

**Fig. 8.** Pseudocode of the DGV Locking module

at every learner $l_j$:
**upon** learning a value $v$
  periodically send signed $\langle DECISION, v\rangle$ to all acceptors and all other learners

**upon** reception of a valid signed $\langle DECISION, v\rangle$
  **if** $l_j$ has not yet learned a value **then learn**(v) **endif**

**upon** value not learned
  **wait** some preset time; send $\langle DECISION - PULL\rangle$ to all acceptors;

at every acceptor $a_j$:
**upon** initialization
  $SuspectTimeout := InitTimeout$
  **trigger**($SuspectTimeout$)

**upon** expiration of ($SuspectTimeout$)
  $SuspectTimeout := SuspectTimeout * 2$
  $NextView_{a_j} := NextView_{a_j} + 1$; $NextLeader = NextView_{a_j} \bmod N_p$
  send to $p_{NextLeader}$ $\langle VIEW - CHANGE, NextView_j\rangle_{\sigma_{a_j}}$
  **trigger**($SuspectTimeout$)

**upon** reception of a valid $\langle NEW - VIEW, w, ViewProof_w\rangle$, such that $w > w_{a_j}$;
  $NextView_{a_j} := w$
  $SuspectTimeout := InitTimeout * 2^w$

**upon** reception of a valid $\langle DECISION, v\rangle$ from some learner;
  **stop**($SuspectTimeout$)

**upon** reception of a $\langle DECISION - PULL\rangle$ from some learner $l_j$
  **if** received a valid signed $\langle DECISION, v\rangle$ **then**
    forward $\langle DECISION, v\rangle$ to $l_j$
  **endif**

at every proposer $p_j$:
**upon** reception of $\lfloor (N_a + M)/2 + 1\rfloor$ signed $\langle VIEW - CHANGE, NextView_{a_i}\rangle$ with the same $NextView_{a_i}$
  **if** ($NextView_{a_i}$ **mod** $N_p = j$) **and** ($NextView_{a_i} > w_{p_j}$) **then**
    $ViewProof_{w_{p_j}} := \cup$ received signed $\langle VIEW - CHANGE, NextView_{a_i}\rangle$
    $w_{p_j} := NextView_i$
    send to all proposers signed $\langle NEW - VIEW, w_{p_j}, ViewProof_{w_{p_j}}\rangle$

upon reception of a valid signed $\langle NEW - VIEW, w', ViewProof_{w'}\rangle$, such that $w' > w_{p_j}$
    $w_{p_j} := w'$
    $ViewProof_{w_{p_j}} := ViewProof_{w'}$

**Fig. 9.** Pseudocode of the DGV Election module

of the algorithm. We assume that privileged proposer proposes in the *Initview* due to an external event. To achieve (very) fast learning privileged proposer should propose at the very beginning of the algorithm.

---

**at every process** $PR_j$:
1: $w_{PR_j} := InitView := NextView_{a_j} := 0$ {*Initialization*}
2: $ViewProof_{w_{PR_j}} := nil$

3:**upon**($w_{p_j}$ **mod** $N_p = j$) **and** ($w_{p_j} \neq InitView$)
4: **propose**($v, w_{p_j}, ViewProof_{w_{p_j}}$) {***propose***() *can be invoked also due to an external event*}

---

**Fig. 10.** Pseudocode of the DGV Wrap-Up algorithm

### 5.6 DGV Correctness

In this section, we prove the correctness of the DGV variation described in Section 5 ($DGV_{Alg.1}$, i.e., we prove proposition Alg.1. First, we give few definitions.

**Definition 2 (Value learned in a view).** *We say that a value $v$ is* Learned-2 *or* Learned-3 *in view $w$, if there is a learner $l$ that eventually learns a value by receiving (respectively):*

- *(Learned-2) $\langle PREPARE, v, w\rangle$ messages from $N_a - Q$ different acceptors.*
- *(Learned-3) $\langle COMMIT, v, w\rangle$ messages from $N_a - F$ different acceptors.*

**Definition 3 (Pre-prepares).** *We say that an acceptor $a_i$* pre-prepares *a value $v$ in view $w$, if it eventually adds a $\langle PRE-PREPARE, v, w, *, *\rangle$ message to its $K_{a_i}$ set, i.e., if eventually $K_{a_i} := (v, w)$ (line 29, Fig. 8).*

**Definition 4 (Prepares).** *We say that an acceptor $a_i$* prepares *a value $v$ in view $w$, if it eventually adds a $N_a - F$ different signed $\langle PREPARE, v, w\rangle$ messages to its $P_{a_i}$ set, i.e., if eventually $P_{a_i} := (v, w)$ (line 35, Fig. 8).*

**Definition 5 (Accepts).** *We say that an acceptor $a_i$* accepts *a value $v$ in view $w$, if it pre-prepares or prepares $v$ in view $w$.*

It is trivial to see that if a learner learns a value, it was Learned-2 or Learned-3 in some view. Note that if non-malicious acceptor $a_i$ prepared a value $v$ in view $w$, it follows that $a_i$ pre-prepared a value $v$ in view $w$.

We proceed with the correctness proof by proving two simple, yet crucial lemmas.

**Lemma 1.** *$N_a - F \geq \lfloor (N_a + M)/2 + 1\rfloor$.*

*Proof.* Our algorithm assumes $N_a > 2F + M$ (general bound on solvability of consensus). If $N_a = 2F + M + 1$, then $N_a - F = F + M + 1$, while $\lfloor (N_a + M)/2 + 1\rfloor = \lfloor F + M + 3/2\rfloor = F + M + 1$. Therefore, $N_a - F = \lfloor (N_a + M)/2 + 1\rfloor$. On the other hand, if $N_a > 2F + M + 1$, then $N_a \geq 2F + M + 2 \Rightarrow 2N_a - 2F \geq N_a + M + 2$. As $N_a + M + 2 \geq 2\lfloor (N_a + M)/2 + 1\rfloor$, we conclude that $N_a - F \geq \lfloor (N_a + M)/2 + 1\rfloor$.

**Lemma 2.** – *(a) Two sets, $A$ and $B$, each containing at least $\lfloor (N_a + M)/2 + 1\rfloor$ acceptors, intersect in at least one non-malicious acceptor.*

- *(b) Set $A$ of $N_a - F$ acceptors and set $B$ of $\lfloor (N_a + M)/2 + 1 \rfloor$ acceptors, intersect in at least one non-malicious acceptor.*
- *(c) Two sets, $A$ and $B$, each containing at least $N_a - F$ acceptors, intersect in at least one non-malicious acceptor.*
- *(d) Set $A$ of $N_a - Q$ different acceptors and set $B$ of $\lfloor (N_a + M)/2 + 1 \rfloor$ acceptors, intersect in at least one non-malicious acceptor.*
- *(e) Set $A$ of $N_a - Q$ different acceptors and set $B$ of $N_a - F$ different acceptors intersect in at least $N_a - Q - M - F$ non-malicious acceptors.*
- *(f) Every set of at least $\lfloor (N_a + M)/2 + 1 \rfloor$ acceptors is a non-malicious majority.*
- *(g) Every set of at least $N_a - Q - M - F$ acceptors contains at least $M + 1$ acceptors.*

*Proof. (a).* From the inequality $N_a + M + 1 \leq 2\lfloor (N_a + M)/2 + 1 \rfloor$, it is obvious that $A$ and $B$ intersect in at least $M + 1$ acceptors. As at most $M$ acceptors are malicious, we conclude that $A$ and $B$ intersect in at least one non-malicious acceptor.

*(b),(c).* Follow directly from Lemma 1 and part (a) of the lemma.

*(d).* $Q \leq F \Rightarrow N_a - Q \geq N_a - F$. Applying part (b) of the lemma, we conclude that $A$ and $B$ intersect in at least one non-malicious acceptor.

*(e).* Sets $A$ and $B$ intersect in at least $(N_a - Q) + (N_a - F) - N_a = N_a - Q - F$ acceptors, out of which are at most $M$ malicious. Therefore, $A$ and $B$ intersect in at least $N_a - Q - M - F$ non-malicious acceptors.

*(f).* Straightforward from part (a) of the lemma.

*(g).* In configuration $C_1$: $N_a \geq 2M + F + 2Q + 1 \Rightarrow N_a - Q - M - F \geq M + Q + 1 \geq M + 1$.
In configuration $C_2$: $N_a \geq 2M + F + Q + 1 \Rightarrow N_a - Q - M - F \geq M + 1$.

**Lemma 3.** *If two values $v$ and $v'$ are Learned-2 in view $w$, then $v = v'$.*

*Proof.* Suppose $v \neq v'$. From Def. 2, a set $X$ of at least $N_a - Q$ acceptors sent $\langle PREPARE, v, w \rangle$ messages and a set $Y$ of at least $N_a - Q$ acceptors sent $\langle PREPARE, v', w \rangle$ messages. As sets $X$ and $Y$ intersect in at least $N_a - 2Q$ acceptors, out of which at least $N_a - 2Q - M$ are non-malicious, and $N_a \geq 2Q + 2M + F + 1$, we have $N_a - 2Q - M \geq 1$. That is, there exists a non-malicious acceptor that has sent different *PREPARE* messages in the same view: a contradiction.

**Lemma 4.** *If $v$ is Learned-2 in view $w$, and a set of at least $N_a - F$ acceptors sent the $\langle PREPARE, v', w \rangle$ message, then $v = v'$.*

*Proof.* Suppose $v \neq v'$. From Def. 2, a set $X$ of at least $N - Q$ acceptors sent $\langle PREPARE, v, w \rangle$ messages in the view $w$. Let $Y$ be the set of at least $N_a - F$ acceptors that sent $\langle PREPARE, v', w \rangle$. As sets $X$ and $Y$ intersect in at least one non-malicious acceptor $a_i$ (Lemma 2(d,g)), we conclude that $a_i$ sent different *PREPARE* messages in the same view: a contradiction.

**Lemma 5.** *If $v$ is Learned-2 in view $w$, and $v'$ is Learned-3 in the same view $w$, then $v = v'$.*

*Proof.* This Lemma is a simple corollary of the Lemma 4.

**Lemma 6.** *If two values $v$ and $v'$ are Learned-3 in view $w$, then $v = v'$.*

*Proof.* Suppose $v \neq v'$. From Def. 2, a set $X$ of at least $N_a - F$ acceptors sent $\langle COMMIT, v, w\rangle$ messages and a set $Y$ of at least $N_a - F$ acceptors sent $\langle COMMIT, v', w\rangle$ messages. As sets $X$ and $Y$ intersect in at least one non-malicious acceptor $a_i$ (Lemma 2(c)), we conclude that $a_i$ sent different *COMMIT* messages with the same view number: a contradiction.

**Lemma 7.** *No two different values can be learned in the same view.*

*Proof.* Follows directly from Lemmas 3, 5 and 6.

**Lemma 8.** *If choose$(v, WriteProof)$ in view $w'$ returns $(v, w)$ and $v$ is a candidate value in $Writeproof$, then at least one non-malicious acceptor $a_i$ pre-prepared the value $v$ in a view higher or equal to $w$.*

*Proof.* Assume *choose*$(v, Writeproof)$ in view $w'$ returns $v, w$ where $v$ is a candidate-3 value in the $Writeproof$. From Definition 1, it follows that a set $X$ of at least $N_a - 2F - M$ acceptors reported a valid $P_* = (v, w)$. A $P_*$ set is valid if: (a) *PREPARE* messages exchanged among acceptors are signed, (b) *PREPARE* messages are not signed, but $P_*$ is accompanied with a "lazy" proof of $N_a - 2F$ signed *SIGN-ACK* messages (when $N_a \leq 2F + 2M$) and (c) *PREPARE* messages are not signed, but $N_a \geq 2F + 2M + 1$. We prove that in each of these three exhaustive cases, there is a set $Y$ of at least $N_a - F$ acceptors that sent $\langle PREPARE, v, w\rangle$ messages.
Case (a): $P_*$ sets that acceptors from set $X$ reported contain signed *PREPARE* messages from $N_a - F$ acceptors. Applying Lemma 2(f) and Lemma 1, we conclude that a set $Y$ contains at least one non-malicious acceptor $a_i$ that pre-prepared $v$ in view $w$.
Case (b): Every $P_*$ set is basically accompanied with $N_a - 2F$ signatures. As $N_a \geq 2F + M + 1 \Rightarrow N_a - 2F \geq M + 1$ we conclude that at least one of these signatures comes from a non-malicious acceptor $a_i$ that pre-prepared $v$ in view $w$.
Case (c): A cardinality of set $X$ is $S_v^3 \geq N_a - 2F - M \geq M + 1$, i.e., at least one of the $P_*$ sets is reported by the non-malicious acceptor $a_i$ that pre-prepared $v$ in view $w$.

Assume now that $w$ is a candidate-2 value in the $Writeproof$. This implies (Definition 1) that there exists a set $X$ of at least $N_a - Q - M - F$ acceptors that reported that they pre-prepared $v$, out of which a set $Y$ of at least $M + 1$ acceptors pre-prepared $v$ in the view higher or equal to $w$ (note that Lemma 2(g) implies that the set $X$ contains at least $M + 1$ acceptors). As there are at most $M$ malicious acceptors, we conclude that the set $Y$ contains at least one non-malicious acceptor that pre-prepared the value $v$ in a view higher or equal to $w$.

**Lemma 9.** *(Validity) If a learner learns a value $v$, then some proposer proposed $v$.*

*Proof.* If a learner learns $v$ in $w$, $v$ was pre-prepared by $N_a - Q > M$ acceptors or *prepared* by $N_a - F > M$ acceptors in view $w$, i.e. at least one non-malicious acceptor accepted $v$ in $w$.

We prove the following statement using induction on view numbers: *if a non-malicious acceptor accepts $v$, then $v$ was some proposer proposed $v$.*
*Base Step:* We prove that if a non-malicious acceptor accepted $v$ in $Initview$, then some proposer proposed $v$.

As non-malicious acceptors accept only values proposed by $p_{Init}$, we conclude that $v$ was proposed by some proposer.

*Remark:* Again, we highlight that it is impossible to ensure that a malicious proposer $P_{Init}$, on proposing a value, will not pretend that it has proposed a different value. A more precise definition of *Validity* would be: if a learner $l$ learns a value $v$ in run $r$, then there is a run $r'$ (possibly different) such that some proposer proposes $v$ in $r'$, and $l$ cannot distinguish $r$ from $r'$. Proof that corresponds to this *Validity* definition follows the same footsteps as this proof.

*Inductive Hypothesis (IH):* For every view $w, k > w \leq Initview$, if a non-malicious acceptor accepted $v$ in $w$, then some proposer proposed $v$.
*Inductive Step:* We prove the statement is true for the view $k$. In view $k$ acceptor accept only values returned by $choose(*, Writeproof)$, where $Writeproof$ is valid. If $choose(*, Writeproof)$ returns a candidate value $v$, by Lemma 8, some non-malicious acceptor accepted $v$ in view $w, w < k$, and by IH, $v$ was proposed by some proposer. If $choose(*, Writeproof)$ returns $v$ in line 34, Figure 7, then $v$ is initial proposal value of the leader of $k$. We conclude (with the same remark as in the Base Step) that $v$ was proposed by some proposer.

**Lemma 10.** *After sending a* NEW-VIEW-ACK *message for view $w$, a non-malicious acceptor cannot accept a value $v$ with view number $w' < w$.*

*Proof.* It is not difficult to see that this lemma holds, as non-malicious acceptor $a_j$ accept a value $v$ with view number $w'$ only if $a_j$ is in view lower or equal to $w'$. As $a_j$ already replied with a *NEW-VIEW-ACK* message for view $w > w'$ and thus is in view $w_{a_j} \geq w > w'$, $a_j$ cannot accept $v$.

**Lemma 11.** *If $w$ is the lowest view number in which some value $v$ is Learned-2, then no non-malicious acceptor $a_i$ pre-prepares any value $v'$, $v' \neq v$ in any view higher than $w$.*

*Proof.* We prove this lemma by induction on view numbers.
*Base Step:* First, we prove that no non-malicious acceptor $a_i$ can pre-prepare any value different from $v$ in view $w + 1$. A non-malicious acceptor $a_i$ in $w + 1$ pre-prepares a value $v'$ only if the emphchoose() function on the valid $WriteProof$ of view $w + 1$ returns $v'$. Therefore, it is sufficient to prove that for any valid $Writeproof$, *choose(*,Writeproof)* returns $v$.

Assume, without loss of generality, that $v$ was Learned-2 by learner $l$ in view $w$. Then (Def. 2, 3), a set $X$ of at least $N_a - Q$ acceptors pre-prepared $v$ in $w$. As the valid $WriteProof$ of view $w + 1$ consists of *NEW-VIEW-ACK* messages from a set $Y$ of $N_a - F$ acceptors, there is a subset $Z$ of the set $X \cap Y$, of cardinality $S_Z \geq N_a - Q - F - M$, that contains only non-malicious acceptors (Lemma 2(e)). By Lemma 10, every acceptor $a_i \in Z$ pre-prepared $v$ in $w$, before replying with the *NEW-VIEW-ACK* message to the leader of view $w + 1$. In the meantime, no acceptor from $Z$ pre-prepared any other value, as this would mean that it would be in the higher view then $w + 1$ when replying with *NEW-VIEW-ACK* for $w + 1$, which is impossible. Therefore, $\forall a_i \in Z, (K_{a_i} = (v, w)) \in WriteProof$. As $S_Z \geq N_a - Q - F - M$, $v$ is the candidate-2 value in $Writeproof$ of view $w+1$, with $M+1^{st}$ highest view number (that exists, as follows from Lemma 2(g)) equal to $w$. Note that, in the case the size of $Z$ equals $N_a - Q - F - M$ and $v$ was Learned-2 by $l$ in view $w$, then every acceptor $a_j$, out of $F$ acceptors whose *NEW-VIEW-ACK* messages are not in the $WriteProof$, is non-malicious and $a_j$ pre-prepared $v$ in $w$, before $a_j$ replied with the *NEW-VIEW-ACK* for view $w + 1$ (if $a_j$ replied to the *NEW-VIEW* message for view $w + 1$ at all).

If $v$ is the only candidate-2 value in the $WriteProof$, then $v_2 := v$ (line 7, Fig. 7) and $view_2\ assignw$ (line 18, Fig. 7).

If there is another candidate-2 value $v'$[6] with its $M + 1^{st}$ view number $view' < w$ (chosen as in lines 9-10, Fig. 7, again $v_2 := v$ and $view_2 := w$ (line 11, Fig. 7). As it is impossible that $view' > w$ in the valid $Writeproof$ of view $w + 1$ (line 4, Fig. 8), we now consider the case where $view' = w$. In this case, it is not difficult to see that the leader of view $w$ is faulty. Indeed, there is a set of at least $N_a - Q - F - M \geq M + 1$ acceptors (Lemma 2(g)) that accepted $v$ in the view $w$, and another set of at least $M + 1$ acceptors that accepted $v'$ in the view $w$, which implies that there are two non-malicious acceptors which accepted different values in $w$, i.e., the leader of view $w$ is

[6] This case is not possible in configuration $C_1$.

malicious. From the *choose()* function, if this case ($view' = w$) occurs, the valid $Writeproof$ does not contain the *NEW-VIEW-ACK* message from the leader of view $w$ (line 13 of Fig. 7). In this case, the size of the set $Z$ is at least $S_Z = S_v^2 \geq N_a - Q - F - M + 1$, as we are sure that the *NEW-VIEW-ACK* message from at least one malicious acceptor (the leader of view $w$) is not included in the $WriteProof$. As there are no two non-intersecting subsets of size $N_a - Q - F - M + 1$ in the set of size $N_a - F$ (if $N_a - F \geq 2(N_a - Q - M - F + 1)$ then $N_a \leq 2M + F + 2Q - 2$, which would contradict our assumptions on the number of acceptors), $v_2 := v$ at line 14, Fig. 7).

If there is no candidate-3 value $v'$, or if there is such a value with the associated view number $view' < w$, or if $v' = v$, then $choose()$ returns $v, w$ (lines 19-20, Fig. 7). Again, it is not possible that $view' > w$, so we discuss the case where $view' = w$ and $v' neqv$. There are three exhaustive possibilities: (a) *PREPARE* messages exchanged among acceptors are authenticated, (b) *PREPARE* messages exchanged among acceptors are not authenticated and the cardinality of the candidate-3 value is $S_{v'}^3 \geq M+1$ and (c) *PREPARE* messages exchanged among acceptors are not authenticated and $S^3 \leq M$.
In case (a), digital signatures from the sets $P_*$ that contain $v'$, certify that $N_a - F$ different acceptors sent $\langle PREPARE, v', w\rangle$ message. Due to Lemma 4, $v' = v$, a contradiction.
In case (b), existence of at least $M + 1$ $P_*$ sets that contain $v'$, i.e., including at least one that is sent by a non-malicious acceptor, certifies that $N_a - F$ different acceptors sent $\langle PREPARE, v', w\rangle$ message. Similarily as in the case (a), we reach a contradiction.
Consider case (c). If configuration is $C_1$, then the $Writeproof$ is not valid (line 23, Fig. 7. Consider now configuration $C_2$. As in this case, $P_*$ sets are accompanied with "lazy" proofs, every valid $P_{a_i}$ set is certified with at least $N_a - 2F \geq M + 1$ signatures, including at least one signature from non-malicious acceptor. In other words, there are two distinct non-malicious acceptors that accepted different values in $w$, i.e., the leader of view $w$ is malicious. From the *choose()* function, if this case occurs, the valid $Writeproof$ does not contain the *NEW-VIEW-ACK* message from the leader of view $w$ (lines 25, Fig. 7). In this case, the size of the set $Z$ is at least $S_Z = S_v^2 \geq N_a - Q - F - M + 1$, as we are sure that the *NEW-VIEW-ACK* message from at least one malicious acceptor (the leader of view $w$) is not included in the $WriteProof$. Again, there are three exhaustive subcases: (1) $S_{v'}^3 < N_a - M - 2F + 1$, (2) $N_a - M - 2F + 1 \leq S_{v'}^3 < M$ and (3) $S_{v'}^3 = M$. In case (1), *choose()* returns $v$ (line 28, Fig. 7). In case (2), $Writeproof$ is not valid (line 29, Fig. 7). In case (3), as there are at most $M - 1$ messages in the $Writeproof$ are from the malicious acceptors (as the message from the malicious leader of view $w$ is not in the $Writeproof$), one non-malicious acceptor $a_i$ sent $P_{a_i}$ that contains $v'$. Similarly as in the case (a), we conclude that, due to Lemma 4, $v = v'$ - a contradiction.
*Inductive Hypothesis (IH):* Assume that no non-malicious acceptor $a_i$ can pre-prepare any value different from $v$ in any view from $w + 1$ to $w + k$. We prove that no non-malicious acceptor $a_i$ can pre-prepare any value different from $v$ in the view $w + k + 1$.
*Inductive Step:* Again, it is sufficient to prove that any *choose()* function on any valid $Writeproof$ of view $w + k + 1$ returns $v$. From Lemma 2(e), there is a set $Z$ of size at least $N_a - Q - M - F$, that contains only non-malicious acceptors, such that every acceptor in $Z$ pre-prepared $v$ in $w$ and its *NEW-VIEW-ACK* message is part of the $WriteProof$ of view $w + k + 1$. In fact, as the set $Z$ contains only non-malicious acceptors, applying IH yields that $\forall a_i \in Z\ \exists w_i \geq w, (K_{a_i} = (v, w_i)) \in WriteProof$. Therefore, $v$ is the candidate-2 value and the $M + 1^{st}$ highest view number associated with $v$ in $Z$ is $w_v \geq w$. Therefore, *choose(*, Writeproof)* in view $w + k + 1$ can only return $(*, w' \geq w)$. The sets $K_*$ and $P_*$ in the valid $WriteProof$ of view $w + k + 1$ contain only values with associated view numbers up to $w + k$ (line 4, Fig. 8). Let *choose(*, Writeproof)* return $(v', w' \leq w + k)$. If $w' > w$, then $v' = v$ because, by IH a value pre-prepared by any non-malicious acceptor in view $w'$, such that $w < w' \leq w + k$ can be only $v$, and by Lemma 8, in order for

*choose(*, WriteProof)* to return $(v', w')$, one non-malicious acceptor must have pre-prepared $v'$ in $w'$. Now we consider the case where $w' = w = w_v$ and $v' \neq v$ (it is not difficult to see that $v = v'$ results in *choose()* returning $v$).

If $v'$ is another candidate-2 value, then we conclude that there exists one non-malicious acceptor $a_j$ that accepted $v' \neq v$ in some view higher or equal to $w$ (as $w'$ is the $M+1^{st}$ highest view number associated to $v'$). From IH, we know that this view can not be higher than $w$, so we conclude that $a_j$ accepted $v'$ in view $w$. As we know that every acceptor from the set $Z$ accepted $v$ in $w$ and as $Z$ contains only non-malicious acceptors, we conclude that the leader of view $w$ was malicious. In this case, from the modified *choose()* function, the valid $Writeproof$ does not contain the *NEW-VIEW-ACK* message from the leader of view $w$ (lines 13 of Fig. 7). In this case, the size of the set $Z$ is at least $S_Z = S_v^2 \geq N_a - Q - M - F + 1$, as we are sure that the *NEW-VIEW-ACK* message from at least one malicious process (the leader of view $w$) is outside the $WriteProof$. As there are no two non-intersecting subsets of size $N_a - Q - M - F + 1$ in the set of $N_a - F$ acceptors, $v_2 := v$ at line 14, Fig. 7).

Assume $v' \neq v$, with $w' = w$ is a candidate-3 value. Here we use the same reasoning as in the Base step, which is repeated here for completeness. Again, there are three exhaustive possibilities: (a) *PREPARE* messages exchanged among acceptors are authenticated, (b) *PREPARE* messages exchanged among acceptors are not authenticated and the cardinality of the candidate-3 value is $S_{v'}^3 \geq M + 1$ and (c) *PREPARE* messages exchanged among acceptors are not authenticated and $S^3 \leq M$.
In case (a), digital signatures from the sets $P_*$ that contain $v'$, certify that $N_a - F$ different acceptors sent $\langle PREPARE, v', w \rangle$ message. Due to Lemma 4, $v' = v$, a contradiction.
In case (b), existence of at least $M + 1$ $P_*$ sets that contain $v'$, i.e., including at least one that is sent by a non-malicious acceptor, certifies that $N_a - F$ different acceptors sent $\langle PREPARE, v', w \rangle$ message. Similarly as in the case (a), we reach a contradiction.
Consider case (c). If configuration is $C_1$, then the $Writeproof$ is not valid (line 23, Fig. 7. Consider now configuration $C_2$. As in this case, $P_*$ sets are accompanied with "lazy" proofs, every valid $P_{a_i}$ set is certified with at least $N_a - 2F \geq M + 1$ signatures, including at least one signature from non-malicious acceptor. In other words, there are two distinct non-malicious acceptors that accepted different values in $w$, i.e., the leader of view $w$ is malicious. From the *choose()* function, if this case occurs, the valid $Writeproof$ does not contain the *NEW-VIEW-ACK* message from the leader of view $w$ (lines 25, Fig. 7). In this case, the size of the set $Z$ is at least $S_Z = S_v^2 \geq N_a - Q - F - M + 1$, as we are sure that the *NEW-VIEW-ACK* message from at least one malicious acceptor (the leader of view $w$) is not included in the $WriteProof$. Again, there are three exhaustive subcases: (1) $S_{v'}^3 < N_a - M - 2F + 1$, (2) $N_a - M - 2F + 1 \leq S_{v'}^3 < M$ and (3) $S_{v'}^3 = M$. In case (1), *choose()* returns $v$ (line 28, Fig. 7). In case (2), $Writeproof$ is not valid (line 29, Fig. 7). In case (3), as there are at most $M - 1$ messages in the $Writeproof$ are from the malicious acceptors (as the message from the malicious leader of view $w$ is not in the $Writeproof$), one non-malicious acceptor $a_i$ sent $P_{a_i}$ that contains $v'$. Similarly as in the case (a), we conclude that, due to Lemma 4, $v = v'$ - a contradiction.

**Lemma 12.** *If $w$ is the lowest view number in which some value $v$ is Learned-3, then no non-malicious acceptor $a_i$ pre-prepares any value $v'$, $v' \neq v$ in any view higher than $w$.*

*Proof.* We prove this lemma by induction on view numbers.
*Base Step:* First, we prove that no non-malicious acceptor $a_i$ can pre-prepare any value different from $v$ in view $w + 1$. A non-malicious acceptor $a_i$ in $w + 1$ pre-prepares a value $v'$ only if the emphchoose() function on the valid $WriteProof$ of view $w + 1$ returns $v'$. Therefore, it is sufficient to prove that for any valid $Writeproof$, *choose(*, Writeproof)* returns $v$.

Assume, without loss of generality, that $v$ was Learned-3 by learner $l$ in view $w$. Then (Def. 2, 3), a set $X$ of at least $N_a - F$ acceptors pre-prepared $v$ in $w$. As the valid $WriteProof$ of view $w+1$ consists of *NEW-VIEW-ACK* messages from a set $Y$ of $N_a - F$ acceptors, there is a non-empty subset $Z$ of the set $X \cap Y$ that contains only non-malicious acceptors with cardinality $S_Z = S_v^3 \geq N_a - 2F - M$. By Lemma 10, every acceptor $a_i \in Z$ pre-prepared and prepared $v$ in $w$, before replying with the message *NEW-VIEW-ACK* to the leader of view $w+1$. Therefore, $\forall a_i \in Z, (P_{a_i} = (v, w)) \in WriteProof$, i.e. $v$ is the candidate-3 value. In a valid $Writeproof$ there are no two candidate-3 values with the same view number (line 5, Fig. 7). As $w$ is the highest view number that can appear in the $WriteProof$ of $w+1$, it follows that the $v_3 := v$, $view_3 = w$ at line 4, Figure 7.

Similarly there cannot be a candidate-2 value $v' \neq v$, with $M + 1^{st}$ highest associated view number $w' > w$ in the valid $Writeproof$ of view $w+1$. Let $v_2$ be the candidate-2 value selected by the lines 7-17 of Figure 7, with associated view number $view_2$ (line 18, Fig. 7). If there is no such a value, or if $view_2 < w$, or if ($v_2 \neq v$ and $S_v^3 > M$), or if *PREPARE* messages exchanged among acceptors are authenticated, or if, finally, $v_2 = v$, then *choose(*, Writeproof)* returns $(v, w)$ (lines 19-21, Fig. 7). Now we consider the only possible case left, the case where: (a) *PREPARE* messages exchanged among acceptors are not authenticated, (b) $view_2 = w$, (c) $v_2 \neq v$ and (d) $S_Z = S_v^3 \leq M$. If system configuration is $C_1$, then the $Writeproof$ is not valid. Consider now configuration $C_2$. As in this case, $P_*$ sets are accompanied with "lazy" proofs, every valid $P_{a_i}$ set is certified with at least $N_a - 2F \geq M + 1$ signatures, including at least one signature from non-malicious acceptor. Furthermore, as $v_2$ is a candidate-2 value with associated view number $view_2 = w$, there are at least $M + 1$ (Lemma 2(g)) acceptors, out of which at least one is non-malicious, that accepted $v_2 \neq v$ in $w$. In other words, there are two distinct non-malicious acceptors that accepted different values in $w$, i.e., the leader of view $w$ is malicious. From the *choose()* function, if this case occurs, the valid $Writeproof$ does not contain the *NEW-VIEW-ACK* message from the leader of view $w$ (line 25, Fig. 7). In this case, the size of the set $Z$ is at least $S_Z = S_v^3 \geq N_a - Q - F - M + 1$, as we are sure that the *NEW-VIEW-ACK* message from at least one malicious acceptor (the leader of view $w$) is not included in the $WriteProof$. From the lines 27-29, Figure 7 it is not difficult to see that, in this case, either *choose(*, Writeproof)* returns $(v, w)$ (line 27), or the $Writeproof$ is not valid (line 29).

*Inductive Hypothesis (IH):* Assume that no non-malicious acceptor $a_i$ pre-prepares any value different from $v$ in any view from $w+1$ to $w+k$. We prove that no non-malicious acceptor $a_i$ can pre-prepare any value different from $v$ in view $w+k+1$.

*Inductive Step:* Again, it is sufficient to prove that any valid $Writeproof$ of view $w+k+1$ witnesses $v$. As in the Base step, we can argue that there is a set $Z$ containing at least $S_Z = S_v^3 \geq N_a - 2F - M$ non-malicious acceptors that pre-prepared and prepared $v$ in $w$ and whose *NEW-VIEW-ACK* message is part of the $WriteProof$ of view $w+k+1$. The sets $K_*$ and $P_*$ in the valid $WriteProof$ of view $w+k+1$ contain only values with associated view numbers up to $w+k$ (line 4, Fig. 8). Assume $choose(*, Writeproof)$ returns $(v', w' \leq w+k$. If $w' > w$ then $v' = v$ because, by IH a value pre-prepared by any non-malicious acceptor in view $w', w < w' \leq w+k$ can be only $v$, and by Lemma 8, in order for $choose(*, WriteProof)$ to return $v', w'$, one non-malicious acceptor must have pre-prepared $v'$ in a view higher or equal to $w'$. On the other hand, it is not possible that $w' < w$, as the presence of the messages sent by the acceptors from the set $Z$ in the $Writeproof$ guarantees that $v$ will be the candidate-3 value with an associated view number higher than or equal to $w$. Therefore, $choose(*, WriteProof)$ will always return a value with the associated view number $w' \geq w$. Finally, if $w' = w$, we can use similar reasoning as in the Base Step and conclude that there can not be more than one candidate-3 values with the associated view number $w$ in the valid $WriteProof$, nor some candidate-2 value $v' \neq v$ with associated view number $w' = w$ can be selected before candidate-3 value $v$. Hence, we conclude that the $choose(*, WriteProof)$ returns $v$.

**Lemma 13.** *If $w$ is the lowest view number in which some value $v$ is learned, then no non-malicious acceptor $a_i$ pre-prepares any value $v' \neq v$ in any view higher than $w$.*

*Proof.* Follows directly from Lemmas 7, 11 and 12.

**Lemma 14.** *(Agreement) No two different values can be learned.*

*Proof.* Follows from Lemma 13 and the fact that if some value $v'$ is learned in view $w$ some non-malicious acceptor pre-prepared $v'$ in $w$.

Lemma 14 proves *Agreement*.

To help prove liveness (i.e., the *Termination* property) and the show that DGV allows very fast (resp. fast) learning, we identify three *Weak Termination* properties of the *Locking* part of our algorithm.

- Very Fast Weak Termination (VFWT) If (a) run $r$ is synchronous, (b) a correct privileged proposer $p_k$ is the only proposer that proposes a value (for a sufficiently long time) in $r$ and (c) at most $Q$ acceptors are faulty, then every correct learner learns a value in two communication rounds.
- Fast Weak Termination (FWT) If (a) run $r$ is synchronous, (b) a correct privileged proposer $p_k$ is the only proposer that proposes a value (for a sufficiently long time) in $r$ and (c) $Q'$, where $Q < Q' \leq F$ acceptors are faulty, then every correct learner learns a value in three communication rounds.
- Eventual Weak Termination (EWT) If (a) run $r$ is eventually synchronous, (b) a correct proposer $p_k$ proposes a value at time $t$, after GST ($t > GST$), with the highest view number out of all proposals invoked up to time $t$, (c) no proposer proposes a value after $t$ with a higher view number (for a sufficiently long time) and (d) at most $F$ acceptors are faulty, then every correct learner eventually learns a value.

The notion of "sufficiently long time" in the case of VFWT, means that no proposer other than $p_k$ proposes before $N_a - Q$ correct acceptors receive the *PRE-PREPARE* message from $p_{Init}$. In the synchronous run in which up to $Q$ acceptors are faulty, this time ($\Delta_c$) is bounded and correctly estimated by every acceptor. In the case of FWT, "sufficiently long time" means that no proposer other than $p_k$ proposes before $N_a - F$ correct acceptors receive the $N_a - F$ *PREPARE* messages that correspond to *PRE-PREPARE message* sent by $p_{Init}$. In the synchronous run in which up to $F$ acceptors are faulty, this time ($\Delta_c$) is bounded and correctly estimated by every acceptor. Finally, in the case of EWT, "sufficiently long time" means that either no proposer proposes the value before $N_a - Q$ correct acceptors receive the *PRE-PREPARE* message from the $p_k$, or $N_a - F$ correct acceptors receive each $N_a - F$ *PREPARE* messages corresponding to $p_k$'s proposal.

**Lemma 15.** *The* Locking *module, from Figure8, satisfies* Very Fast Weak Termination.

*Proof.* As the run is synchronous, correct acceptors receive the *PRE-PREPARE* message from the correct privileged proposer within a known time period (i.e., $\Delta_c$). As no other proposer proposes for a sufficiently long time, i.e. until $N_a - Q$ correct acceptors receive *PRE-PREPARE* message, $N_a - Q$ correct acceptors pre-prepare leader's proposal and send a *PREPARE* message to learners. Again, as the run is synchronous, every correct learner receives $N_a - Q$ *PREPARE* messages within $2\Delta_c$ after the value was proposed, when it learns a value,. Therefore, every correct learner learns a value in two communication rounds after the correct proposer proposed a value.

**Lemma 16.** *The* Locking *module, from Figure 8, satisfies* Fast Weak Termination*.*

*Proof.* The proof is analogous to that of Lemma 15.

Using the *VFWT* property of the *Locking* part, we can prove that DGV provides very fast learning.

**Lemma 17.** *If the run is synchronous and up to $Q$ acceptors fail and the privileged proposer $p_{Init}$ ($p_0$) is correct and proposes a value (very favorable run), then DGV algorithm provides very fast learning.*

*Proof.* To prove this lemma, we assume that $p_{Init}$ proposes immediately after the initialization of the algorithm. From line 2, Figure 8, we see that when the correct $p_{Init}$ proposes, it skips the *READ* phase and directly sending the *PRE-PREPARE* message to the acceptors. As the run is synchronous, all messages sent among correct processes are delivered within a correctly estimated (by every correct process) bound on the message transmission delay ($\Delta_c$). This guarantees that no correct acceptor receives *PRE-PREPARE* message after $\Delta_c$ and that the timers at acceptors are set according to $\Delta_c$ (i.e., no acceptor times out at $t = \Delta_c$). In other words, as the $p_{Init}$ proposes immediately after initialization of the algorithm, no correct acceptor will suspect the leader at $\Delta_c$ and all correct acceptors will receive a *PRE-PREPARE* message before any other proposer proposes (with a valid view proof), i.e., no other proposer proposes for a sufficiently long time. From the *VFWT* property of the *Locking* module we conclude that all the correct learners learn a value within two communication rounds.

**Lemma 18.** *If the run is synchronous and up to $F$ acceptors fail and the privileged proposer $p_{Init}$ ($p_0$) is correct and proposes a value (very favorable run), then DGV algorithm provides fast learning.*

*Proof.* The proof is analogous to that of Lemma 17.

We proceed with few more lemmas to prove *Termination*.

**Lemma 19.** *If a valid $Writeproof$ consists only of* NEW-VIEW-ACK *messages sent by non-malicious acceptors, $choose(*, Writeproof)$ never aborts.*

*Proof.* It is sufficient to prove that if $choose(*, Writeproof)$ aborts, then the $Writeproof$ contains a *NEW-VIEW-ACK* message from at least one malicious acceptor. First, consider the case where $choose(*, Writeproof)$ aborts with $flag = true$ (in lines 13 and 25, Figure 7). We consider the following two exhaustive subcases:
Case (a): $choose()$ aborts in line 13, as there are two candidate-2 values $v'$ and $v'' \neq v'$ with the same $M + 1^{st}$ highest associated view number $w$ and the leader of the view $w$ is in the $Writeproof$. Therefore, in $Writeproof$ there are (at most) three acceptors: $a_w$, the leader of the view $w$, $a_i$, that claims it received $v'$ from $a_w$ in view $w$, and $a_j$, that claims that it received $v''$ from $a_w$ in view $w$. It is not difficult to see that at least one of these acceptors is malicious.

Case (b): $choose()$ aborts in line 25, as there is a candidate-2 values $v_2$ with the $M + 1^{st}$ highest associated view number $w$, and a candidate-2 value $v_3$ with the same associated view number $w$ and the leader of the view $w$ is in the $Writeproof$. Similarly as in the case (a), in $Writeproof$ there are (at most) three acceptors: $a_w$, the leader of the view $w$, $a_i$, that claims it received $v_2$ from $a_w$ in view $w$, and $a_j$, that claims it received $v_3$ from $a_w$ in view $w$. It is not difficult to see that at least one of these acceptors is malicious.

Consider now the case where $choose(*, Writeproof)$ function with $flag = false$ (lines 5,23 and 29 of Fig. 7. We consider the following three exhaustive subcases:

Case (a): $flag = false$ because there is more than one candidate-3 value (line 5). In this case, there are two acceptors $a_i$ and $a_j$ that claim that a set $A$ of $N_a - F$ acceptors accepted a value $v$ and a set B of $N_a - F$ acceptors accepted a value $v' \neq v$ in the same view $w$. As $N_a > 2F + M$, the sets $A$ and $B$ intersect in at least one non-malicious acceptor, so either acceptor $a_i$ or acceptor $a_j$ is malicious.

Case (b): $flag = false$ because there is a candidate-2 value $v_2$ and a candidate-3 value $v_3$, with the same associated view number, such that $v_2 \neq v_3$ and $S^3_{v_3} \leq M$ in configuration $C_1$, where *PREPARE* messages exchanged among acceptors are not authenticated (line 23). In this case, the bound on the number of acceptors is $N_a \geq 2F+M+Q+1$, where $M > Q$ (otherwise $S^3_{v_3} \geq N_a - 2F - M \geq M+1$). In the $WriteProof$, there is a set $A$ of at least $N_a - Q - M - F \geq F + 1$ acceptors that claim they accepted $v_2$. Also, there is a set $B$ of at least $N_a - M - 2F \geq Q + 1$ acceptors that claim that they received (independently of each other) $v_3$ from $N_a - F$ acceptors. Therefore, every acceptor from the set $B$ claims that some acceptor from $A$ sent $v_3$ to it. Obviously, there is at least one malicious acceptor in the set $A \cup B$.

Case (c): $flag = false$ because there is a candidate-2 value $v_2$, with cardinality $S^2_{v_2} \geq N_a - Q - M - F + 1$ and a candidate-3 value $v_3$, with cardinality $S^3_{v_3} \geq N_a - M - 2F + 1$ and $M > S^3_{v_3}$, where $v_2$ and $v_3 \neq v_2$ have the same associated view number $w$, in configuration $C_2$, in the case *PREPARE* messages exchanged among acceptors are not authenticated and the *NEW-VIEW-ACK* message from the (malicious) leader of view $w$ is not in the $WriteProof$ (line 29). In this case, the bound on the number of acceptors is $N_a \geq 2F + M + Q$, where $M - 1 > Q$ (otherwise, if $M - 1 \leq Q$, $N_a \geq 2F + 2M - 1$ and $S^3_{v_3} \geq N_a - M - 2F + 1$ yields $S^3_{v_3} \geq M$). In $Writeproof$, there is a set $A$ of at least $N_a - Q - M - F + 1 \geq F + 1$ acceptors that claim they accepted $v_2$. Also, there is a set $B$ of at least $N_a - M - 2F + 1 \geq Q + 1$ acceptors that claim that they received (independently of each other) $v_3$ from $N_a - F$ acceptors. Therefore, every acceptor from the set $B$ claims that some acceptor from $A$ sent $v_3$ to it. Obviously, there is at least one malicious acceptor in the set $A \cup B$.

For the EWT property of the *Locking* module to hold, malicious acceptors need to be prevented from sending false, but valid *NEW-VIEW-NACK* messages. To satisfy this, it is sufficient to guarantee that no acceptor can have a valid view proof of the view number that has not been proposed.[7] We call this property the *No-Creation* property of the view proofs.

**Lemma 20.** *The* Locking *module, from Figure 8, satisfies* Eventual Weak Termination*, given that the view proofs satisfy the* No-Creation *property.*

*Proof.* As (a) the run is eventually synchronous, (b) a correct proposer $p_k$ proposes at time $t$ after $GST$, with a highest view number among all proposals up to $t$, (c) no proposer other than $p_k$ proposes the value for a sufficiently long time, we conclude that $N_a - F$ correct acceptors receive the *NEW-VIEW* message from $p_k$, complete the LPO subprotocol (if necessary) and reply with the *NEW-VIEW-ACK* message to $p_k$ which eventually receives all messages from correct acceptors. By Lemma 19, the $choose(v_{p_k}, Writeproof)$ does not abort and returns $v$. Furthermore, as the *No-Creation property* of the view proofs holds, no malicious acceptor can reply with a valid *NEW-VIEW-NACK* message to $p_k$. Therefore, $p_k$ sends the *PRE-PREPARE* message containing the proposal value $v$. As no other process proposes until (a) $N_a - Q$ correct acceptors receive *PRE-PREPARE* message from $p_k$ or (b) $N_a - F$ correct acceptors receive each $N_a - F$ *PREPARE* messages, we conclude that (a) at least $N_a - Q$ *PREPARE* messages or (b) at least $N_a - F$

[7] Under the notion of *proposing*, we consider only *propose()* invocations with the valid view proofs.

*COMMIT* messages are sent to every correct learner. Note that $N_a - Q$ correct acceptors from the case (a) might not exist, as only $N_a - F \leq N_a - Q$ correct acceptors are guaranteed to exist. If there are more than $Q$ acceptor failures, $N_a - F$ *COMMIT* messages will be sent as there are at least $N_a - F$ correct acceptors which pre-prepare $p_k$'s proposal and no proposer other than $p_k$ proposes a value for a sufficiently long time. As the run is eventually synchronous and *PREPARE* messages in case (a) and *COMMIT* messages in case (b) are sent after GST, eventually every correct learner $l_j$ receives $N_a - Q$ *PREPARE* messages or $N_a - F$ *COMMIT* messages, and thus, $l_j$ learns a value if it did not learn some value before. Therefore, eventually every correct learner learns a value.

**Lemma 21.** *View proofs generated in the* Election *module of the DGV algorithm satisfy the* No-Creation *property.*

*Proof.* To prove this lemma, we show that the way the view proofs are generated (lines 13,24-26, Fig. 9) together with Lemma 2(f), guarantee that no process other than the leader of the view $w$ can generate a valid $ViewProof_w$ before the leader of $w$ received all signed *VIEW-CHANGE* messages from correct acceptors contained in $ViewProof_w$. Note that a valid $ViewProof_w$ contains *VIEW-CHANGE* messages from $\lfloor (N_a + M)/2 + 1 \rfloor$ acceptors, i.e. from at least a majority of non-malicious acceptors (Lemma 2(f)) and non-malicious acceptors send the *VIEW-CHANGE* message for a view $w$ only to the leader of the view $w$. By lines 26-27 of Figure 9 and lines 3-4 of Fig 10, a correct proposer $p_k$, leader of view $w$, will immediately propose the value upon reception of necessary *VIEW-CHANGE* messages, so no acceptor can receive the view proof for the view $w$, $ViewProof_w$, before $p_k$ proposes the value. On the other hand, a malicious proposer might not follow the algorithm. Note that, however, it is safe to make the assumption that the malicious leader of the view $w$, $p_B$, proposed the value as soon as some other (malicious) process generated the valid $ViewProof_w$. This is reasonable, as we can not distinguish the case in which $p_B$ does not invoke *propose()* from the case in which $p_B$ invokes *propose()* with a valid view proof, but $p_B$ does not send any protocol message. As $p_B$ must receive the signed *VIEW-CHANGE* messages sent by non-malicious acceptors before any other process receives them, we can safely argue that, if $p_B$ had followed the algorithm, it could have had proposed. Therefore, we say that $p_B$ proposed but did not send any protocol message. Therefore view proofs that we use in our algorithm satisfy the *No-Creation* property.

Now we prove the *Termination* property. This requires the correct learners to learn a value, if a correct proposer proposes and at most $F$ acceptors are faulty, under the assumption of the eventually synchronous system. Note also that immediately after the initialization of the algorithm, correct acceptors trigger the $SuspectTimeout$ timer. After triggering the $SuspectTimeout$, no correct acceptor will stop its timer permanently until it receives a *DECISION* message from at least one learner.

**Lemma 22.** *(Termination) If a correct proposer proposes a value, then eventually, every correct learner learns a value.*

*Proof.* Suppose there is a single, correct proposer $p_k$ that proposes at time $t$, after GST, with with a highest view number $w$ among all proposals invoked up to $t$. Let $\delta'$ be the upper bound on the time interval required for the execution of the following sequence of operations (after GST): the last acceptor sends the *VIEW-CHANGE* message necessary for $ViewProof_w$, $p_k$ checks the signatures and generates the $ViewProof_w$, $p_k$ sends *NEW-VIEW* message to acceptors, $p_k$ completes successfully the *READ* phase (including the possible LPO), $p_k$ generates the $WriteProof$ and *chooses* proposal value, $p_k$ sends *PRE-PREPARE* messages, acceptors receive the proposal, check the $Writeproof$ and proposal value, send *PREPARE* messages to all learners and acceptors and, finally, all correct acceptors *prepare* the $p_k$'s proposal. As we assume that there is an upper bound on the time required for every local computation related to authentication, $\Delta_{auth}$, and that there is an upper bound $\Delta_c$ on the message transmission delay after GST, such finite upper bound $\delta'$ exists (actually, it is sufficient that $\delta' > 7(\Delta_c + \Delta_{auth})$, as there are at most seven communication rounds in the above described sequence of rounds, some of which involve local computations related to authentication).

Suppose, by contradiction, that some correct learner never learns a value even if some correct proposer proposes. We distinguish two cases: (a) when no correct acceptor receives a *DECISION* message from any learner and (b) when some correct acceptor receives a *DECISION* message from some learner.

We first consider case (a). First, we claim that, in this case, every correct acceptor goes through an infinite number of views. Basically, we show that there will be an infinite number of new views after GST, as the system is eventually synchronous.

Suppose that there is a finite number of views and let $w$ be the highest view number among them. Due to our assumption that no correct acceptor receives a *DECISION* message, no correct acceptor stops its $SuspectTimeout$ permanently. Therefore, $SuspectTimeout$ keeps expiring and being reset at every acceptor and, consequently, every correct acceptor issues *VIEW-CHANGE* messages for an infinite number of views. As we assume that there are at least $N_a - F$ correct acceptors, $N_a - F \geq \lfloor (N_a + M)/2 + 1 \rfloor$ (Lemma 1) and as the messages among correct processes are delivered in a timely manner after GST, there will be a view number $w'$ higher than $w$ for which some correct proposer sends a *NEW-VIEW* message. Therefore, every correct acceptor will accept the *NEW-VIEW* message for the view $w'$ unless it is already in the higher view than $w'$. In any case, every correct acceptor will be in the higher view than $w$ - a contradiction. Therefore, every correct acceptor goes through an infinite number of views and the $SuspectTimeout$ grows infinitely at every correct acceptor.

Note that when an acceptor sends a *VIEW-CHANGE* message for a view $w$, it triggers a timeout of duration $InitTimeout * 2^w$, where $InitTimeout$ is the initial value of $SuspectTimeout$. The value of $InitTimeout$ is the same at every acceptor. Let $t_{delivery}$ be the time at which all the *VIEW-CHANGE* messages sent before GST that are not lost are delivered. The time $t_{delivery}$ exists as there is a finite number of *VIEW-CHANGE* messages sent before GST and as the messages that are not lost are eventually delivered. Let $t > max(GST, t_{delivery})$ be the time where $\forall a_i \in C, w_{a_i} > \lceil log_2(\delta'/InitTimeout) \rceil$, where $C$ is a set of correct acceptors. Let $w_{min} = min\{w_{a_i} | a_i \in C\}$ and $NextView_{max} = max\{NextView_{a_i} | a_i \in C\}$ at time $t$. In other words, $t$ is the time at which all correct acceptors are in the view higher or equal to $w_{min}$, where $InitTimeout * 2^{w_{min}} > \delta'$.

Let $w$ be the lowest view number higher than $NextView_{max} + 1$ in which some correct proposer $p_k$ is the leader. As there is an infinite number of view changes, all correct acceptors will send a *VIEW CHANGE* message for view $w$, at latest at $t_w = t + InitTimeout * (2^{w_{min}+1} + \ldots + 2^{w-1})$. Furthermore, no correct acceptor will send any *VIEW-CHANGE* message for any view higher than $w$ before $t_{w+1} = t + InitTimeout * 2^w$. Note that for the time $t_{w+1} - t_w$ there will be no proposer that proposes with a higher view number than the $p_k$. $P_k$ will propose at the latest right after $t_w$, when it receives $\lfloor (N_a + M)/2 + 1 \rfloor$ *VIEW-CHANGE* messages. Note that $t_{w+1} - t_w = InitTimeout * (2^{w_{min}} + \ldots + 2^1 + 2^0 + 1) > InitTimeout * 2^{w_{min}} > \delta'$ Basically, $p_k$ will be the only proposer that proposes a value for a period of time greater than $\delta'$, with a highest view number of all proposals up to $t_w$. In other words, $p_k$ will propose at $t_w > GST$, with a highest view number up to $t_w$, for a sufficiently long time. By the *Eventual Weak Termination* property of the *Locking* module, every correct learner eventually decides - a contradiction.

Consider now the case (b) where some correct acceptor receives a *DECISION* message from some learner. As a correct learner periodically sends a query to acceptors if it does not learn a value and as, after GST, all messages sent among correct processes are delivered, a correct acceptor will eventually forward a *DECISION* message to a correct learner that will thus learn a value.

As $a_i$ is correct and as after GST messages among correct processes are delivered within a bounded of time, new leader can wait for additional *NEW-VIEW-(N)ACK* message if $choose()$ function detects a malicious acceptor within the $WriteProof$. After the reception of an additional message the new leader invokes $choose()$ function on every subset of size $N_a - F$ of the set of received *NEW-VIEW-ACK* messages (e.g., in a set of $N_a - F + 1$ *NEW-VIEW-ACK* messages, there are $N_a - F + 1$ subsets of size $N_a - F$). If every $choose()$ invocation aborts with $flag = false$ on every subset of received *NEW-VIEW-ACK* messages of size $N_a - F$, the leader waits for another *NEW-VIEW-ACK* message and so on.

### 5.7 DGV variants

First, we give the DGV variants that match the lower bounds for configuration $C_2$. This is followed by discussing DGV optimization in a special case of parameter values, namely $Q = F$.

**Configuration $C_2$** As we pointed out earlier the DGV variation $DGV_{Alg.1}$ addressed a special case of configuration $C_2$, where all proposers are also acceptors and where authentication is not used for very fast learning. This variant is optimized for using in the state-machine replication in a model where there can be more than one privileged proposer that is also an acceptor, but, for a single consensus instance, there is only one privileged proposer/acceptor. However, to give a generic lower bound matching solution for configuration $C_2$ (part 2 of the theorem), we need to modify $DGV_{Alg.1}$. First, we give the general solution for configuration $C_2$ in the case authentication is not used for very fast learning. Namely we show that:

**Proposition Alg.3** There is a consensus algorithm $A$, with a single proposer $p_l$, where $p_l$ is also an acceptor, such that: in every very favorable run of $A$ every correct learner learns a value by round 2 without using authentication despite the failure of $Q$ acceptors whenever $N_a > max(2(M-1) + Q + 2F, 2M + Q + F)$. This matches the bound established by combining propositions L.4 and L.5 from Section 4.

In addition, in every favorable run of $A$, every correct learner learns a value by round 3 despite the failure of $F$ acceptors:

(a) using authentication when $N_a \leq 2F + (M-1) + min(M-1, Q)$,

(b) without using authentication when $N_a > 2F + (M-1) + min(M-1, Q)$. This matches the bound established by proposition L.6) from Section 4.

Interestingly, to prove proposition Alg.3, we simplify the $DGV_{Alg.1}$ in a way that we use very fast learning techniques only in $Initview$. Namely,

1. Acceptors do not send *PREPARE* messages to learners in any view other than $Initview$.
2. Acceptors do not modify their $K_{a_i}$ sets in a view other than $InitView$.

The rest of the algorithm stays the same (performance optimizations are possible). Practically, these small changes significantly simplify the DGV variation $DGV_{Alg.1}$ and, especially, its proof. Basically, solving potential disputes among candidate values becomes a lot easier whenever there is a candidate (namely candidate-3) value with associated view number higher than $Initview$).

Now we match the lower bound from part 2 of the theorem in the case authentication is used in very fast learning. Namely we show that:

**Proposition Alg.4** There is a consensus algorithm $A$, with a single proposer $p_l$, where $p_l$ is also an acceptor, such that: in every very favorable run of $A$ every correct learner learns a value by round

2 (using authentication) despite the failure of $Q$ acceptors whenever $N_a > 2(M-1)+Q+2F$. This matches the bound established by proposition L.4 from Section 4.

To prove proposition Alg.4, we need to introduce two additional changes to $DGV_{Alg.1}$. Namely, in addition to changes (1) and (2), i.e., not using very fast learning techniques in any view other than $Initview$, we introduce the following modifications:

- Privileged proposer $p_{Init}$ authenticates (signs) its *PRE-PREPARE* message in $Initview$.
- We simplify and somewhat modify the *choose*() function.

Note that for simplicity, we give a variant of DGV that satisfies proposition Alg.4, but does not consider fast learning. The DGV Alg.4 variant that allows fast learning, both with and without using authenticated (very) fast learning messages (with respect to the number of available acceptors) can be obtained by merging the original Alg.1 *choose()* function we gave in Section 5.4 in Figure 7 with the variant of *choose()* function we give below.

The *choose*() function for Alg.4 variant of DGV is given in Figure 11. Note that this special variant of DGV assumes $N_a > max(2F+M, 2(M-1)+F+2Q)$. This special case is interesting only in the cases where $Q = 0, 1$, as for $Q \geq 2$, $2(M-1)+F+2Q > 2M+F+Q$ and, according to part 2 of the lower bound theorem, DGV variant that does not use authentication for very fast learning (i.e., Alg.3) is feasible.

---

**choose**$(v, WriteProof)$ **returns**$(v, view)$ **is** {
  $view_2, view_3 := -1$; $v_2, v_3 := nil$

  **sort** all (if any) candidate-3 values by their associated view no.; let $w_3$ be the highest among those view no.
  **if** $\exists$ a candidate-3 value $v_3'$ associated with $w_3$ **then** $v_3 := v_3'$; $view_3 := w_3$ **endif**

  **if** there is a single candidate-2 value $v'$ **then** $v_2 := v'$;
  **elseif** there are two candidate-2 values $v'$ and $v''$ **then**
    **if** *NEW-VIEW-ACK* sent by the privileged proposer $p_{Init}$ is in $Writeproof$ **then abort**
    **elseif** $S^2_{v'} \geq N_a - Q - F - M + 1$ **then** $v_2 := v'$ **elseif** $S^2_{v''} \geq N_a - Q - F - M + 1$ **then** $v_2 := v''$
    **endif**
  **endif**
  **if** $v_2 \neq nil$ **then** $view_2 := Initview$ **endif**

  **if** $view_2 > view_3$ **then return**$(v_2, view_2)$ **elseif** $view_3 > view_2$ **return**$(v_3, view_3)$ **else return**$(v, \perp)$ **endif**

---

**Fig. 11.** DGV-Alg.4: Configuration $C_2$, matching the lower bound when authentication is used for very fast learning - *choose()* function

We sketch the correctness proof for the Alg.4 variant of DGV. If the value was learned in $Initview$ it will be reported by $N_a-Q-M-F$ acceptors in any $WriteProof$. First, as the inequality $N_a - Q - M - F \geq M + 1$ does not hold in this case, we need to show that $N_a - Q - M - F > 0$. From $N_a > max(2F+M, 2(M-1)+F+2Q)$, we have $N_a - Q - M - F > max(F-Q, M+Q-2)$. As $F \geq Q$, we conclude $N_a - Q - M - F > 0$. Therefore, if a value was learned in $Initview$ it will certainly be a candidate-2 value in every $Writeproof$ as at least $N_a - Q - M - F$ acceptors will report it and non-malicious acceptors do not change their $K_{a_i}$ set in any view $w > Initview$. If the value $v$ is learned in $w > Initview$ we can use the similar reasoning as in Lemma 12, Section 5.6 to conclude that every $choose(*, Writeproof)$ in any view $w' > w$ must return $v$. It is also not difficult to see that if the value is learned in view $w > Initview$ it was accepted by at least one non-malicious acceptor in $w$. However, this is not true for $Initview$.

Therefore, we show that if $v$ was learned in $Initview$ non-malicious acceptors accept only $v$ in every $w > Initview$. The proof uses induction on view numbers, but we prove here only the Base Step. If $v$ is the only candidate-2 value in $Writeproof$ of $Initview + 1$ than $choose(*, Writeproof)$ returns $v$. If there is another candidate-2 value, obviously privileged proposer $p_{Init}$ is malicious and the valid $Writeproof$ does not contain a message sent by $p_{Init}$. Therefore, in this case, in valid $Writeproof$ $S_v^2 \geq N_a - Q - M - F + 1$ and as there cannot be two candidate-2 values with cardinalities of at least $N_a - Q - M - F + 1$ (as this would contradict $N_a > 2(M-1) + F + 2Q$), $choose(*, Writeproof)$ returns $v$. Therefore, *Agreement* cannot be violated.

One may argue that *Validity* can be violated as a set of malicious acceptors can make up a (single) candidate-2 value from a "thin" air. However, this is not the case. To see this, consider the case where $p_{Init}$ is not malicious. In this case, as $p_{Init}$ authenticates the *PRE-PREPARE* message, malicious acceptors cannot forge this signature and, therefore, cannot make up a candidate-2 value. On the other hand if $p_{Init}$ is malicious, any value that malicious acceptors can make up, $p_{Init}$ could have had proposed, without non-malicious acceptors distinguishing these cases.

**Q=F** When $Q = F$, the fast learning does not provide any additional guarantees with respect to the very fast learning. Therefore, in every *WRITE* phase of any DGV variation *PREPARE* messages exchanged among acceptors in $Initview$ are not necessary. In the case of DGV variants Alg.1 and Alg.2, this extends to *WRITE* phase in every view. The algorithm becomes much simpler as the *WRITE* phase of the *Locking* module consists always of only 2 communication rounds, regardless of the other algorithm parameters, namely $M$ and $F$.


## Acknowledgements

We are very grateful to Lorenzo Alvisi, Gildas Avoine, Leslie Lamport, Jean-Philippe Martin and Piotr Zielinski for their very helpful comments.


## References


1. Marcos Aguilera, Carole Delporte-Gallet, Hugues Fauconnier, and Sam Toueg. Consensus with Byzantine failures and little system synchrony. Technical Report 2004-008, Université Denis Diderot, Laboratoire d'Informatique Algorithmique, Fondements et Applications (LIAFA), September 2004.
2. Bowen Alpern and Fred B. Schneider. Defining liveness. *Information Processing Letters*, 21:181–185, 1985.
3. Romain Boichat, Partha Dutta, Svend Frölund, and Rachid Guerraoui. Deconstructing paxos. *SIGACT News in Distributed Computing*, 34(1):47–67, 2003.
4. Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *Journal of the ACM*, 32(4):824–840, October 1985.
5. Miguel Castro and Barbara Liskov. Authenticated Byzantine fault tolerance without public-key cryptography. Technical Report MIT/LCS/TM-589, MIT Laboratory for Computer Science, 1999.
6. Miguel Castro and Basrbara Liskov. Practical Byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, New Orleans, USA, February 1999.
7. Miguel Correia, Nuno Ferreira Neves, L. C. Lung, and Paulo Veríssimo. Low complexity Byzantine-resilient consensus. DI/FCUL TR 03–25, Department of Informatics, University of Lisbon, August 2003.
8. D. Dolev and H.R. Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.
9. Danny Dolev, Ruediger Reischuk, and H. Raymond Strong. Early stopping in Byzantine agreement. *Journal of the ACM*, 37(4):720–741, 1990.
10. Assia Doudou, Benoît Garbinato, and Rachid Guerraoui. Encapsulating failure detection: from crash to Byzantine failures. In *Proceedings of the Int. Conference on Reliable Software Technologies*, Vienna, May 2002.
11. Partha Dutta and Rachid Guerraoui. The inherent price of indulgence. In *Proceedings of the twenty-first annual symposium on Principles of distributed computing*, pages 88–97. ACM Press, 2002.
12. Cynthia Dwork, Nancy Lynch, and Larry Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, April 1988.

13. M. Fischer and N. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183–186, 1982.
14. Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.
15. Roy Friedman, Achour Mostefaoui, and Michel Raynal. Simple and efficient oracle-based consensus protocols for asynchronous Byzantine systems. In *23rd IEEE International Symposium on Reliable Distributed Systems*, pages 228–237, Florianpolis, Brazil, October 2004. IEEE Computer Society Press.
16. Eli Gafni. Round-by-round fault detectors (extended abstract): unifying synchrony and asynchrony. In *Proceedings of the seventeenth annual ACM symposium on Principles of distributed computing*, pages 143–152. ACM Press, 1998.
17. Idit Keidar and Sergio Rajsbaum. A simple proof of the uniform consensus synchronous lower bound. *Information Processing Letters*, 85(1):47–52, 2003.
18. Klaus Kursawe. Optimistic Byzantine agreement. In *Proceedings of the 21st IEEE Symposium on Reliable Distributed Systems*, page 352. IEEE Computer Society, 2002.
19. Leslie Lamport. Time, clocks and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558–565, July 1978.
20. Leslie Lamport. The part-time parliament. *ACM Transactions on Computer Systems*, 16(2):133–169, 1998.
21. Leslie Lamport. Lower bounds for asynchronous consensus. In *Future Directions in Distributed Computing*, Springer Verlag (LNCS), pages 22–23, 2003.
22. Leslie Lamport. Lower bounds for asynchronous consensus. Technical Report MSR-TR-2004-72, Microsoft Research, 2004.
23. Leslie Lamport. Private communication. November 2004.
24. Dahlia Malkhi and Michael Reiter. A high-throughput secure reliable multicast protocol. *Journal of Computer Security*, 5(2):113–127, 1997.
25. J-P. Martin and L. Alvisi. Fast Byzantine paxos. Technical Report TR-04-07, The University of Texas at Austin, Department of Computer Sciences, 2004.
26. M. Pease, R. Shostak, and L. Lamport. Reaching agreements in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.
27. Michael K. Reiter. Secure agreement protocols: reliable and atomic group multicast in rampart. In *Proceedings of the 2nd ACM Conference on Computer and communications security*, pages 68–80. ACM Press, 1994.
28. Michael K. Reiter and Kenneth P. Birman. How to securely replicate services. *ACM Transactions on Programming Languages and Systems*, 16(3):986–1009, 1994.
29. Ronald L. Rivest. The MD5 message-digest algorithm. *Internet RFC-1321*, 1985.
30. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.
31. Fred B. Schneider. Implementing fault-tolerant services using the state machine approach: A tutorial. *ACM Computing Surveys*, 22(4):299–319, December 1990.
32. Gene Tsudik. Message authentication with one-way hash functions. In *Proceedings of the 11th Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 2055–2059, 1992.

## Appendix: Removing authentication from *DECISION* messages

In Section 5.3, we argued that the signing of the *DECISION* message that learners send upon learning a value can be avoided. A *DECISION* message permanently stops the *SuspectTimeout*, and, thus, the *Election* at an acceptor module. The idea is to use the generalized consistent broadcast (GCB) subroutine [1, 4]. GCB has three primitives: (1) *gcBcast*$(v)$, (2) *gcDeliver*$(v)$ and (3) *gcWeakDeliver*$(v)$. GCB has the following properties:

- (Validity) If a correct learner *gcBcasts* $v$, then all correct acceptors *gcDeliver* $v$.
- (No Creation) If some learner *gcWeakDelivers* $v$, then $v$ was *gcBcasted* by some learner.
- (Relay) If a correct acceptor *gcDelivers* $v$, then all correct acceptors eventually *gcDeliver* $v$.
- (Priority) If a correct acceptor *gcDelivers* $v$, then all correct learners eventually *gcWeakDeliver* $v$.

Implementation of the general consistent broadcast is given in Figure 12. It is slightly modified (generalized with respect to $M$ and $F$ and configuration where learners and acceptors are distinct) from the implementation of [1]. Every message sent within the GCB subroutine is retransmitted periodically, to circumvent our assumption on unreliable channels, i.e., to implement virtual reliable channels.

---

**at every learner** $l_j$:
**to** *gcBcast*$(v)$:
    **send** $\langle INIT, v\rangle$ to all acceptors

**upon** reception of $\langle READY, v\rangle$ from $M+1$ different acceptors
    **if** no value already *gcWeakDelivered* **then** $gcWeakDeliver(v)$

**at every acceptor** $a_j$:
**upon** reception of $\langle INIT, v\rangle$ from some learner
    **if** no $\langle ECHO, *\rangle$ message already sent **then send** $\langle ECHO, v\rangle$ to all acceptors

**upon** reception of $\langle ECHO, v\rangle$ from $\lfloor (N_a + M)/2 + 1\rfloor$ different acceptors
    **if** no $\langle ECHO, *\rangle$ message already sent **then send** $\langle ECHO, v\rangle$ to all acceptors
    **if** no $\langle READY, *\rangle$ message already sent **then send** $\langle READY, v\rangle$ to all acceptors and learners

**upon** reception of $\langle READY, v\rangle$ from $M+1$ different acceptors
    **if** no $\langle ECHO, *\rangle$ message already sent **then send** $\langle ECHO, v\rangle$ to all acceptors
    **if** no $\langle READY, *\rangle$ message already sent **then send** $\langle READY, v\rangle$ to all acceptors and learners

**upon** reception of $\langle READY, v\rangle$ from $N_a - F$ different acceptors
    **if** no value already *gcDelivered* **then** *gcDeliver*$(v)$

---

**Fig. 12.** Implementation of a Generalized Consistent Broadcast

Having the implementation of GCB, *Election* module is modified to have every learner $l_j$ *gcBcast* a value $v$, once $l_j$ learns $v$ and, furthermore to have $l_j$ learn $v$, once that $l_j$ *gcWeakDelivered* $v$, unless $l_j$ already learned a value. Furthermore, acceptor $a_j$ permanently stops its *SuspectTimeout* once it *gcDelivers* some value.

Proof of correctness of GCB is similar to the proof of consistent unique broadcast given in [1], using Lemmas 1 and 2 from Section 5.6 to prove intersection of subsets of acceptors. We omit the complete proof.