Vol.2010-AL-128 No.2
2010/1/26

Adaptive One-Step Byzantine Consensus

Nazreen Banu,[†1] Taisuke Izumi[†1] and Koichi Wada [†1]

It is known that Byzantine consensus algorithms guarantee one-step decision only in favorable situations (e.g. when all processes propose the same value) and no one-step algorithm can support two-step decision. In this paper, we present a novel one-step Byzantine algorithm *DEX* based on the condition-based approach that circumvents the above impossibilities. Algorithm *DEX* is adaptive in the sense that it is sensitive to only actual number of failures, and hence achieves fast termination for large number of inputs when less number of processes are faulty. In addition, it also has double-expedition property, so that it allows two-step decision in addition to one-step decision by introducing two condition-based mechanisms running concurrently. To the best of our knowledge, double-expedition property is a new concept introduced by this paper and *DEX* is the first algorithm having such a feature. Even though *DEX* takes four steps at worst in well-behaved runs while existing algorithms takes only three, it provides fast termination for large number inputs, which makes us to expect that our algorithm works faster in average in practical situations.

1. Introduction

1.1 Background

The *consensus* problem plays an important role in the construction of fault-tolerant distributed systems. In the consensus problem, each process proposes a value, and all non-faulty processes have to agree on a common value which is one of the proposed values. Several practical agreement problems (such as atomic broadcast, view synchrony, state-machine replication, etc.) can be implemented using a solution to the consensus problem, and thus the consensus algorithm is an important building block in designing distributed systems.

The consensus problem has been studied with various failure models and different synchrony assumptions. This paper is concerned with Byzantine consensus in asynchronous distributed systems, where faulty processes can behave in arbitrary way, and there are no assumptions on relative speed of processes nor timely delivery of messages.

To reach a single decision value, consensus protocols need to exchange messages. Each message exchange constitutes a communication step. The number of communication steps taken for reaching agreement is an important measure to evaluate the efficiency of consensus algorithms. In previous works, it has been proved that any consensus algorithm requires at least two communication steps for decision even in failure-free executions[1)]. This lower bound often becomes a dominant part of the performance overhead imposed to consensus-based applications. However, this fact does not implies that the two-step lower bound is always incurred for any inputs (an input to consensus algorithms is defined as an n-tuple consisting of all proposed values). Actually, for example, it does not hold in the case where all processes propose the same value. Furthermore, in typical runs of consensus-based applications, the consensus algorithm often receives such "good" inputs. Let us take an example of the state-machine replication: In the state-machine replication approach, the consensus algorithm is used to agree with the processing order of update requests. If some update request arises, it is broadcast to all replicated servers. The server receiving the update request proposes a received request as the candidate it will handle next. Then, without contention (that is, there is no other concurrent update request), all servers propose the same request. Practically, it is not so often the case that two or more requests concurrently updates the same data object.

This observation raises up the interest of one-step decision in the case of good inputs. The attempts to circumvent two-step lower bound is initiated by Brasileiro et al.[2)]. It proposes a general framework to convert any crash-tolerant algorithm into the one that solves the consensus for any input, and especially terminates in one step when all processes propose the same value. In following literatures[4),5)], the notion of one-step decision is considered in combination with other schemes such as randomization and failure detectors.

An interesting aspect of one-step decision schemes is to characterize the situations where one-step decision is possible. The first investigation from that aspect is considered by Mostefaoui et.al.[7)], which applies the *condition-based approach* for obtaining a good one-step decision scheme. In general, the condition-based

†1 Nagoya Institute of Technology

approach defines a set of inputs, called *condition*, for which the algorithm guarantees a certain kind of good property. The first result with this approach[7] gives a sufficient class of conditions such that we can construct the algorithm guaranteeing one-step decision for any input in the condition. This result is extended by Izumi and Masuzawa[6]. It gives the complete characterization of conditions that makes one-step decision possible.

While all of the above results are considered on crash-failure models, a recent work[3] devises one-step consensus algorithms on Byzantine failure models. They show two variants of one-step Byzantine consensus problem, weak and strong ones. The weak one-step Byzantine consensus guarantees one-step decision in any situation where all processes propose the same value and no process is faulty, but the strong must guarantee it in the situation only with a common proposed value, regardless of the number of faulty processes. They propose two algorithms for those variants, and also prove the assumption $n > 5t$ and $n > 7t$ is necessary for weak and strong one-step Byzantine consensus respectively, where n is the number of processes and t is the maximum number of faulty processes.

1.2 Our Contribution

As seen in the above, the research challenge centered in one-step consensus is to enhance and clarify the situations making the system reach one-step decision. With the same research direction, this paper also explores Byzantine consensus algorithms with better one-step decision schemes. In particular, we focus on two features for one-step decision schemes shown as follows:

Adaptive Condition-Based Approach Most of one-step decision scheme is designed so that it never violates the agreement even if the number of faulty processes is at the maximum. However, such a design works as a pessimistic approach when actual number of faulty processes are small, which is the usual case in real systems. An approach to circumvent this drawback is the use of *adaptive condition-based approach*. Informally, the adaptive condition-based approach handles the condition that dynamically changes according to the actual number of faulty processes (typically, less faults allows the condition with larger number of inputs). In the context of one-step consensus, it means that the algorithm can terminate in one-step for large number of inputs when less processes are faulty. The notion of the adaptive condition-based approach is first introduced by Izumi and Masuzawa[8], and applied to one-step consensus problem in crash-failure models[6]. However, there is no results to apply it in Byzantine-failure models.

Double Expedition of One-Step Consensus One of serious drawback incurred by one-step decision schemes is the impossibility of zero-degradation[4),9)]. Informally, the zero-degradation is one of important features of consensus algorithms based on failure detectors, which always guarantees the best complexity (i.e., two steps) in *stable* runs where the failure detector does not mistake and its output is stable. The intuition of this result is that, to achieve one-step decision, any algorithm must sacrifice the decision at the end of the second step. It is also shown that achieving both one-step decision and zero-degradation needs more stronger assumption about failure detections such as eventually perfect failure detectors $\diamond P$. However, similar with the impossibility of one-step decision, this result does not necessarily imply the impossibility of two-step decision for any inputs. Thus, it yields an interest to realize a *doubly-expedited* consensus algorithm, which equips a "conditional" two-step decision scheme combined with one-step decision.

The contribution of this paper is to propose a doubly-expedited Byzantine consensus algorithm based on the adaptive condition-based approach. The distinguished features of the proposed algorithm can be summarized as follows:

- In our construction, we show a generic framework of the algorithm based on the notion of the adaptive condition-based approach. Generally, the adaptiveness property in the condition-based approach can be characterized by a *condition sequence*, which is defined as a sequence of t conditions such that the k-th condition is valid when the actual number of correct processes is k. To handle double-expedition property in adaptive manner, the framework is instantiated by a pair of condition sequences, each of which corresponds to the situations of one-step and two-step decision respectively. We also show a sufficient criteria, say *legality*, of condition-sequence pair for which doubly-expedited algorithms can be instantiated.
- Two examples of legal condition-sequence pairs are proposed, called

frequency-based pair and *privileged-value-based pair*. They have distinct advantages in the sense that the expedited situations corresponding to each pair is complementary. Interestingly, the algorithm instantiated by the frequency-based pair takes more chances to decide in one or two steps, compared to existing one-step Byzantine consensus algorithms.

- One drawback of the proposed framework is that it trades the decision scheme at third step for double-expedition property. This drawback may causes a performance degradation of consensus-based applications if we consider pessimistic runs (that is, the given input is out of the condition). However, standing on its optimistic counterpart, we make more inputs belong to the conditions, which implies that the algorithm decides in two steps for many cases, and totally achieves better performance in average.

To the best of our knowledge, the property of double expedition is the concept newly introduced in this paper. Hence, this paper is the first result showing the feasibility of taking both one- and two-step decision schemes simultaneously with no help of additional stronger assumptions.

1.3 Roadmap

The paper is organized as follows: Section 2 presents the system model, the definitions of Byzantine consensus problem, and other necessary formalizations. Section 3 provides the legality criteria for doubly-expedited consensus and its examples. In Section 4, we present our generic framework of doubly-expedited one-step consensus algorithms. Section 5 provides our final remarks.

2. Preliminaries

2.1 System model

An asynchronous distributed system consists of n processes $\prod = \{p_1, p_2, ...p_n\}$. Each pair of processes can communicate with each other by sending messages over a reliable link where neither message loss, creation nor corruption occurs. Since we assume asynchronous systems, there is no assumption on relative speed of processes or message delay.

As we consider the Byzantine failure model, a faulty process can behave arbitrarily, which means that even it is allowed not to follow the deployed algorithm. A process that is not faulty is said to be *correct*. We assume the upper bound on the number of faulty processes, which is denoted by t. Every process knows the value of t in advance. Throughout this paper, we assume $5t < n$, which is the necessary assumption to make one-step decision possible. We also denote by f, the actual number of failures during executions. Notice that each process cannot be aware of the value of f.

2.2 Byzantine Consensus and Underlying Consensus Primitive

The Byzantine consensus problem has been informally stated in the introduction: Each process proposes a value, and all correct processes have to decide a common value which is proposed by at least one process. Formally, the Byzantine consensus is defined by the following requirements.

Termination Each correct process eventually decides a value.

Agreement If two correct processes decide, they must decide the same value.

Unanimity If all correct processes propose the same value v, then no correct process decides the value different from v.

In general, Byzantine consensus is not solvable in the asynchronous system with no additional assumption. Thus, we need some assumptions to guarantee correct termination for arbitrary inputs. While many kinds of assumptions are considered in past literatures, our research objective is finding the feasibility of one-step decision, and thus we simply assume an abstraction of them. More precisely, the system is assumed to be equipped with the *underlying consensus primitive*. This primitive ensures agreement, termination, and unanimity, but has no guarantees about its running time.

2.3 Condition-Based Approach

In the condition-based approach, an input vector is a n dimensional vector, whose i-th entry contains the value proposed by process p_i. A condition defined for n processes is a subset of all possible input vectors. Adaptiveness in the condition-based approach is the property that a condition can change dynamically according to the actual number of faulty processes. Thus, it is defined by a *condition sequence* $(C_0, C_1, ...C_k...C_t)$ satisfying $C_k \supseteq C_{k+1}$ for any $k(0 \leq k \leq t-1)$, where k-th condition corresponds to the set of input vectors that is valid when actual number of faults is equal to k.

2.4 Doubly-Expedited Consensus

In this subsection, we introduce a novel feature of consensus algorithms, called

double-expedition property. In the execution of doubly-expedited algorithms, each process has two chances for faster decision by running one-step and two-step decision schemes concurrently. Since both decision schemes guarantees faster decision only for good inputs, we can characterize their property by a pair of condition sequences: Throughout this paper, we introduce a pair of condition sequences $(S^1, S^2) = ((C^1_0, C^1_1, \cdots C^1_k, \cdots C^1_t), (C^2_0, C^2_1, \cdots C^2_k, \cdots C^2_t))$, where S^1 and S^2 correspond to the condition sequences identifying the situations that guarantee one-step and two-step decisions respectively. For example, consider the input vector I such that $I \notin C^1_k$, $I \in C^1_{k-1}$ and $I \in C^2_k$ hold. Then, if I is given to the consensus algorithm and the number of faulty processes is less than k, all processes decide in one step because $I \in C^1_{k-1}$. If (exactly) k processes are faulty, one-step decision is no more guaranteed, but all processes necessarily decide within two steps because of $I \in C^2_k$.

3. Legality for Double Expedition

It is clear that we cannot design the doubly-expedited consensus algorithm for any pair of condition sequences. In this section, we propose a sufficient criteria such that we can construct the doubly-expedited algorithm characterized by the condition-sequence pair satisfying it. It is also shown that two examples of condition-sequence pairs satisfying the criteria.

3.1 Notations

Let $\mathcal{V}$ be the domain of possible proposed values. We introduce the default value $\perp$ not in $\mathcal{V}$. Letting I be an input vector in $\mathcal{V}^n$, we define a *view* J of I to be a vector in $(\mathcal{V} \cup \{\perp\})^n$ which is obtained by replacing at most t entries in I by $\perp$. As well as, we define $\perp^n$ be a vector with all entries equal to $\perp$. The number of occurrences of value v in a view J is denoted by $\#_v(J)$. For two views J_1 and J_2, let $dist(J_1, J_2)$ be the Hamming distance between J_1 and J_2 (that is, $dist(J_1, J_2) = |\{k \in \{1, 2, ..n\} | J_1[k] \neq J_2[k]\}|$). As well as $\mathcal{V}^n_k$ denotes the set of all views where $\perp$ values appears at most k times. The number of non-default values in J is denoted by $|J|$. We define $\mathcal{I}_k$ as the set of all possible views J such that $dist(J, I) \leq k$.

3.2 Legality Criteria

Given a pair of condition sequences $(S^1_, S^2)$, we consider two predicates $P1, P2$: $\mathcal{V}^n_t \to \{\text{True}, \text{False}\}^{\star 1}$ and a function $F : \mathcal{V}^n_t \to \mathcal{V}$. Then, (S^1, S^2) is said to be *legal* if we can define $P1$, $P2$ and F satisfying the following five properties:

- LT1: $\forall J \in \mathcal{V}^n_t : \exists I : I \in C^1_k \wedge dist(J, I) \leq k \Rightarrow P1(J)$.
- LT2 :$\forall J \in \mathcal{V}^n_t : \exists I : I \in C^2_k \wedge dist(J, I) \leq k \Rightarrow P2(J)$.
- LA3 : $\forall J, J' \in \mathcal{I}_t : P1(J) \wedge dist(J, J') \leq t + \#_\perp(J) + \#_\perp(J') \Rightarrow F(J) = F(J')$.
- LA4 : $\forall J, J' \in \mathcal{I}_t : P2(J) \wedge dist(J, J') \leq \#_\perp(J) + \#_\perp(J') \Rightarrow F(J) = F(J')$.
- LV5 : $\forall J \in \mathcal{V}^n_t \Rightarrow F(J) =$ (the most common non $\perp$ value in J) $\vee F(J) = a : \#_a(J) > t$.

These properties are used to enforce the basic requirements of the doubly-expedited Byzantine consensus. Informally, $P1$ and $P2$ are the predicates to test whether the current view contains sufficient information to decide in one or two step(s) respectively, and F is the function to obtain the decision from the current view. Thus, the first property *LT1* is for imposing one-step termination. The predicate *P1* must allow each correct process to decide in one step if its own view has the possibility to come from an input vector included in the condition. Similarly, the property *LT2* corresponds to two-step decision. The property *LA3* (or *LA4*) implies the agreement between one-step (or two-step) decision and others. The last property *LV5* is the one to guarantee unanimity.

3.3 Example 1: Construction by the Frequency-Based Condition

This subsection introduces a legal condition-sequence pair that is based on frequency-based conditions and prove its legality. Let $1st(J)$ be a non $\perp$ value that appears most often in a vector J. If two or more values appear most often in J, then the largest one is selected. Let $\hat{J}$ be the vector obtained by replacing $1st(J)$ from J by $\perp$, and we define $2nd(J) = 1st(\hat{J})$. That is, $2nd(J)$ is the second most frequent value in J. The frequency-based condition C^{freq}_d is defined as follows:

$$C^{freq}_d = \{I \in \mathcal{V}^n | \#_{1st(I)}(I) - \#_{2nd(I)}(I) > d\}$$

It is known that C^{freq}_d belongs to *d-legal* conditions[7], which are necessary and sufficient to solve the consensus in failure prone asynchronous systems, where at most d processes can crash.

⋆1 In what follows $P1(J) =$ true is abbreviated as $P1(J)$

Using this condition, we can construct a legal condition-sequence pair $(S^{freq1}, S^{freq2}) = ((C_0^{freq1}, C_1^{freq1}, ...C_k^{freq1}...C_t^{freq1}), (C_0^{freq2}, C_1^{freq2} ...C_k^{freq2} ...C_t^{freq2}))$ with the associated parameters $P1^{freq}$, $P2^{freq}$ and F^{freq} as follows:

$$C_k^{freq1} = C_{4t+2k}^{freq}$$
$$C_k^{freq2} = C_{2t+2k}^{freq}$$

- $P1^{freq}(J) \equiv \#_{1st(J)}(J) - \#_{2nd(J)}(J) > 4t$.
- $P2^{freq}(J) \equiv \#_{1st(J)}(J) - \#_{2nd(J)}(J) > 2t$.
- $F^{freq}(J) = 1st(J)$.

Notice that, since there are at most t Byzantine processes, the stronger assumption $n > 7t$ is required to construct (S^{freq1}, S^{freq2}).

Theorem 1 Let $n > 7t$. The condition-sequence pair (S^{freq1}, S^{freq2}) is legal.

Proof *LT1:* We show that $\#_{1st(I)}(I) - \#_{2nd(I)}(I) > 4t + 2k \wedge dist(J, I) \leq k \Rightarrow \#_{1st(J)}(J) - \#_{2nd(J)}(J) > 4t$.

Assume I satisfies $\#_{1st(I)}(I) - \#_{2nd(I)}(I) > 4t + 2k$. Since $dist(J, I) \leq k$, $\#_{1st(I)}(J) \geq \#_{1st(I)}(I) - k$. Also, for any value $x \neq 1st(I)$, $\#_x(J) \leq \#_x(I) + k$. Since $2nd(I)$ is the second most frequent value in I, $\#_x(J) \leq \#_{2nd(I)}(I) + k$. Hence, $\#_{1st(I)}(J) - \#_x(J) > \#_{1st(I)}(I) - k - \#_{2nd(I)}(I) - k$. Thus, we get $\#_{1st(I)}(J) - \#_x(J) > 4t$. It implies that $1st(I) = 1st(J)$, and thus we obtain $\#_{1st(J)}(J) - \#_{2nd(J)}(J) > 4t$.

LT2: We show that $\#_{1st(I)}(I) - \#_{2nd(I)}(I) > 2t + 2k \wedge dist(J, I) \leq k \Rightarrow \#_{1st(J)}(J) - \#_{2nd(J)}(J) > 2t$.

The proof is exactly the same as the proof *LT1* with only replacing $4t$ by $2t$.

LA3: Consider $J, J' \in \mathcal{I}_t$. We show that if $P1^{freq}(J) \wedge dist(J, J') \leq t + \#_\perp(J) + \#_\perp(J')$, $1st(J) = 1st(J')$ holds.

Since $P1^{freq}(J)$, we have $\#_{1st(J)}(J) - \#_{2nd(J)}(J) > 4t$. Let x be any value such that $x \neq 1st(J)$. Since $dist(J, J') \leq t + \#_\perp(J) + \#_\perp(J')$, J' can contain at most t entries with the value x, which are occupied by the value $1st(J)$ in J (due to Byzantine). In addition, J' can contain $\#_\perp(J')$ entries with the default value, which are also occupied by $1st(J)$ in J (due to asynchrony) and at most $\#_\perp(J)$ entries of J' can contain the value x, which are occupied by $\perp$ in J. Hence, $\#_{1st(J)}(J') \geq \#_{1st(J)}(J) - t - \#_\perp(J')$ and $\#_x(J') \leq \#_x(J) + t + \#_\perp(J)$. Since $2nd(J)$ is the most frequent value in J except for $1st(J)$, $\#_x(J') \leq \#_{2nd(J)}(J) + t + \#_\perp(J)$. Hence, $\#_{1st(J)}(J') - \#_x(J') > \#_{1st(J)}(J) - t - \#_\perp(J') - \#_{2nd(J)}(J) - t - \#_\perp(J)$. Since $J, J' \in \mathcal{I}_t$, $\#_\perp(J) \leq t$ and $\#_\perp(J') \leq t$. Consequently, we obtain $\#_{1st(J)}(J') - \#_x(J') > 0$. It implies that $1st(J') = 1st(J)$.

LA4: Consider $J, J' \in \mathcal{I}_t$. It suffices to show that if $P2^{freq}(J) \wedge dist(J, J') \leq \#_\perp(J) + \#_\perp(J')$, $1st(J) = 1st(J')$ holds.

Since $P2^{freq}(J)$, we have $\#_{1st(J)}(J) - \#_{2nd(J)}(J) > 2t$. Let x be any value such that $x \neq 1st(J)$. Since $dist(J, J') \leq \#_\perp(J) + \#_\perp(J')$, at most $\#_\perp(J)$ entries of J' can contain the value x, which are occupied by $\perp$ in J. In addition, at most $\#_\perp(J')$

entries of J' can contain $\perp$, which are occupied by $1st(J)$ in J (due to asynchrony). As a result, $\#_{1st(J)}(J') \geq \#_{1st(J)}(J) - \#_\perp(J')$ and $\#_x(J') \leq \#_x(J) + \#_\perp(J)$. Since $2nd(J)$ is the most frequent value in J except $1st(J)$, $\#_x(J') \leq \#_{2nd(J)}(J) + \#_\perp(J)$. Therefore, $\#_{1st(J)}(J') - \#_x(J') > \#_{1st(J)}(J) - \#_\perp(J') - \#_{2nd(J)}(J) - \#_\perp(J)$. Since $J, J' \in \mathcal{I}_t$, $\#_\perp(J) \leq t$ and $\#_\perp(J') \leq t$. Hence, we obtain $\#_{1st(J)}(J') - \#_x(J') > 0$. It implies that $1st(J') = 1st(J)$.

LV5: This property is trivially satisfied since $F^{freq}(J) =$ the most frequent non $\perp$ value in J. □

3.4 Example 2: Construction by the Privileged-Value-Based Condition

In this subsection, we present another legal condition-sequence pair (S^{prv1}, S^{prv2}) constructed from privileged-value-based conditions, and prove its legality. In some practical agreement problems such as atomic commitment, a single value (i.e., Commit) is often proposed by most of the processes. The previous results[2)] have showed that, if this value is assigned some privilege, it is possible to expedite the decision. Let us assume that there is a value (say m) that is privileged among the set of all proposal values. Each process knows the value m a priori. Then, the privileged-based condition $C_d^{prv(m)}$ can be defined as

follows:

$$C_d^{prv(m)} = \{I \in \mathcal{V}^n | \#_m(I) > d\}$$

Note that $C_d^{prv(m)}$ also belongs to *d-legal* conditions[7], which are necessary and sufficient to solve the consensus in failure prone asynchronous systems where at most d processes can crash.

Using this condition, we can construct the privileged-value-based condition-sequence pair (S^{prv1}, S^{prv2})= $((C_0^{prv1}, C_1^{prv1}, ...C_k^{prv1}...C_t^{prv1}), (C_0^{prv2}, C_1^{prv2}...C_k^{prv2}$ $...C_t^{prv2}))$ with the associated parameters $P1^{prv}$, $P2^{prv}$ and F^{prv} as follows:

$$C_k^{prv1} = C_{3t+k}^{prv(m)}$$
$$C_k^{prv2} = C_{2t+k}^{prv(m)}$$

- $P1^{prv}(J) \equiv \#_m(J) > 3t$.
- $P2^{prv}(J) \equiv \#_m(J) > 2t$.

$$F^{prv}(J) \equiv \begin{cases} m & \text{if } \#_m(J) > t \\ \text{the most freq.val.in } J & \text{otherwise} \end{cases}$$

Notice that, since there are at most t Byzantine processes, the assumption $n > 5t$ is required to make (S^{prv1}, S^{prv2}) meaningful.

- **Theorem 2** Let $n > 5t$. The condition-sequence pair (S^{prv1}, S^{prv2}) is legal.

Proof *LT1:* We show that $\#_m(I) > 3t + k \wedge dist(J, I) \leq k \Rightarrow \#_m(J) > 3t$.

Let I satisfies $\#_m(I) > 3t + k$. Since $dist(J, I) \leq k$, $\#_m(J) \geq \#_m(I) - k$. Hence $\#_m(J) > 3t + k - k$. Thus we obtain $\#_m(J) > 3t$.

LT2: We show that $\#_m(I) > 2t + k \wedge dist(J, I) \leq k \Rightarrow \#_m(J) > 2t$.

This proof is exactly the same as the proof of *LT1* (with only replacing 3t by 2t).

LA3: Consider $J, J' \in \mathcal{I}_t$. It suffices to show that if $P1^{prv}(J) \wedge dist(J, J') \leq t + \#_\perp(J) + \#_\perp(J')$, $F^{prv}(J) = F^{prv}(J')$ holds.

Since $P1^{prv}(J)$, we have $\#_m(J) > 3t$ and $F^{prv}(J) = m$. Since $dist(J, J') \leq t + \#_\perp(J) + \#_\perp(J')$, J' can contain at most t entries with some values different from m, which are occupied by the value m in J(due to Byzantine). In addition, at most $\#_\perp(J')$ entries of J' can contain $\perp$ value, which are also occupied by the value m in J(due to asynchrony) and at most $\#_\perp(J)$ entries of J' can contain some values, which are occupied by $\perp$ in J. Thus, $\#_m(J') \geq \#_m(J) - t - \#_\perp(J')$. Since $\#_m(J) > 3t$ and $\#_\perp(J') \leq t$, we obtain $\#_m(J') > t$. This implies that, $F^{prv}(J') = m = F^{prv}(J)$.

LA4: Consider $J, J' \in \mathcal{I}_t$. We show that if $P2^{prv}(J) \wedge dist(J, J') \leq \#_\perp(J) + \#_\perp(J')$, $F^{prv}(J) = F^{prv}(J')$ holds.

Since $P2^{prv}(J)$, we have $\#_m(J) > 2t$ and $F^{prv}(J') = m$. Since $dist(J, J') \leq \#_\perp(J) + \#_\perp(J')$ and Byzantine failures appear as crash failures, J' can contain at most $\#_\perp(J')$ entries with the default value, which are occupied by the value m in J. As well as, at most $\#_\perp(J)$ entries of J' can contain some values different from m, which are occupied by $\perp$ in J. Thus, $\#_m(J') \geq \#_m(J) - \#_\perp(J')$. Since $\#_m(J) > 2t$ and $\#_\perp(J') \leq t$, we get $\#_m(J') > t$. This implies that, $F^{prv}(J) = m = F^{prv}(J')$.

LV5: This property is trivially satisfied because $F^{prv}(J)$ is either m (when $\#_m(J) > t$) or the most frequent value in J. □

4. Algorithm *DEX*

In this section, we present a generic doubly-expedited algorithm *DEX* for one-step Byzantine consensus. The algorithm can be instantiated with any legal condition-sequence pair.

Figure 1 provides the pseudocode of the algorithm. It uses an extra communication primitive, called *identical broadcast*, which corresponds to the primitives Id-Send() and Id-Receive() in Figure 1. In contrast, P-Send() and P-Receive() correspond to the standard send/receive primitives. The underlying consensus is served by two primitives UC_propose(v) and UC_decide(v), each of which corresponds to proposal of value v and decision by v.

Informally, the identical broadcast guarantees the delivery of the same message to all processes, even if the message is sent by a faulty process. Its formal specification is described as follows:

```
Function Consensus(v_i)

init: J1_i, J2_i ←⊥^n , decided_i ← False , proposed_i ← False

begin
 1 :  Upon Propose(v_i) do:
 2 :      J1_i[i] ← v_i ; J2_i[i] ← v_i
 3 :      P-Send(v_i) to all processes;
 4 :      Id-Send(v_i) to all processes;

 5 :  Upon P-Receive(v_j) from any process p_j do:
 6 :      J1_i[j] ← v_j;
 7 :      if |J1_i| ≥ n − t and P1(J1_i) and decided_i = False then
 8 :          Decide_i(F(J1_i)); decided_i = True
 9 :      end if

10 :  Upon Id-Receive(v_j) from any process p_j do:
11 :      J2_i[j] ← v_j;
12 :      if |J2_i| ≥ n − t and proposed_i = False then
13 :          UC_propose (F(J2_i));
14 :          proposed_i = True;
15 :      end if
16 :      if |J2_i| ≥ n − t and P2(J2_i) and decided_i = False then
17 :          Decide_i(F(J2_i)); decided_i = True
18 :      end if

19 :  Upon UC_decide(v) do:
20 :      if decided_i = False then
21 :          Decide_i(v) and decided_i = True;
22 :      end if

end
```

Fig. 1 Algorithm DEX: Doubly-Expedited Adaptive algorithm for Byzantine Consensus

Termination If a correct process invokes Id-Send(m), Id-Receive(m) occurs on all correct processes.

Agreement If two processes invoke Id-Receive(m_1) and Id-Receive(m_2) for the same sender, $m_1 = m_2$ holds.

Validity If a correct process invokes Id-Receive(m) for a correct sender p_i, p_i invokes Id-Send(m) exactly at once.

Notice that the use of the identical broadcast does not imply introducing additional assumptions to the system. The identical broadcast can be implemented by using only the standard send/receive primitives. The implementation is easily obtained as a weaker form of simulating identical Byzantine failure on the top of general Byzantine failure models[10]. It should be noted that in that simulation, a single communication step of the identical broadcast is realized by two communications steps of the standard send/receive primitive. Our algorithm uses the identical broadcast to develop the two-step decision scheme. In that sense, our two-step decision scheme can be regarded as a one-step decision scheme in identical Byzantine failure models.

In our algorithm, the part made up of lines 5-9 corresponds to one-step decision, and the another one made up of lines 10-18 corresponds to two-step decision. The algorithm works as follows: Each process p_i starts a consensus execution with invocation of *Consensus(v_i)* where v_i is its initial proposal value. The process p_i sends v_i to other processes by using both *P-send()* and *Id-send()* concurrently, and wait for receiving messages from other processes. By receiving messages, each process constructs views $J1_i$ and $J2_i$, which correspond to one- and two-step decision. The views $J1_i$ and $J2_i$ are maintained incrementally. That is, they are updated by the reception of a message. When at least $n - t$ messages are received in $J1_i$, p_i tries to make a decision by evaluating $P1(J1_i)$. If $P1(J1_i)$ is true, p_i immediately decides $F(J1_i)$, that is, decides in one-step. Otherwise, p_i continues to update $J1_i$. Similarly, when p_i receives at least $n - t$ messages at $J2_i$, it activates the underlying consensus with $F(J2_i)$. In addition, p_i evaluates $P2(J2_i)$ to check whether $J2_i$ is sufficient for taking decision. If $P2(J2_i)$ is true, p_i immediately decides $F(J2_i)$, that is, decides in two steps. Otherwise, p_i repeats the check of $J2_i$ with update of $J2_i$. When the underlying consensus decides, each p_i simply borrows the decision of the underlying consensus unless it has decided already.

4.0.1 Correctness.

We prove the correctness of our algorithm by showing that it provides one-step or two-step decision when it is instantiated with any legal condition-sequence pair (S^1, S^2).

Lemma 1 (Termination) Each correct process p_i eventually decides.

Proof Since there are at most t Byzantine processes, each correct process p_i receives messages from at least $n-t$ processes. It implies that $|J2_i| > n-t$. Hence, p_i certainly initiates the underlying consensus. Since the underlying consensus guarantees termination, p_i can decide when the underlying consensus decides. It follows that each process eventually decides.

Lemma 2 (Agreement) No two correct processes decide different values.

Proof Let two correct processes p_i and p_j decide v_i and v_j respectively. Then we prove $v_i = v_j$. Consider the following six cases.

- (**Case 1:**) *When both p_i and p_j decide in one step at line 8.*
  Since p_i and p_j decide in one step, $P1(J1_i)$ and $P1(J1_j)$ hold. Since there are at most t Byzantine processes, we obtain $dist(J1_i, J1_j) \leq t + \#_\perp(J1_i) + \#_\perp(J1_j)$. From property *LA3*, it follows that $v_i = F(J1_i) = F(J1_j) = v_j$.
- (**Case 2:**) *When p_i decides in one step at line 8 and p_j decides in two steps at line 17.*
  As p_i decides in one step, $P1(J1_i)$ holds. In any view $J2_j$, since there are at most t Byzantine processes, $dist(J1_i, J2_j) \leq t + \#_\perp(J1_i) + \#_\perp(J2_j)$. By property *LA3*, it is clear $v_i = F(J1_i) = F(J2_j) = v_j$. Hence, if p_j decides in two steps using $J2_j$, then its decision value $v_j = v_i$.
- (**Case 3:**) *When both p_i and p_j decide in two steps at line 17.*
  Since p_i and p_j decide in two steps, $P2(J2_i)$ and $P2(J2_j)$ hold. From the agreement property of the identical broadcast primitive, if an entry in $J2_i$ contains a non-default value v, then the same entry in $J2_j$ also contains v. Thus, we obtain $dist(J2_i, J2_j) \leq \#_\perp(J2_i) + \#_\perp(J2_j)$. Because of property $LA4$, if p_i and p_j decide in two steps, then $v_i = v_j$ holds.
- (**Case 4:**) *When p_i decides in one step at line 8 and p_j decides using underlying consensus at line 21.*
  Since p_j decides v_i by the underlying consensus and the underlying consensus satisfies unanimity, it suffices to show that every correct process p_k proposes v_i at line 13. Since p_i decides in one step, $P1(J1_i)$ becomes true. In addition, $dist(J1_i, J2_k) \leq t + \#_\perp(J1_i) + \#_\perp(J2_k)$ because at most t processes are Byzantine. By property *LA3*, $v_k = F(J2_k) = F(J1_i) = v_i$. It implies that every process p_k proposes v_i.
- (**Case 5:**) *When p_i decides in two steps at line 17 and p_j decides using underlying consensus at line 21.*
  Since p_j decides by the underlying consensus, similar to Case 4, we have to show that every correct process p_k proposes v_i to the underlying consensus at line 13. Since p_i decides in two steps, $P2(J2_i)$ holds. In addition, by the same way as the case 3, we obtain $dist(J2_i, J2_k) \leq \#_\perp(J2_i) + \#_\perp(J2_k)$ for any view $J2_k$. By property *LA4*, $F(J2_i) = F(J2_k)$. It implies that every process p_k proposes v_i to the underlying consensus.
- (**Case 6:**) *When both p_i and p_j decide at line 21:* Since the underlying consensus guarantees agreement property, we can conclude $v_i = v_j$.

Lemma 3 (Unanimity) If all correct processes propose the same value v, then no correct process decides the value different from v.

Proof Let f be the actual number of Byzantine processes, and all correct processes propose the same value v. Since $f \leq t$, in each correct process p_i, the views $J1_i$ and $J2_i$ contain no value except v more than t times. If p_i decides at line 8 or 17, its decision value is either $F(J1_i)$ or $F(J2_i)$. From the definition of *LV5*, it is clear that it decides only v. Similarly, since each p_i proposes $F(J2_i)$ to the underlying consensus, and the underlying consensus satisfies unanimity, any correct process that decides using underlying consensus decides only v. Hence, the unanimity holds.

Lemma 4 The algorithm *DEX* guarantees one-step decision for any input vector I, $I \in C^1_k$ if at most k processes exhibit Byzantine behavior.

Proof Since there are at most k Byzantine processes, each correct process p_i receives messages from all $(n-k)$ correct processes. Thus, eventually $J1_i$ becomes a member of $[C^1_k]_k$. This implies that p_i decides in one step.

Lemma 5 The algorithm *DEX* guarantees two-step decision if the input vector I belongs to C^2_k and at most k processes are Byzantine.

Proof Since there are at most k Byzantine processes each correct process p_i receives messages from all $(n-k)$ correct processes. Thus, eventually $J2_i$ becomes a member of $[C_k^2]_k$. This implies that p_i decides in two steps.

The above lemmas imply the following theorem:

Theorem 3 For any instantiation with legal condition-sequence pairs, the algorithm *DEX* is a doubly-expedited one-step consensus algorithm.

5. Conclusion

We proposed a novel one-step Byzantine algorithm *DEX* has two distinguished features: Adaptiveness and double-expedition property. Due to adaptiveness, its condition is sensitive to the actual number of failures, and hence achieves fast termination for large number of inputs when less number of processes are faulty. In addition, due to double-expedition property, it supports two-step decision in addition to one-step decision. Even though *DEX* takes four steps at worst in well-behaved runs while existing algorithms takes only three, it provides fast termination for large number inputs. Practically, this is a favorable feature because the worst case is not so often in real systems, and thus our algorithm can work efficiently in average.

Acknowledgments This work is supported in part by the Japan Society for the Promotion Of Science Grant-in-Aid for Scientific Research(C) 21500013, and the Telecommunication Advancement Foundation.

References

1) I. Keider and S. Rajsbaum : On the cost of fault-tolerant consensus when there are no faults, SIGACT News, Vol.32(9), pp.45–63 (2001)
2) V.F Brasileiro, F. Greve, A. Mostéfaoui and M.Raynal : Consensus in One Communication Step, In proc. of the 6th International Conference on Parallel Computing Technologies, Vol.2127 of LNCS, pp.42–50, Springer-Verlag (2001)
3) Y.J. Song and R.V. Renesse : Bosco: One-Step Byzantine Asynchronous Consensus, In proc. of the 22nd international symposium on Distributed Computing(DISC'08), Vol.5218 of LNCS, pp.438–450, Springer-Verlag (2008)
4) D.Dobre and N. Suri : One-step Consensus with Zero-Degradation, In proc. of the International Conference on Dependable Systems and Networks(DSN'06), pp. 137–146 (2006)
5) P. Dutta and R. Guerraoui : Fast Indulgent Consensus with Zero Degradation, In proc. of the 4th European Dependable Computing Conference on Dependable Computing, Vol. 2485 of LNCS, pp.191–208, Springer-Verlag (2002)
6) T. Izumi and T. Masuzawa : One-Step Consensus Solvability, In proc. of the 22nd international symposium on Distributed Computing(DISC'06), Vol.4167 of LNCS, Springer, pp.224–237 (2006)
7) A. Mostefaoui, S. Rajsbaum and M. Raynal : Conditions on input vectors for consensus solvability in asynchronous distributed systems, In proc. of the thirty-third annual ACM symposium on Theory of computing (STOC'01), pp.153–162 (2001)
8) T. Izumi and T. Masuzawa : Condition Adaptation in Synchronous Consensus, IEEE Transactions on Computers, Vol.55(7), pp.843–853 (2006)
9) R. Guerraoui and M. Raynal : The Information Structure of Indulgent Consensus, IEEE Transactions on Computers, Vol.53(4), pp.453–466 (2004)
10) H. Attiya and J. Welch : Distributed Computing: Fundamentals, Simulations and Advanced Topics, Wiley (2004)