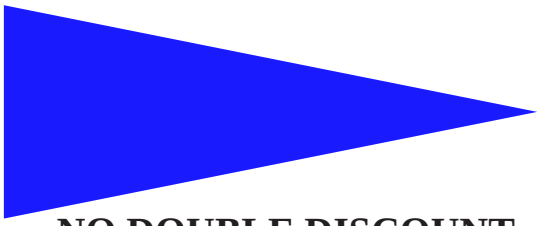
No Double Discount: Condition-based Simultaneity Yields Limited Gain

Y. Moses, Michel Raynal

I R I S A
INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES

PUBLICATION
INTERNE
N° 1898



NO DOUBLE DISCOUNT:
CONDITION-BASED SIMULTANEITY YIELDS LIMITED GAIN

Y. MOSES M. RAYNAL

ISSN 1166-8687

IRISA
CAMPUS UNIVERSITAIRE DE BEAULIEU - 35042 RENNES CEDEX - FRANCE

IRISA

INSTITUT DE RECHERCHE EN INFORMATIQUE ET SYSTÈMES ALÉATOIRES
Campus de Beaulieu – 35042 Rennes Cedex – France
Tél. : (33) 02 99 84 71 00 – Fax : (33) 02 99 84 71 71
http://www.irisa.fr

No Double Discount: Condition-based Simultaneity Yields Limited Gain

Y. Moses* M. Raynal**

Systèmes communicants
Projet ASAP

Publication interne n ° 1898 — Juillet 2008 — 20 pages

Abstract: Assuming each process proposes a value, the consensus problem requires the non-faulty processes to agree on the same value that has to be a proposed value. Solutions to the consensus problem in synchronous systems are based on the round-based model, namely, the progress of the processes is according to synchronous rounds. Simultaneous consensus requires that the non-faulty processes decide not only on the same value, but decide during the very same round. It has been shown by Dwork and Moses that, in a synchronous system prone to t process crashes, the earliest round at which a common decision can be simultaneously obtained is $(t+1)-D$ where D is a non-negative integer determined by the actual failure pattern F.

The condition-based approach to solve consensus assumes that the input vector belongs to a set C (a set of input vectors satisfying a property called legality). Interestingly, the conditions for synchronous consensus define a hierarchy of sets of conditions $\mathcal{S}_t^0 \subset \cdots \subset \mathcal{S}_t^d \subset \cdots \subset \mathcal{S}_t^t$ (where the set $\mathcal{S}_t^t$ contains the condition made up of all possible input vectors). It has been shown that $d+1$ is a tight lower bound on the minimal number of rounds for synchronous condition-based consensus.

This paper considers the synchronous condition-based consensus problem with simultaneous decision. It first presents a simple algorithm that directs the processes to decide simultaneously at the end of the round $RS_{t,d,F} = \min((t+1)-D, d+1)$ (i.e., $RS_{t,d,F} = (t+1) - \max(D,\delta)$ with $\delta = t-d$). The paper then shows that $RS_{t,d,F}$ is a lower bound for the condition-based simultaneous consensus problem. It thus follows that the algorithm designed is optimal *in each and every run*, and not just in the worst case: For every choice of failure pattern by the adversary (and every input configuration), the algorithm reaches simultaneous as fast as any correct algorithm could do under the same conditions.

This shows that, contrary to what could be hoped, when considering condition-based consensus with simultaneous decision, we can benefit from the best of both actual worlds (either the failure world when $RS_{t,d,F} = (t+1)-D$, or the condition world when $RS_{t,d,F} = d+1$), but we cannot benefit from the sum of savings offered by both. Only one discount applies.

Key-words: Agreement problem, Condition-based agreement, Consensus, Distributed algorithm, Early decision, Lower bound, Modularity, Process crash failure, Round-based computation model, Simultaneity, Synchronous message-passing system.

(Résumé : tsvp)

* Department of Electrical Engineering, Technion, Haifa, 32000 Israel moses@ee.technion.ac.il
** IRISA, Université de Rennes 1, Campus de Beaulieu, 35042 Rennes Cedex, France raynal@irisa.fr

Proceedings of the 22nd International Symposium on Distributed Computing (DISC'2008).

Centre National de la Recherche Scientifique
(UMR 6074) Université de Rennes 1 – Insa de Rennes

Institut National de Recherche en Informatique
et en Automatique – unité de recherche de Rennes

Utiliser les conditions pour une décision simultanée donne peu de gain

Résumé : Ce rapport montre qu'utiliser les conditions pour une décision simultanée ne donne qu'un gain limité.

Mots clés : Système synchrone, algorithme distribué, condition, consensus, crash de processus, décision simultaée, modèle de calcul fondé sur les rondes, système synchrone.

1 Introduction

The consensus problem Fault-tolerant systems often require a means by which processes or processors can arrive at an exact mutual agreement of some kind [17]. If the processes defining a computation have never to agree, that computation is actually made up of a set of independent computations, and consequently is not an inherently distributed computation. The agreement requirement is captured by the *consensus* problem that is one of the most important problems of fault-tolerant distributed computing. It actually occurs every time entities (usually called agents, processes -the word we use in the following-, nodes, sensors, etc.) have to agree. The consensus problem is surprisingly simple to state: each process is assumed to propose a value, and all the processes that are not faulty have to agree/decide (termination), on the same value (agreement), that has to be one of the proposed values (validity). The failure model considered in this paper is the process crash model.

While consensus is impossible to solve in pure asynchronous systems despite even a single process crash [5] ("pure asynchronous systems" means systems in which there is no upper bound on process speed and message transfer delay), it can be solved in synchronous systems (i.e., systems where there are such upper bounds) whatever the number n of processes and the number t of process crashes ($t < n$).

An important measure for a consensus algorithm is the time it takes for the non-faulty processes to decide. As a computation in a synchronous system can be abstracted as a *sequence of rounds*, the time complexity of a synchronous consensus algorithm is measured as the minimal number of rounds (R_t) a process has to execute before deciding, in the worst case scenario. It has been shown (see, e.g., in [4, 11]) that $R_t = t + 1$. Moreover, that bound is tight: there exist algorithms (e.g., see [1, 9, 18]) where no process ever executes more than R_t rounds (these algorithms are thus optimal with respect to that bound).

Early decision While $t + 1$ rounds are needed in the worst case scenario, the major part of the executions have few failures or are even failure-free. So, an important issue is to be able to design *early deciding* algorithms, i.e., algorithms that direct the processes to decide "as early as possible" in good scenarios. Let f, $0 \leq f \leq t$, be the number of actual process crashes in an execution. It has been shown that the lower bound on the number of rounds is then $R_{t,f} = \min(f + 2, t + 1)$ (e.g., [2, 11, 19]). As before, this bound is tight: algorithms in which no process ever executes more than $R_{t,f}$ exist (e.g., see [2, 7, 18]).

The condition-based approach and the hierarchy of synchronous conditions The condition-based approach has been initially introduced to circumvent the impossibility of solving consensus in an asynchronous system prone to crash failures [14]. It consists in restricting the set of input vectors that can be proposed (such a set is called a *condition*) in such a way that consensus becomes solvable in crash-prone asynchronous systems every time the input vector belongs to the condition. A main result associated with this approach is the following [14] : a condition C allows to solve consensus in an asynchronous system prone to up to x process crashes iff it is x-legal (a condition is x-legal if each of its input vectors contains more than x times the value decided from that vector, and the Hamming distance of two vectors from which different values can be decided is greater than x). As we can see, there is a strong connection between the condition-based approach and error-correcting codes [6].

The condition-based approach has then been extended from computability in asynchronous systems to efficiency in synchronous systems. More precisely, let us consider a synchronous system where up to t processes can crash, and let $\mathcal{S}_t^d$ be the set containing all the $(t-d)$-legal conditions (the parameter d is called the *degree* of the condition, and the quantity $(t-d)$ measures the *difficulty* of the condition). The following hierarchy has been established in [15]: $\mathcal{S}_t^0 \subset \mathcal{S}_t^1 \subset \cdots \subset \mathcal{S}_t^d \subset \cdots \subset \mathcal{S}_t^t$.

Considering a condition $C \in \mathcal{S}_t^d$ and an input vector $I \in C$, it is shown in [15] that synchronous consensus can be solved in two rounds when $d = 0$, and in $d + 1$ rounds when $1 \leq d \leq t$. It is also shown that $d + 1$ is a tight lower bound, when the input vector belong to C (with $C \in \mathcal{S}_t^d$ and $C \notin \mathcal{S}_t^{d-1}$). It is worthwhile looking at the "extreme" sets $\mathcal{S}_t^0$ and $\mathcal{S}_t^t$. On one side, $\mathcal{S}_t^t$ includes the condition that contains all the possible input vectors. On the other side, the family of conditions $\mathcal{S}_t^0$, that is the largest set of conditions that allow *to solve* the consensus problem in asynchronous systems prone to up to t crashes, is also the family of conditions that allows to solve the consensus problem *optimally* in a synchronous system prone to t crashes.

Simultaneous decision Consensus agreement is a data agreement property, namely the processes have to agree on the same value. According to the actual failure pattern, and the way this pattern is perceived by the processes, it is possible for several processes to decide at distinct rounds. The only guarantee lies in the fact that this round can be bounded (by R_t or $R_{t,f}$ if we consider an early-deciding algorithm, or $d+1$ if we consider a condition-based algorithm

instantiated with a condition $C \in \mathcal{S}_t^d$ and assume the input vector always belongs to C). This uncertainty on the set of round numbers at which the processes decide, can be a serious drawback for the real-time oriented applications where agreement is required, not only on the decided value, but also on the time the decision is taken. More precisely, these applications require that the processes decide on the same value (*data agreement*), during the very same round (*time agreement*). This property is also called *simultaneous decision*.

All the "classical" consensus algorithms where all the processes that do not crash decide systematically at the end of the round $R_t = t + 1$ ensure simultaneous decision. It is shown in [2, 12] that the earliest round number at the end of which a common decision can be simultaneously taken is $RS_{t,F} = (t + 1) - D$ where D is a non-negative integer whose value depends on the actual failure pattern F.[1] More precisely, let $C[r]$ be the set of the processes that are seen as crashed by (at least) one of the processes that survive (i.e., do not crash before the end of) round r. For any r, let $d_r = \max(0, |C[r]| - r)$. We have $D = \max_{r \geq 0}(d_r)$. So, when considering the bound $(t + 1) - D$, D represents the number of rounds that could be saved with respect to the worst case $t + 1$ bound, while guaranteeing simultaneous agreement. Algorithms that solve simultaneous consensus optimally are described in [3, 12]. It is interesting to notice that the greatest values of D are obtained when many processes crash early. If at most one process crashes at each round, we obtain $D = 0$ and the earliest round for simultaneous decision is then $t + 1$. At first glance, this can appear counter-intuitive. Actually it is not, it is only the consequence of the fact that crashes are stable while "not to be crashed" is not a stable property, and simultaneous decision requires a common knowledge that can be based only on stable facts.

Content of the paper The paper investigates the simultaneous decision in the context of the condition-based approach. It addresses the following question: "Assuming a condition $C \in \mathcal{S}_t^d$, an input vector $I \in C$, a failure pattern F, which is the earliest round at which a simultaneous decision can be obtained?"

Let $\delta = t - d$ (i.e., C is δ-legal). The paper shows that $RS_{t,d,F} = (t + 1) - \max(D, \delta)$ (or equivalently, $\min(t + 1 - D, d + 1)$) is a tight lower bound. To show it, the paper first presents a simple condition-based algorithm where the processes simultaneously decide at the end of the round $RS_{t,d,F}$. It then addresses the more difficult side, namely it shows that there is no condition-based algorithm for simultaneous agreement that, given a run with the failure pattern F, directs the processes to simultaneously decide at round r with $r < RS_{t,d,F}$. Hence, the paper shows that there is no "double discount", one coming from the condition the input vector belongs to, the other one coming from the best gain (in terms of the number of rounds) due to early crashes. The best that can be done is only to benefit from the best world that occurs in the actual run, namely, $(t + 1) - D$ or $(t + 1) - \delta$. The benefits of both worlds cannot be "added" to provide a bound smaller than the smallest of them.

Roughly speaking, our algorithm is obtained by running a condition-based algorithm that is based on [14], that runs for $t + 1 - \delta$ rounds in parallel with a simultaneous consensus algorithm based on [3, 12], that in turn decides in $t + 1 - D$ rounds, where D depends on the failure pattern. If $D \geq \delta$ then the latter algorithm halts first, and the decision value that it produces is adopted. Otherwise, consensus is reached in $t + 1 - \delta$ rounds, using the decision value produced by the first (condition-based) algorithm. Because decisions are simultaneous, the algorithms can be seamlessly combined in this way. The main technical contribution of the paper is in the lower bound proof, showing that this scheme cannot be improved on. It yields optimally fast agreement in *all* runs.

2 Computation model, conditions and problem specification

2.1 Computation model

Round-based synchronous system The system model consists of a finite set of processes, namely, $\Pi = \{p_1, \ldots, p_n\}$, that communicate and synchronize by sending and receiving messages through channels. (Sometimes we also use p and q to denote processes.) Every pair of processes is connected by a bi-directional reliable channel (which means that there is no creation, alteration, loss or duplication of message).

The system is *round-based synchronous*. This means that each execution consists of a sequence of *rounds*. Those are identified by the successive integers $1, 2$, etc. For the processes, the current round number appears as a global variable r that they can read, and whose progress is managed by the underlying system. A round is made up of three consecutive phases:

- A send phase in which each process sends the same message to all the processes (including itself).

[1] Let us recall that the actual number of failures f ($0 \leq f \leq t$) is only a digest of the failure pattern. It does not state at which rounds the failures do occur, and which are the processes that received the round r message from a process that crashed during round r.

- A receive phase in which each process receives messages.
  The fundamental property of the synchronous model lies in the fact that a message sent by a process p_i to a process p_j at round r, is received by p_j at the very same round r.
- A computation phase during which each process processes the messages it received during that round and executes local computation.

Process failure model A process is *faulty* during an execution if its behavior deviates from that prescribed by its algorithm, otherwise it is *correct*. We consider here the crash failure model, namely, a faulty process stops its execution prematurely. After it has crashed, a process does nothing. If a process crashes in the middle of a sending phase, only a subset of the messages it was supposed to send might actually be received.

As already indicated, the model parameter t ($1 \leq t < n$) denotes an upper bound on the number of processes that can crash in a run. A *failure pattern* F is a list of at most t triples $\langle q, k_q, B_q \rangle$. A triple $\langle q, k_q, B_q \rangle$ states that the process q crashes while executing the round k_q (hence, it sends no messages after round k_q), while the set B_q denotes the set of processes that do not receive the message sent by q during the round k_q.

2.2 The simultaneous consensus problem

The problem has been informally stated in the introduction: every process p_i *proposes* a value v_i (called its *initial value*) and the processes have to *decide*, during the very same round, on the same value that has to be one of the proposed values. This can be stated as a set of four properties that any algorithm solving the problem has to satisfy.

- Decision. Every correct process decides.
- Validity. A decided value is a proposed value.
- Data agreement. No two processes decide different values.
- Simultaneous decision (or Time agreement). No two processes decide at distinct rounds.

2.3 The condition-based approach

Notation Let $\mathcal{V}$ be the set containing all the values that can be proposed, with $|\mathcal{V}| \geq 2$. An *input configuration* is an assignment $\mathcal{I} : \Pi \rightarrow \mathcal{V}$ of an initial value $v_i \in \mathcal{V}$ to each process p_i. An *input vector* is a size n vector corresponding to an input configuration. A condition is a set of input vectors.

Let $\perp$ denote a default value such that $\perp \notin \mathcal{V}$ and $\forall a \in \mathcal{V}$, $\perp < a$. We usually denote by I an input vector (all its entries are in $\mathcal{V}$), and with J a vector that may have some entries equal to $\perp$. Such a vector J is called a *view*. The number of occurrences of a value $a \in \mathcal{V} \cup \{\perp\}$ in the vector J is denoted $\#_a(J)$. Given two vectors $I1$ and $I2$, $dist(I1, I2)$ denotes their Hamming distance.

Legality Not all the conditions allow to solve consensus in an asynchronous distributed system prone to process crashes (due to the FLP impossibility result [5]). So, to solve consensus in such a system, a condition C has to meet constraints. Those can be expressed with the legality notion defined as follows:

Definition 1 [6, 14] *A condition C is* x-legal *if there exists a function $\mathcal{H} : C \mapsto \mathcal{V}$ with the following properties:* (1) $\forall I \in C$: $\#_{\mathcal{H}(I)}(I) > x$, *and* (2) $\forall I1, I2 \in C$: $\mathcal{H}(I1) \neq \mathcal{H}(I2) \Rightarrow dist(I1, I2) > x$.

This means that the value extracted from I by $\mathcal{H}()$ appears "often enough" in I (more than x times), and two vectors from which different values are extracted by $\mathcal{H}()$ are "far enough apart" from the Hamming distance point of view. A main result of [14] is the characterization of the largest set of conditions that allow to solve consensus in an asynchronous system:

Theorem 1 [14] *If C is x-legal then consensus can be solved under C in an asynchronous system prone to up to x process crashes. Conversely, there exists an $(x-1)$-legal condition C' for which consensus is unsolvable in the presence of x process crashes.*

Intuitively, a condition selects a proposed value to become the decided value, namely, the value decided from an input vector I is $\mathcal{H}(I)$. A general method to define conditions is described in [16]. As an example, let us consider the condition M_x defined as follows from the function $\max()$ (where $\max(I)$ returns the greatest value in I): $I \in M_x \Leftrightarrow \#_{\max(I)}(I) > x$. It is shown in [14] that M_x is x-legal, and consequently consensus can be solved despite up to x process crashes when the input vectors are such that each of them has at least $x+1$ entries equal to its greatest entry.

Lemma 1 [15] *Let C be an x-legal condition. If $I1, I2 \in C$, $\#_\perp(J) \leq x$, $J \leq I1$ and $J \leq I2$, we have $\mathcal{H}(I1) = \mathcal{H}(I2)$.*

Because of this lemma, $\mathcal{H}()$ can be extended to vectors J with at most x entries equal to $\perp$ by choosing an arbitrary $I \in C$ with $J \leq I$.

Definition 2 *Let C be an x-legal condition, I a vector in C, and J any vector such that $J \leq I$ and $\#_\perp(J) \leq x$. The definition of $\mathcal{H}()$ is extended to such vectors J as follows: $\mathcal{H}(J) = \mathcal{H}(I)$.*

Theorem 2 [14] *Let C be an x-legal condition. It is also $(x-1)$-legal. Moreover, there exist conditions that are $(x-1)$-legal but not x-legal.*

The synchronous hierarchy of classes of conditions Let us consider a synchronous system prone to up to t process crashes, and let $\mathcal{S}_t^d$ be the set of all the $(t-d)$-legal conditions. Let us observe that $d = t$ defines the class of 0-legal conditions which is the largest one and includes every condition (and in particular the "trivial" condition that contains all the possible input vectors). Due to Theorem 2 the classes $(\mathcal{S}_t^d)_{0 \leq d \leq t}$ define the following hierarchy [14]:

$$\mathcal{S}_t^0 \subset \cdots \subset \mathcal{S}_t^d \subset \mathcal{S}_t^{d+1} \subset \cdots \subset \mathcal{S}_t^t.$$

As we shall see, this hierarchy of conditions allows solving synchronous consensus with protocols that take more and more rounds, as we go from $d = 0$ to $d = t$. And $d = 0$ is the borderline case where consensus can be solved in an asynchronous system with t failures, and can be solved optimally in a synchronous system.

2.4 The simultaneous condition-based consensus problem

Let us assume that the input vectors always belong to a condition $C \in \mathcal{S}_t^d$. Let $\delta = t - d$ (i.e., C is δ-legal). In order to eliminate the trivial solution where the processes always decide at round $d + 1$ (= $t + 1 - \delta$), the simultaneous condition-based consensus problem is defined by the four properties stated in Section 2.2, plus the following early-deciding property:

- Early decision. No process decides after the round $RS_{t,d,F} = (t+1) - \max(D, \delta)$ (where D is the value derived from failure pattern F as described in the introduction).

3 An optimal condition-based simultaneous consensus algorithm

This section presents a simple condition-based simultaneous consensus algorithm in which the processes decide at the end of the round $RS_{t,d,F} = (t+1) - \min(D, \delta)$. It will be shown in Section 4 that $RS_{t,d,F}$ is the smallest number of rounds for simultaneous condition-based consensus.

The proposed algorithm is built modularly. It combines two base algorithms. One is a condition-based algorithm that, when instantiated with a δ-legal condition C (i.e., $C \in \mathcal{S}_t^d$, with $\delta = t - d$, $1 \leq d \leq t$, and the input vector I belongs to C) directs the processes to decide simultaneously at the end of round $d + 1 = t + 1 - \delta$. The second is a simultaneous (non-condition-based) consensus algorithm that directs the processes to decide at the end of the round $t + 1 - D$. These base algorithms are first presented. Their combination in which the processes simultaneously decide at the end of the round round $RS_{t,d,F} = (t+1) - \min(D, \delta)$ is then described.

3.1 A condition-based simultaneous consensus algorithm

3.1.1 Preliminary definitions

As the system model requires each process to send a message to all the processes at each round, process failures can be easily detected, and this detection is done as soon as possible.

Failure discovery The failure of a process q is *discovered* (for the first time) in round r if r is the first round such that there is a process p that (1) does not receive a round r message from q, and (2) survives (i.e., completes without crashing) round r.

Clean rounds A round r is *clean* if no process is discovered faulty for the first time in that round [3]. This means that a process that crashes during a clean round r has sent its round r message to all the processes that proceed to the round $r+1$.

The following property is an immediate consequence of the previous definitions. (In a precise sense, a clean round can be used to ensure that the knowledge of the various processes is identical. Once this happens the processes are in agreement about initial values. They then need to discover this and coordinate their decisions.)

Property 1 *If round r is clean, then all the processes that proceed to round $r+1$ received, during the round r, messages from the same set of processes (including at least all of them).*

Observe that a clean round is not necessarily a failure-free round. It is possible for a process p to crash in a clean round r in such a way that no process surviving round r notices its crash in round r (p crashes after sending its round r messages at least to all processes that survive round r and before the end of the round). Similarly, a failure-free round is not necessarily clean. For example, a failure-free round $r+1$ that follows a clean round r during which a crash occurred is not clean.

3.1.2 A simple (non-optimal) condition-based algorithm

The first algorithm we present directs the processes to decide at the end of the round $(t+1)-\delta$. It is an adaptation of an algorithm described in [15] that does not satisfy the simultaneous decision property[2]. The algorithm is described in Figure 1. Each process p_i uses three local variables.

- V_i is an array whose aim is to contain p_i's local view of the input vector. Initialized to $[\perp, \ldots, \perp]$, it contains at most t entries equal to $\perp$ at the end of the first round (line 105).
- v_cond_i (initialized to $\perp$) is destined to contain the value $\mathcal{H}(I)$ that the condition C associates with the input vector I (line 106). As the condition C is δ-legal, $\mathcal{H}(I)$ can be computed from $\mathcal{H}(V_i)$ only when the local view V_i of p_i has at most δ entries equal to $\perp$ (see Lemma 1 and Definition 2).
- v_nocond_i (initialized to $\perp$) is destined to contain the value to be decided when no value can be decided from the condition because there are too many failures during the first round (more than δ processes crash). When this occurs, a process will decide the greatest proposed value it has ever seen.

The behavior of a process p_i is simple. During the first round (lines 102- 108), p_i determines its local view V_i of the input vector I, computes v_cond_i if it sees not too many failures (i.e., at most δ crashes), and computes v_nocond_i in case the condition is useless because there are more than δ crashes. Then, from the second round until round $t+1-\delta = d+1$, the processes use the flood set strategy to exchange their current values v_cond_i and v_nocond_i, and keep the greatest ones of each. At the end of the round $t+1-\delta$, a process p_i decides the value in v_cond_i if that value is not $\perp$; otherwise, it decides the value in v_nocond_i (that is then different from $\perp$ as we will see in the proof).

Theorem 3 *The algorithm described in Figure 1 solves the condition-based consensus problem. Moreover, the processes decide at the end of the round $(t+1)-\delta$.*

The proof appears in Appendix A.

3.2 An optimal algorithm for simultaneous consensus

The second base algorithm solves optimally the simultaneous consensus problem, i.e., the processes decide at the end of the round $(t+1)-D$ (where the non-negative integer D - defined in the Introduction- depends on the failure pattern). This algorithm is from [12]. We describe it for self-containment of the paper.

3.2.1 The horizon notion

Given a process p_i and a round $r \geq 1$, let x be the number of process crashes that (1) occurred between the round 1 and the round $r-1$ (included) and (2) are known by p_i by the end of r.

The horizon notion was introduced in [10]. A horizon value is associated with each round r, more precisely, the value $h_i(r) = r+t-x$ is the *horizon* of p_i at round r. We have $h_i(1) = t+1$. If three crashes occurred by the end of the first round and are reported to p_i during the second round, we have $h_i(2) = t-1$.

[2] The algorithm described in [15] works be the input vector in the condition or not. When the input vector I belongs to the condition C, a process decides in two rounds if $f \leq t-d$, and in at most $d+1$ rounds when $f > t-d$. When the input vector is not in the condition a process decides in at most $t+1$ rounds.

```
operation propose(v_i):
(101)  V_i ← [⊥, ..., ⊥]; v_cond_i ← ⊥; v_nocond_i ← ⊥;
(102)  when r = 1
(103)  begin round
(104)    send (v_i) to all;
(105)    for each v_j received do V_i[j] ← v_j end for;
(106)    if (#_⊥(V_i) ≤ δ) then v_cond_i ← H(V_i) end if;
(107)    v_nocond_i ← max( all the v_j received during r)
(108)  end round;
(109)  when r = 2, 3, ... do
(110)  begin round
(111)    send (v_cond_i, v_nocond_i) to all;
(112)    v_cond_i   ← max( all the v_cond_j received during r);
(113)    v_nocond_i ← max( all the v_tmf_j received during r);
(114)    if (r = (t + 1) − δ) then
(115)      if (v_cond_i ≠ ⊥)  then return (v_cond_i) else return (v_nocond_i) end if
(116)    end if
(117)  end round
```

Figure 1: A synchronous condition-based simultaneous consensus algorithm (code for p_i)

As we will see, the horizon notion is a key notion to determine the smallest round at the end of which the same value can be simultaneously decided. The following simple theorem (that will be exploited in the presentation of the algorithm) explains why this notion is crucial.

Theorem 4 [10] *Let x be defined as indicated above, and p_i a process that survives round r. There is a clean round y such that $r \leq y \leq h_i(r) = r + t - x$.*

Proof Let us first observe that, as at least x processes have been discovered as faulty between the round 0 and the round $r-1$ (included), at most $t-x$ processes can be discovered as faulty between the round r (included) and the round $r+t-x$ (included). But there are $t-x+1$ rounds from r to $r+t-x$, from which we conclude that at least one of these rounds is clean. $\square_{Theorem\ 4}$

3.2.2 The algorithm

Local variables Each process p_i manages the following local variables. Some variables are presented as belonging to an array. This is only for notational convenience, as such array variables can be implemented as simple variables.

- est_i contains, at the end of r, p_i's current estimate of the decision value. Its initial value is v_i, the value proposed by p_i.
- $f_i[r]$ denotes the set of processes from which p_i has not received a message during the round r. (So, this variable is the best current estimate that p_i can have of the processes that have crashed.)
  Let $\overline{f_i[r]} = \Pi \setminus f_i[r]$ (i.e., the set of processes from which p_i has received a round r message).
- $f'_i[r-1]$ is a value computed by p_i during the round r, but that refers to crashes that occurred up to the round $r-1$ (included), hence the notation. It is the value $\bigcup_{p_j \in \overline{f_i[r]}} f_j[r-1]$, which means that $f'_i[r-1]$ is the set of processes that were known as crashed at the end of the round $r-1$ by at least one of the processes from which p_i has received a round r message. This value is computed by p_i during the round r. As a process p_i receives its own messages, we have $f_i[r-1] \subseteq f'_i[r-1]$.
- $bh_i[r]$ represents the best (smallest) horizon value known by p_i at round r. It is p_i's best estimate of the smallest round for a simultaneous decision. Initially, $bh_i[0] = h_i(0) = t+1$.

Process behavior Each process p_i not crashed at the beginning of r sends to all the processes a message containing its current estimate of the decision value (est_i), and the set $f_i[r-1]$ of processes it currently knows as faulty. After it has received the round r messages, p_i computes the new value of est_i and the value of $bh_i[r]$. The new value of est_i is the smallest of the estimates values it has seen so far. As far as the value of $bh_i[r]$ is concerned, we have the following.

- The computation of $bh_i[r]$ has to take into account $h_i(r)$. This is required to benefit from Theorem 4 that states that there is a clean round y such that $r \leq y \leq h_i(r)$. When this clean round will be executed, any two processes p_i and p_j that execute it will have $est_i = est_j$, and (as they will receive messages from the same set

of processes, see Property 1) will be such that $f'_i[r-1] = f'_j[r-1]$. It follows that, we will have $h_i(y) = h_j(y)$, thereby creating correct "seeds" for determining the smallest round for a simultaneous decision. This allows the processes to determine rounds at which they can simultaneously decide.

- As we are looking for the first round where a simultaneous decision is possible, $bh_i[r]$ has to be set to $\min\big(h_i(0), h_i(1), \ldots, h_i$ i.e., $bh_i[r] = \min\big(bh_i[r-1], h_i(r)\big)$.

Finally, according to the previous discussion, the algorithm directs a process p_i to decide at the end of the first round r that is equal to the best horizon currently known by p_i, i.e., when $r = bh_i[r]$.

The resulting algorithm is presented in Figure 2, where $h_i(r)$ (see line 208) is expressed as a function of $r-1$ to emphasize the fact that it could be computed at the end of the round $r-1$ by an external omniscient observer. The local boolean variable $decided_i$ is used only to prove the optimality of the combined algorithm (see Section 4). Its suppression does not alter the algorithm.

```
algorithm PROPOSE(v_i):
(201)  est_i ← v_i; bh_i[0] ← t+1; f_i[0] ← ∅; decided_i ← false;      % initialization %
(202)  when r = 1, 2, ... do      % r: round number %
(203)  begin round
(204)      send (est_i, f_i[r−1]) to all;      % including p_i itself %
(205)      let f'_i[r−1] = the union of the f_j[r−1] sets received during r;
(206)      let f_i[r] = the set of processes from which p_i has not received a message during r;
(207)      est_i ← min( all the est_j received during r);
(208)      let h_i(r) = (r−1) + (t+1−|f'_i[r−1]|);
(209)      bh_i[r] ← min(bh_i[r−1], h_i(r));
(210)      if r = bh_i[r] then decided_i ← true; return (est_i) end if
(211)  end round
```

Figure 2: Optimal simultaneous consensus despite up to t crash failures (code for p_i)

3.3 Proof of the algorithm

Definition 3 *Given an execution, let F be the failure pattern that occurs in that execution.*

- *$S[r] = S[r, F]$ is the set of processes that survive (i.e., complete) round r according to F.*
- *$C[r] = C[r, F] = \bigcup_{p_i \in S[r]} f_i[r]$, i.e., the set of the processes that are known to have crashed by at least one of the processes that survives round r. Observe that $f'_i[r] \subseteq C[r]$, for any $p_i \in S[r]$.*
- *$d_r = |C[r]| - r$, for every round r. Based on the values d_r, we define $D = \max_{r \geq 0}(d_r)$. (In the introduction, D has been called the waste inherent in F, i.e., the number of rounds the adversary has lost in his quest to delay decision for as long as possible.)*

Let us observe that, as no process can be discovered faulty before the first round, we have $C[0] = 0$. More generally, we assume $C[r] = 0$ for all $r \leq 0$ (the fictitious rounds -1 and 0 will be used in section 4 for ease of exposition). Notice also that $D \geq 0$, since $C[0] = 0$ and $D \geq d_0 = C[0] - 0 = 0$.

The following optimality results are shown in [3]: the smallest number of rounds $RS_{t,F}$ that any simultaneous decision consensus algorithm can achieve is $RS_{t,F} = (t+1) - D$ when $t < n-1$, and $RS_{t,F} = t - D$ when $t = n-1$. The following theorem is proved in [12]. For completeness, this proof is given in appendix B.

Theorem 5 *Let $t < n$. The algorithm described in Figure 2 solves the consensus problem with simultaneous decision. In a run with failure pattern F, decision is reached in round $t+1-D$, where $D = D(F)$ is the waste inherent in F.*

3.4 An optimal condition-based simultaneous consensus algorithm

As previously announced, an optimal condition-based simultaneous consensus algorithm can be obtained from the two base algorithms described in Figures 1 and 2. Their combination consists in executing both algorithms in parallel as follows:

1. The r-th round, $1 \leq r \leq t+1-\delta$, of the combined algorithm is a simple merge of the r-th round of both algorithms. This means that the message sent by p_i at round r now piggybacks v_cond_i, v_nocond_i, est_i and $f_i[r-1]$.[3]
2. The lines 114-116 of the algorithm in Figure 1, and the line 210 of the algorithm in Figure 2 are replaced by the following lines:

```
if (r = bh_i[r]) ∨ (r = t + 1 − δ) then
  if (r = bh_i[r]) then decided_i ← true; return (est_i)
                   else if (v_cond_i ≠ ⊥) then return (v_cond_i)
                                          else return (v_nocond_i) end if
end if
```

The following theorem is an immediate consequence of the combination of Theorem 3 and Theorem 5. More precisely, if the algorithm described in Figure 2 terminates first (case $r = bh_i[r] < t+1-\delta$), or both terminate at the same round (case $r = bh_i[r] = t+1-\delta$), that algorithm imposes the common early decision round, namely $bh_i[r]$. Otherwise, the condition-based algorithm imposes $t+1-\delta$ as the common early decision round.

Theorem 6 *Let $t < n$. The algorithm obtained by the combined execution (as described in the previous items) of the algorithms described in Figures 1 and 2 solves the condition-based simultaneous consensus problem. In a run with failure pattern F, decision is reached in round $t+1-\max(D,\delta)$.*

4 On the optimality of the algorithm: $t+1-\min(D,\delta)$ is a lower bound

This section proves that the algorithm described in Section 3.4 is optimal: In a synchronous system prone to up to t process crashes (with $t < n-1$), there is no deterministic algorithm that can ever solve the simultaneous condition-based consensus problem in fewer than $(t+1)-\max(D,\delta)$ rounds. The proof relies on notions introduced in [3, 10, 12]. We are unable to present the definitions and the proof in the main text due to page limitation. A complete treatment is provided in Appendix C. In this section we sketch the main ideas of the proof, with precise definitions and proofs in the Appendix.

Consider a condition $C = \{0^n, 1^n\}$ containing only two extreme initial configurations: All zeros and all ones. Clearly, consensus can be solved for C with no rounds of communication. Observe that C is x-legal for $1 \leq x \leq n$. Thus, the property of being x-legal is useful mainly for establishing upper bounds on consensus as demonstrated in Theorems 1 and 3. For the purpose of proving a matching lower bound, we define a condition C to be *x-coverable* if, roughly speaking, for every $v \in \mathcal{V}$ and $I \in C$, an input configuration that does not contain v is reachable from I by a finite sequence of steps among elements of C where a step is allowed between configurations whose Hamming distance is at most x.

The lower bound is based on the well-known connection between simultaneous agreement and common knowledge [3, 13]. This connection implies that it is possible to simultaneously decide on a value $v \in \mathcal{V}$ only once it becomes common knowledge that one of the initial values of I is v. This has a nice graph-theoretic interpretation: For a fixed protocol and a round $r \geq 0$ we can define a graph $\mathsf{G}(r)$ whose nodes correspond to runs of the protocol. This graph has the property that the existence of an initial value of I is common knowledge in round r of a run σ if and only if every run in σ's connected component in $\mathsf{G}(r)$ contains at least one initial value of I. Let D be the *waste* inherent in a failure pattern F. The analysis of common knowledge in crash failures shows that common knowledge of initial values is attained at the end of round $t+1-D$ [3, 10]. However, before round $t+1-D$, it is not common knowledge that even one failure has occurred. The crux of the proof involves proving the following claim.

Claim *Suppose that the condition C is x-coverable and that $t+1-D \leq t+1-x$. Moreover, let σ and σ' be two runs with input configurations I, I', respectively, where the waste in each of the runs does not exceed D. If the Hamming distance dist$(I, I') \leq x$ then σ and σ' are in the same connected component of $\mathsf{G}(r)$ for all rounds $r < t+1-D$.*

Using this claim and the definition of x-coverable, it is then shown that no initial value becomes common knowledge (and thus simultaneous decision is impossible) before round $RS_{t,d,F} = (t+1)-\max(D,\delta)$. This yields matching upper and lower bounds for each and every run.

[3] A close look at the base algorithms shows that their variables v_nocond_i and est_i play the same role. Consequently, only of them has to be kept in the combined algorithm.

5 Conclusion

This paper focused on simultaneous decision in the condition-based consensus setting. It has presented two results. The first is a condition-based consensus algorithm in which processes decide simultaneously at the end of the round $RS_{t,d,F} = (t+1) - \max(D, \delta)$ where $D \geq 0$ is a value that depends on the actual failure pattern, and $\delta = t - d$ depends on the position of the condition C (the algorithm is instantiated with) in the hierarchy of synchronous legal conditions, namely $C \in \mathcal{S}_t^d$, where the hierarchy is $\mathcal{S}_t^0 \subset \cdots \subset \mathcal{S}_t^d \subset \cdots \subset \mathcal{S}_t^t$ (the set $\mathcal{S}_t^t$ containing the condition made up of all possible input vectors).

The second result is a proof that $RS_{t,d,F}$ is a lower bound on the number of rounds of the simultaneous condition-based consensus problem. This bound shows that we can benefit from the best world provided by the actual run (failure world when $RS_{t,d,F} = (t+1) - D$ or condition world when $RS_{t,d,F} = (t+1) - \delta$), but not from an "addition" of both. There is no double discount for simultaneous condition-based consensus.

References

[1] Attiya H. and Welch J., *Distributed Computing: Fundamentals, Simulations and Advanced Topics (2nd Edition)*, Wiley Interscience, 414 pages, 2004.

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

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

[4] Fischer M.J. and Lynch N., A Lower Bound for the Time to Assure Interactive Consistency. *Information Processing Letters*, 71:183-186, 1982.

[5] Fischer M.J., Lynch N. and Paterson M.S., Impossibility of Distributed Consensus with One Faulty Process. *Journal of the ACM*, 32(2):374-382, 1985.

[6] Friedman R., Mostéfaoui A., S. Rajsbaum S. and Raynal M., Asynchronous Agreement and its Relation with Error-Correcting Codes. *IEEE Transactions on Computers*, 56(7):865-876, 2007.

[7] Garg V.K., *Elements of Distributed Computing*, Wiley, 423 pages, 2002.

[8] Halpern J.Y. and Moses Y., Knowledge and Common Knowledge in a Distributed Environment. *Journal of the ACM*, 37(3):549-587, 1990.

[9] Lynch N.A., Distributed Algorithms. *Morgan Kaufmann Pub.*, San Francisco (CA), 872 pages, 1996.

[10] Mizrahi T. and Moses Y., Continuous Consensus via Common Knowledge. *Distributed Computing*, 20(5):305-321, 2008.

[11] Moses Y. and Rajsbaum S., A Layered Analysis of Consensus. *SIAM J. of Computing*, 31(4):989-1021, 2002.

[12] Moses Y. and Raynal M., Revisiting Simultaneous Consensus with Crash Failures. *Tech Report 1885*, 17 pages, IRISA, Université de Rennes 1, France, 2008. http://hal.inria.fr/inria-00260643/en/

[13] Y. Moses and Tuttle, M. R., Programming Simultaneous Actions Using Common Knowledge. *Algorithmica*, 3:121-169, 1988.

[14] Mostéfaoui A., Rajsbaum S. and Raynal M., Conditions on Input Vectors for Consensus Solvability in Asynchronous Distributed Systems. *Journal of the ACM*, 50(6):922-954, 2003.

[15] Mostéfaoui A., Rajsbaum S. and Raynal M., Synchronous Condition-Based Consensus. *Distributed Computing*, 18(5):325-343, 2006.

[16] Mostefaoui A., Rajsbaum S., Raynal M. and Roy M., Condition-based Consensus Solvability: a Hierarchy of Conditions and Efficient Protocols. *Distributed Computing*, 17:1-20, 2004.

[17] Pease L., Shostak R. and Lamport L., Reaching Agreement in Presence of Faults. *Journal of the ACM*, 27(2):228-234, 1980.

[18] Raynal M., Consensus in Synchronous Systems: a Concise Guided Tour. *Proc. 9th IEEE Pacific Rim Int'l Symposium on Dependable Computing (PRDC'02)*, IEEE Computer Press, pp. 221-228, 2002.

[19] Wang X., Teo Y.M. and Cao J., A Bivalency Proof of the Lower bound for Uniform Consensus. *Information Processing Letters*, 96:167-174, 2005.

A Proof of Theorem 3

Theorem 3 The algorithm described in Figure 1 solves the condition-based consensus problem. Moreover, the processes decide at the end of the round $(t+1)-\delta$.

Proof Let C be the δ-legal condition the algorithm is instantiated with, and $I \in C$ the actual input vector.
Validity. Let us observe that every process p_i that terminates the first round is such that $v\_nocond_i \neq \perp$ and v_nocond_i contains a proposed value. Moreover, if $v\_cond_i \neq \perp$, we have $v\_cond_i = \mathcal{H}(I)$ that (due to Lemma 1 and Definition 2) is a value belonging to I. Then, due to the exchanges of values during the next rounds, v_nocond_i always contains a proposed value, and the same is true for v_cond_i if $v\_cond_i \neq \perp$. It follows that the value decided by a process at line 114 is a proposed value.

Decision and simultaneous decision. The fact that the processes that decide (this set of processes trivially includes all the correct processes), decide during the round $(t+1)-\delta$ follows directly from line 114.

Data agreement. We consider two cases.

- Each process p_i that executes the first round is such that $\#_\perp(V_i) \leq \delta$. In that case, each process computes $v\_cond_i = \mathcal{H}(V_i)$ (line 106). It follows from Lemma 1 and Definition 2 that we have $\mathcal{H}(V_i) = \mathcal{H}(I) = v$ for any process p_i, from which we conclude that $v\_cond_i = v$ for any process p_i that terminates the first round. Consequently, no process that executes line 115 decides a value different from v.
- One process p_i that executes the first round is such that $\#_\perp(V_i) > \delta$. This means that at least $\delta + 1 = t - d + 1$ processes crashed during the first round. This means that at most $d-1$ processes can crash between the end of the first round and the end of round $r = (t+1) - \delta = d+1$, i.e. during a period of d rounds. It follows that there is a clean round r' such that $2 \leq r' \leq (t+1) - \delta = d+1$. Due to Property 1, all the processes that execute round r' receive the same set of messages. Consequently, at the end of r', all these processes have the same value v' (v' can possibly be $\perp$) in their local variable v_cond_i, and the same value v'' (with $v'' \neq \perp$) in their local variable v_nocond_i, from which follows that, in that case, no two processes can decide different values.

$\square_{Theorem\ 3}$

B Proof of Theorem 5

This appendix proves the (non-condition-based) simultaneous consensus algorithm described in Figure 2 (Section 3.2) is correct.

Lemma 2 Validity property. *A decided value is a proposed value.*

Proof The proof is an immediate consequence of the initialization of the est_i local variables (line 201), the reliability of the channels, and the $\min()$ operation used at line 207. $\square_{Lemma\ 2}$

Lemma 3 *Let p_i be a correct process. $\forall r \geq 0$ we have $h_i(r) \geq r$.*

Proof Since the processes in the set $f'_i[r-1]$ are processes that have crashed by the end of the round $r-1$, it follows that $t - |f'_i[r-1]| \geq 0$. Consequently, $h_i(r) = r + t - |f'_i[r-1]| \geq r$. $\square_{Lemma\ 3}$

Notation: Considering an arbitrary execution, let p_i be a process that is correct in that execution.

- Let $BH_i = \min_{r \geq 0} h_i(r)$. BH_i is the smallest value ever attained by the function $h_i(r)$, i.e., the smallest horizon value determined by p_i.
- Let $L_i = \max(\{r \mid h_i(r) = BH_i\})$. L_i is the last round whose horizon value is BH_i.

It follows from these definitions that if $L' > L_i$ then $h_i(L') > h_i(L_i)$.

Irisa

Lemma 4 *Let $t < n$. The round L_i is a clean round (i.e., no process is discovered faulty for the first time in that round).*

Proof Assume, by way of contradiction, that L_i is not clean (recall that p_i is a correct process). This means there is a process p_z that is seen faulty for the first time in round L_i by some process p_y. Notice that $p_z \notin f'_i[L_i - 1]$ since p_z was not discovered faulty in the previous rounds. There are two cases.

- Case 1: p_i receives a message from p_y in round $L_i + 1$.
  (This case includes the case where p_i and p_y are the same process). As p_y does not receive a message from p_z during L_i, and a crash is stable, we have $p_z \in f_y[L_i]$. Moreover, due to the case assumption, and the fact that the round $L_i + 1$ message from p_y to p_i carries $f_y[L_i]$, it follows that $f'_i[L_i]$ contains $f'_i[L_i - 1] \cup \{p_z\}$. Consequently, $|f'_i[L_i]| > |f'_i[L_i - 1]|$. It follows that $h_i(L_i + 1) \leq h_i(L_i)$, contradicting the definition of L_i.
- Case 2: p_i does not receive a message from p_y in round $L_i + 1$.
  In that case, both p_z and p_y are seen faulty for the first time by p_i during the round $L_i + 1$. So, $f_i[L_i + 1]$ contains $f'_i[L_i - 1] \cup \{p_y, p_z\}$. Since $f'_i[L_i + 1]$ (computed by p_i during the round $L_i + 2$) contains $f_i[L_i + 1]$, we have $|f'_i[L_i + 1]| \geq |f'_i[L_i - 1]| + 2$. Thus, we have

$$\begin{array}{rcl} h_i(L_i + 2) & = & (L_i + 2) + t - |f'_i[L_i + 1]|, \\ & \leq & (L_i + 2) + t - (|f'_i[L_i - 1]| + 2), \\ & = & L_i + t - |f'_i[L_i - 1]|, \\ & = & h_i(L_i), \end{array}$$

  which again contradicts the definition of L_i.

$\square_{Lemma\ 4}$

Lemma 5 *Let $t < n$. Every correct process decides. Moreover, all processes that decide do so in the same round and decide on the same value.*

Proof

Decision property. Let us consider a correct process p_i. Notice that, due to the initialization and line 209 we have $\forall r : bh_i[r] \leq t + 1$, from which we conclude $BH_i \leq t + 1$. So, to prove that p_i decides we have to show that p_i does not miss the test $r = BH_i$ at line 210. This could happen if the first round ℓ such that $bh_i[\ell - 1] > BH_i$ and $bh_i[\ell] = BH_i$ is such that $\ell > BH_i$. We prove that this cannot happen.

Let us observe that, due to Lemma 3, we have $h_i(\ell) \geq \ell$. It then follows from $bh_i[\ell - 1] > BH_i$, $h_i(\ell) \geq \ell$, $bh_i[\ell] = BH_i$, and line 209, that $BH_i = bh_i[\ell] = \min(bh_i[\ell - 1], h_i(\ell)) = h_i(\ell) \geq \ell$, i.e., $BH_i \geq \ell$, which establishes the result. It follows that p_i decides no later than round $t + 1$.

Simultaneous decision for the correct processes. We first show that no two correct processes p_i and p_j decide at distinct rounds. Due to the algorithm, if p_i and p_j decide, they decide at round BH_i and BH_j, respectively. We show that $BH_i = BH_j$. Due to Lemma 4, the round L_i is clean. Hence, during the round L_i, p_j receives the same messages that p_i receives (Property 1). Thus $f'_i[L_i - 1] = f'_j[L_i - 1]$ and consequently, $h_i(L_i) = h_j(L_i)$. Since $bh_j[L_i] \leq h_j(L_i)$ by line 09, it follows that $bh_j[L_i] \leq bh_i[L_i] = BH_i$, and thus $BH_j \leq BH_i$. By symmetry the same reasoning yields $BH_i \leq BH_j$, from which it follows that $BH_i = BH_j$. This proves that no two correct processes decide at distinct rounds.[4]

Simultaneous decision for the faulty processes. BH being the round at which the correct processes decide, let us now consider the case of a faulty process p_j. As p_j behaves as a correct process until it crashes, and as the correct processes decide in the same round BH, it follows that no faulty process decides before BH, and if p_j executes line 210 of round BH, it does decide as if it was a correct process.

[4] In [3, 8] it is shown that before performing such a simultaneous action the processes must attain common knowledge that they are doing so. In particular, they must have common knowledge that the decided value v is one of the initial values in the run.

Data agreement property. The fact that no two processes decide different values comes from the existence of the clean round L_i that appears before a process decision. During that round, all the processes that are alive at the end of this round have received the same set of estimate values (Property 1), and selected the smallest of them. It follows that, from the end of that round, there is a single estimate value in the system, which proves the data agreement property. $\square_{Lemma\ 5}$

Theorem 5 Let $t < n$. The algorithm described in Figure 2 solves the consensus problem with simultaneous decision. In a run with failure pattern F, decision is reached in round $t + 1 - D$ where $D = D(F)$ is the waste inherent in F.

Proof The proof of the validity, decision, simultaneous decision and data agreement properties follow from the Lemmas 2 and 5. We now show that the decision is obtained in round $t + 1 - D$. Let us consider an arbitrary run of the algorithm. It follows from Lemma 5 that $BH_i = BH_j$ for any pair of processes p_i and p_j that decide. Let BH denote this round. The proof of the claim amounts to showing that $BH \leq t + 1 - D$ and $BH \geq t + 1 - D$.

Let p_i be a process that decides and R the last round such that $|C[R]| - R = D$ (i.e., $|C[R + x]| - (R + x) < D = |C[R]| - R$, for any $x > 0$). Let us observe that, due to the lines 208-210 of the algorithm, BH is attained at the round numbers that make the function $h_i()$ minimal. Moreover, it follows from the definition of D and R that $|C[R + 1]| \leq |C[R]|$. Since $C[R] \subseteq C[R + 1]$, it follows that $C[R] = C[R + 1]$, i.e., no new process failure is discovered in round $R + 1$, so the round $R + 1$ is clean and we have $|f'_i[R]| = |C[R]|$. Due to line 208 of the round $R + 1$, we have $h_i(R + 1) = R + t + 1 - |f'_i[R]| = (t + 1) - (|f'_i[R]| - R) = t + 1 - D$, from which we conclude $BH \leq t + 1 - D$.

For the other direction, let us recall that, due to Lemma 4, the round $L_i > 0$ is clean. It follows that $f'_i[L_i - 1] = C[L_i - 1]$, since any p_i hears in round L_i from all processes that survived round $L_i - 1$. Therefore, $BH = t + 1 - (|f'_i[L_i - 1]| - (L_i - 1)) = t + 1 - (|C[L_i - 1]| - (L_i - 1)) = t + 1 - d_{(L_i-1)} \geq t + 1 - D$, which completes the proof of the theorem. $\square_{Theorem\ 5}$

C On the optimality of the algorithm: $t + 1 - \min(D, \delta)$ is a lower bound

This part is an essential part of the paper: it proves that the algorithm described in Section 3.4 (called PROPOSE in the following) is optimal, namely, in a synchronous system prone to up to t process crashes (with $t < n - 1$), there is no deterministic algorithm that can ever solve the simultaneous condition-based consensus problem in fewer than $(t + 1) - \max(D, \delta)$ rounds. The proof relies on notions introduced in [3, 10, 12]. It also uses notations introduced in Section 3.3.

The problem of simultaneous consensus is closely related to the knowledge-theoretic notion of similarity among runs at a given time. This notion is captured by the following definitions. For later use, these definitions are made with respect to an arbitrary round-based synchronous deterministic algorithm P (they consequently apply in particular to the PROPOSE algorithm).

C.1 Preliminary definitions and lemmas

For ease of exposition, the runs of an arbitrary deterministic algorithm P are denoted by σ, σ', etc. $S[r, \sigma]$ denotes the set of processes that survive round r of σ, while $ls(p, r, \sigma)$ denotes the local state of p at the end of r in the run σ (i.e., its set of local variables and their current values).

Definition 4 *Given a deterministic algorithm P, a process p, and a round r,* the runs σ and σ' of P are indistinguishable to p after round r *(denoted $\sigma \overset{r}{\sim}_p \sigma'$) if both (i) $p \in S[r, \sigma] \cap S[r, \sigma']$ (i.e., p has survived round r in both runs), and (ii) $ls(p, r, \sigma) = ls(p, r, \sigma')$.*

Definition 5 *The runs σ and σ' are* connected *at the end of round r, denoted $\sigma \overset{r}{\approx} \sigma'$, if there is a sequence of runs and processes such that $\sigma = \sigma_0 \overset{r}{\sim}_{p_0} \sigma_1 \overset{r}{\sim}_{p_1} \cdots \overset{r}{\sim}_{p_{k-1}} \sigma_k = \sigma'$.*

In other words, an undirected graph $\mathsf{G}(P, r)$ (in short $\mathsf{G}(r)$) can be associated with each round r of a protocol P. This graph is called P's *similarity graph* for round r. Its vertices are the runs of P and there is an edge connecting

σ and σ' if there is a process q such that $\sigma \overset{r}{\sim}_q \sigma'$. It is easy to see that $\sigma \overset{r}{\approx} \sigma'$ holds if σ and σ' belong to the same connected component in $\mathsf{G}(r)$. Because $\mathsf{G}(r)$ is undirected, being connected ($\overset{r}{\approx}$) is an equivalence relation.[5] Moreover, observe that if we can show that some property A is maintained under $\overset{r}{\sim}_q$ for all $q \in \Pi$, then whenever σ has property A and $\sigma \overset{r}{\approx} \sigma'$, we are guaranteed that σ' has property A as well.

Lemma 6 *Let σ and σ' be runs of a deterministic algorithm P that solves simultaneous consensus. If some process decides on value v in round r of σ and $\sigma \overset{r}{\approx} \sigma$, then the processes in $S[r, \sigma']$ decide the same value v in the same round r of σ'.*

Proof It suffices to show the claim for any two runs σ, σ' such that $\sigma \overset{r}{\sim}_q \sigma'$ for some $q \in S[r, \sigma] \cap S[r, \sigma']$. In this case, q decides v in round r of σ. Because P is deterministic and q has the same local state at the end of round r of σ and σ', q decides v at the end of round r in σ'. Finally, as P solves simultaneous consensus (lemma assumption) it follows that all processes in $S[r, \sigma']$ decide v in round r (and no other process decides a different value in a different round). $\square_{Lemma\ 6}$

An immediate consequence of Lemma 6 is captured by the following corollary.

Corollary 1 *Let P be a deterministic algorithm that solves simultaneous consensus. If σ' is a run of P such that (1) no initial value in σ' is v, and (2) $\sigma \overset{r}{\approx} \sigma'$, then no process can decide v in round r of σ.*

Proof Since $n > t$, the set $S[r, \sigma']$ is nonempty. By Lemma 6, if some process q decides v in round r of σ, the processes in $S[r, \sigma']$ decide v in round r of σ'. But this contradicts the Validity property of the simultaneous consensus algorithm P, since the decision value v is not one of the initial values in σ'. $\square_{Corollary\ 1}$

C.2 A full-information algorithm

For the purpose of proving optimality, we make use of a *full-information* algorithm, denoted FIP and described in Figure 3.

The algorithm In the first round, each process sends its initial value v_i to all processes (including to itself). The algorithm then constructs an array $\mathsf{Inp}_i[1..n]$ containing the incoming message from each of the processes (itself included). If p_i does not receive a message from p_j then it sets $\mathsf{Inp}_i[j]$ to the default value $\perp$. In each of the later rounds, every process p_i first sends Inp_i to all others, and then uses the incoming messages of the current round to construct an updated array Inp_i in the same way as in the first round. The local state of the process at the end of round r is identified simply with the contents of its array Inp_i.

This algorithm is introduced in order to establish, for each failure pattern F, times at which simultaneous consensus cannot be attained by any algorithm whatsoever. Optimality will then be established by showing that the PROPOSE algorithm decides as soon as possible, for each and every possible failure pattern (and initial configuration).

```
algorithm FIP:
(01)  for j ∈ {1, ..., n} \ {i} do Inp_i[j] ← ⊥ end for; Inp_i[i] ← v_i;
(02)  when r = 1, 2, ... do
(03)  begin round
(04)      send Inp_i to all;      % including to p_i itself %
(05)      for j ∈ {1, ..., n} do
(06)          Inp_i[j] ← message received from p_j during r if any, otherwise ⊥
(07)      end for
(08)  end round
```

Figure 3: The full-information algorithm FIP (code for p_i)

[5] In the knowledge terminology, process *q knows* a fact A at the end of round r in σ if it is true of all runs σ' satisfying $\sigma' \overset{r}{\sim}_q \sigma$; it is *common knowledge* there if A holds at all runs $\sigma' \overset{r}{\approx} \sigma$. Thus, the set of runs connected to σ determines what is common knowledge in σ at the end of round r.

Observe that a deterministic algorithm P, an initial configuration I (set of initial values), and a failure pattern F determine a run $\sigma = P(I, F)$ of P.

Definition 6 *A run σ of P* corresponds to *a run ρ of an algorithm P' if, for some initial configuration I and failure pattern F, it is the case that $\sigma = P(I, F)$ and $\rho = P'(I, F)$.*

The next lemma captures, in a precise sense, the fact that the connected components of the similarity graph for FIP refine those of any other deterministic algorithm.[6] This lemma is from [12]. For completeness, its proof is given in appendix D.

Lemma 7 *Let P be a deterministic algorithm for simultaneous consensus. Let us assume that the runs σ and σ' of P correspond to the runs ρ and ρ' of* FIP*, respectively. Then (i) if $\rho \overset{r}{\sim}_q \rho'$ then $\sigma \overset{r}{\sim}_q \sigma'$, and (ii) if $\rho \overset{r}{\approx} \rho'$ then $\sigma \overset{r}{\approx} \sigma'$.*

On failure patterns and full-information algorithms Observe that in both the FIP and PROPOSE algorithms, a correct process is required to send a message to each process in every round. As a result, in runs of both FIP and PROPOSE, a process p knows by the end of round r that q has crashed if the failure pattern F is such that q has crashed before it sent its round r message to p.

The set $f_i[r]$ of the processes that r has not heard from in round r can be directly computed from the local state in FIP as $\{p_j | \mathsf{Inp}_i[j] = \perp\}$. Since p_i sends Inp_i to all other processes, the set $f'_i[r-1]$ of processes that p_i knows at r to have been discovered as crashed by round $r-1$ is thus easily computed from the messages it receives in round r. Since for corresponding runs of FIP and PROPOSE these sets coincide (in fact, their values depend only on the failure pattern), we find it convenient to talk about the values of $f_i[r, F]$, $f'_i[r, F]$, $C[r, F]$, $D[F]$, etc. for runs of FIP as well.

Since the Validity property states that it is illegal to decide v in a run that does not contain v as one of its initial values, Corollary 1 implies that it is impossible to decide on v as long as there is a connected run that does not contain v as one of its initial values. In light of this, we can now show that the algorithm FIP reaches a decision as soon as it possibly can.

C.3 Premature rounds

It has been shown in Theorem 5 that the algorithm PROPOSE decides on a value in round $BH = t+1-D$. Let $BH(\rho)$ denote the value of the round number BH of the the run $\rho = \text{PROPOSE}(I, F)$. Recall that the value of BH is solely a function of ρ's failure pattern F (and not of the initial configuration).

Definition 7 *A round ℓ is* premature[7] *in F if $\ell < t + 1 - D = BH(\rho)$ for every run $\rho = \text{PROPOSE}(I, F)$.*

As $h_i(r+1) = r + (t + 1 - |f'_i[r]|)$, and $h_i(r+1) \geq BH(\rho)$, this means that, for every r such that $r + 1 \leq \ell$, the property $r + t + 1 - |C[r, F]| > \ell$ holds. Notice that the failure pattern F that occurs during the run ρ determines whether or not ℓ is premature: In all runs of PROPOSE with the same F the sets $f_i[r, F]$ and $C[r, F]$ are the same for every i and r, and so are D and BH.

Lemma 8 *Let $t < n - 1$, $\ell \geq 0$, $\rho = \text{FIP}(I, F)$ and $\rho' = \text{FIP}(I, F')$. If $\rho \overset{\ell}{\approx} \rho'$ then ℓ is premature in F iff ℓ is premature in F'.*

Proof Let $\rho = \text{FIP}(I, F)$ and $\rho' = \text{FIP}(I', F')$. As in the proof of item (ii) of Lemma 7, it suffices to show that, for all q, if $\rho \overset{\ell}{\sim}_q \rho'$ then ℓ is premature in F iff ℓ is premature in F'. Thus, let us assume that $\rho \overset{\ell}{\sim}_q \rho'$. Let $\sigma = \text{PROPOSE}(I, F)$ and $\sigma' = \text{PROPOSE}(I', F')$ be the runs of PROPOSE corresponding to the runs $\rho = \text{FIP}(I, F)$ and $\rho' = \text{FIP}(I', F')$, respectively.

By item (ii) of Lemma 7, it follows that $\sigma \overset{\ell}{\sim}_q \sigma'$. Round ℓ is premature in F iff $BH(\sigma) > \ell$, which by Theorem 5 implies that q does not decide in σ by the end of round ℓ. Thus, since the **if** test on line 114* of PROPOSE fails, $decided_q = false$ continues to hold in σ. The fact that $\sigma \overset{\ell}{\sim}_q \sigma'$ implies that $decided_q = false$ holds at the end of the

[6] In the sequel, we interpret $\overset{r}{\sim}$ and $\overset{r}{\approx}$ among runs of an algorithm P in terms of the similarity graph $\mathsf{G}(r, P)$ defined on the runs of P. Thus, the interpretation of $\overset{r}{\sim}$ and $\overset{r}{\approx}$ in statements such as that of Lemma 7 is always with respect to the algorithm generating the related runs.

[7] This is short for *premature for simultaneous consensus*, a term that will be justified by the technical analysis in this section.

round ℓ of σ' as well. It follows that $BH(\sigma') > \ell$, and consequently ℓ is premature at F', as desired. The 'only-if' direction of the lemma is obtained by a symmetric argument. $\Box_{Lemma\ 8}$

Definition 8 *A process is* silent *in a round r of a run ρ if it has crashed before sending its round r messages.*

Definition 9 *Given a failure pattern F, a process q and a round r, let $F_{q,r}$ be the failure pattern that satisfies the following four conditions: (i) $F_{q,r}$ coincides with F for the first $r-1$ rounds, (ii) in round r exactly the failures detected in $C[r,F]$ occur in $F_{q,r}$, (iii) process q is silent from round $r+1$ on, and (iv) no process other than q fails after round r.*

Let us remark that if the first k rounds of both F and F' are the same, then $F_{q,k} = F'_{q,k}$. Since the rounds $r \leq 0$ of all failure patterns are the same, *we have that* $F_{q,0} = F'_{q,0}$ for all F and F'.[8] Indeed, $F_{q,0}$ is the failure pattern in which process q is silent and sends no messages whatsoever, while no other process crashes. In runs with such a pattern, the execution cannot depend on q's initial value.

Lemma 9 *If ℓ is premature in F and $k < \ell$, then no more than t processes crash in $F_{q,k}$.*

Proof Let $H[r,F] = r + t + 1 - |C[r,F]|$. Let us observe that the number of processes that crash in $F_{q,k}$ is at most $|C[k,F]| + 1$. It suffices to show that $|C[k,F]| < t$. As ℓ is premature and $k < \ell$, we have $H[k,F] = k + t + 1 - |C[k,F]| > \ell$. Since $k < \ell$ we have that $\ell \geq k+1$. We thus obtain that $t + k + 1 - |C[k,F]| > k + 1$, which implies that $t - |C[k,F]| > 0$, and $t \geq |C[k,F]| + 1$, as desired. $\Box_{Lemma\ 9}$

Definition 10 *Given a run $\rho = \text{FIP}(I,F)$, let $\rho_{q,k}$ be the run $\rho_{q,k} = \text{FIP}(I,F_{q,k})$.*

Let us notice that, due to Lemma 9, if ρ is a run of FIP in which at most t processes fail, then so is $\rho_{q,k}$.

The following lemma is from [12]. Its states an equivalence on classes of runs that is used to prove optimality (see Lemma 11 in the next section).

Lemma 10 *Let $t < n-1$ and fix $\ell > 0$. Moreover, let $\rho = \text{FIP}(I,F)$ and let $q \in \Pi$. If ℓ is premature in F, then $\rho \overset{\ell}{\approx} \rho_{q,k}$ for all k satisfying $0 \leq k \leq \ell - 1$.*

Proof Let $\ell > 0$. We prove the lemma for all runs ρ, by induction on $s = \ell - k$. For the base case, assume that $s = 1$, and so $k = \ell - 1$. Choose $q \in \Pi$, and let $p \in S[\ell,\rho]$. Since ℓ is premature in ρ, we have $h_p(\ell) = (\ell-1) + t + 1 - |f_p[\ell-1,F]| > \ell$, i.e., $|f_p[\ell-1,F]| \leq t-1 \leq n-3$. Let $p' \in S[\ell,\rho] \setminus \{p\}$ (such a process p' is guaranteed to exist since by assumption $t \leq n-2$).

Let ρ' be the run that is identical to ρ up to and including round $\ell-1$, where in round ℓ process p receives the same messages as in ρ, but process p' (who is non-faulty in ρ' too) receives messages from all processes in $\Pi \setminus f_p[\ell-1,F]$. Finally, no process other than those in $f_p[\ell-1,F]$ fails in ρ'. There are exactly $|f_p[\ell-1,F]| < t$ failures in ρ' and $p \in S[\ell,\rho']$ has the same local state at the end of round ℓ in both ρ and ρ'. Hence, we have $\rho \overset{\ell}{\sim}_p \rho'$.

If q is silent in round ℓ in ρ' then we are done. Otherwise, let ρ'' be a run that is the same as ρ' except that q crashes in round ℓ by not sending a round ℓ message to p. At most $|f_p[\ell-1,F]| + 1 \leq t - 1 + 1 = t$ processes fail in ρ'', and p' has the same state in ρ' as in ρ''. Hence, $\rho' \overset{\ell}{\sim}_{p'} \rho''$. Finally, observe that $\rho_{q,\ell-1}$ is identical to ρ'' except that q is silent in round ℓ. Process p does not distinguish ρ'' from $\rho_{q,\ell-1}$ since in both it receives the same messages in round ℓ. Thus, $\rho'' \overset{\ell}{\sim}_p \rho_{q,\ell-1}$, and by definition of $\overset{\ell}{\approx}$ we have that $\rho \overset{\ell}{\approx} \rho_{q,\ell-1}$, completing the base case.

Induction step. Let $s = \ell - k > 1$ and assume that the claim holds for round $s-1$ (i.e., round $k+1$) in all runs in which ℓ is premature. We prove the claim for round k. Let ρ be a run in which ℓ is premature, and choose an arbitrary process $q \in \Pi$. We will use the inductive assumption to find a connected run ρ' that coincides with ρ for the first k

[8] The proof of some lemmas that follow uses the fictitious round $r = 0$. This motivates the definition $C[-1,F] = C[r,F] = 0$ introduced in connection with Definition 3. This allows us to define failure patterns and associated runs in which "we crash" a priori a given process q, without being bothered by the processes that crash in the first round of F before they send their round 1 messages.

rounds where no process crashes in round $k+1$. We then use the inductive assumption again to show that $\rho' \overset{\ell}{\approx} \rho_{q,k}$, and obtain by transitivity that $\rho \overset{\ell}{\approx} \rho_{q,k}$, as desired.

Let ρ' be a run that coincides with ρ for the first k rounds where no process crashes in round $k+1$, and process p_n is silent from round $k+2$ on. We show that $\rho \overset{\ell}{\approx} \rho'$. Define $\rho^1, \ldots, \rho^n$ where $n = |\Pi|$ to be runs such that in ρ^j the first k rounds are identical to ρ, process p_j is silent from round $k+2$ on, no process other than (possibly) p_j fails in rounds $k+2, \ldots, \ell$, and no new failure in round $k+1$ is seen by processes $p_1, \ldots, p_j$. Denoting $\rho = \rho^0$, we prove by induction on j that $\rho \overset{\ell}{\approx} \rho^j$. The case $j = 0$ is immediate, since $\rho^0 = \rho$. Let $j > 0$ and assume inductively that $\rho \overset{\ell}{\approx} \rho^{j-1}$. Since ℓ is premature in $F = F(\rho)$, it follows from Lemma 8 that ℓ is premature in ρ^{j-1}. By the inductive assumption for $k+1$ we have that $\rho^{j-1} \overset{\ell}{\approx} \widehat{\rho}$, where $\widehat{\rho} = (\rho^{j-1})_{p_j,k+1}$. In particular, Lemma 9 implies that there are at most t failures in $\widehat{\rho}$. Observe that (i) p_j is silent from round $k+2$ in $\widehat{\rho}$, (ii) every process other than p_j has the same local state at the end of round k in both $\widehat{\rho}$ and ρ^j, (iii) the same messages are sent in both runs from round $k+2$ on, and (iv) since no more processes fail in ρ^j than do in $\widehat{\rho}$, there are at most t failures in ρ^j. It thus follows that $\widehat{\rho} \overset{\ell}{\approx} \rho^j$. Moreover, since $\rho \overset{\ell}{\approx} \rho^{j-1} \overset{\ell}{\approx} \widehat{\rho} \overset{\ell}{\approx} \rho^j$, we have by transitivity of $\overset{\ell}{\approx}$ that $\rho \overset{\ell}{\approx} \rho^j$, completing the inductive step. We conclude that $\rho \overset{\ell}{\approx} \rho^n = \rho'$, as claimed. Notice that since $\rho \overset{\ell}{\approx} \rho'$ and ℓ is premature in ρ it follows that ℓ is also premature in ρ'.

Our goal now is to "silence" process q from round $k+1$ on. Let $\rho'' = \text{FIP}(I, F'')$ where F'' agrees with the pattern F' of ρ' on everything, except that in F'' process q is silent from round $k+2$ on. Thus, in ρ'' the first k rounds are identical to ρ, no process fails in round $k+1$, and processes p_n and q are silent from round $k+2$ on. We first claim that at most t processes fail in F''. Since ℓ is premature in ρ' we have that $t+1-D(F') > \ell$ and hence $t - D \geq \ell$. Let h be the number of process failures up to round k, inclusive, in ρ' (and hence also in ρ''). Since no failures are detected in round $k+1$ of ρ' we have that $d_k = h - k$. As $D \geq d_k$ we have that $t - (h-k) \geq \ell$. It follows that $t - h \geq \ell - k = s \geq 2$. We conclude that $t \geq h + 2$. The claim now follows since, by definition, F'' contains at most $h+2$ failures, and we have just shown that $h + 2 \leq t$.

The values of $d_1, \ldots, d_k, d_{k+1}$ (defined in Definition 3) are the same in ρ' and in ρ''. Since no failure is detected in round $k+1$ of either, we have that $d_{k+1} = d_k - 1$. In ρ'' we have that $d_{k+2} = d_k$ and $d_{k'} < d_{k+2}$ for all rounds $k' > k+2$. It follows that $D(F'') = D(F')$. Since ℓ is premature in ρ' we have that $t+1-D(F'') = t+1-D(F') > \ell$, and ℓ is premature in ρ'' as well. Observe that $\rho' = \rho''_{p_n,k+1}$. By the inductive hypothesis for $k+1$ we have that $\rho' \overset{\ell}{\approx} \rho''$. Let $\bar{\rho} = \text{FIP}(I, \bar{F})$, where $\bar{F}$ differs from F'' only in that p_n does not receive a message from q in round $k+1$. Let $p \in S[\ell, F''] = S[\ell, \bar{F}]$. Since p_n is silenced from round $k+2 \leq \ell$ in ρ'' and in $\bar{F}$, we have that $\rho'' \overset{\ell}{\sim}_p \bar{\rho}$ and thus $\rho \overset{\ell}{\approx} \rho'' \overset{\ell}{\approx} \bar{\rho}$. Now define runs $\sigma^1, \ldots, \sigma^{n-1}, \sigma^n$ where σ^j coincides with ρ on the first k rounds, no process other than q fails in round $k+1$, in round $k+1$ processes $p_j, \ldots, p_n$ do not receive a message from q, ($p_1, \ldots, p_{j-1}$ receive round $k+1$ messages from q iff they do so in ρ), and process p_j is silent from round $k+2$ on. Notice that $\bar{\rho} = \sigma^n$. An inductive proof identical to the one for $\rho^1, \ldots, \rho^n$ above (this time moving down from σ^n to σ^1) shows that $\rho \overset{\ell}{\approx} \sigma^1$. In σ^1 process q is silenced from round $k+1$ on, since no process receives a message from it in this round. Moreover, it is easy to check that $\sigma^1_{q,k} = \rho_{q,k}$. Since ℓ is premature inσ^1 we obtain that $\rho \overset{\ell}{\approx} \sigma^1 \overset{\ell}{\approx} \sigma^1_{q,k} = \rho_{q,k}$, and we are done. $\square_{Lemma\ 10}$

C.4 Optimality of the proposed algorithm

Definition 11 *C being a legal condition, let the x-graph over C be the graph $\mathcal{G}_x = (C, E_x)$ where $E_x = \{(I, I') : dist(I, I') \leq x$.*

Definition 12 *The condition C is x-*coverable *if for every vector $I \in C$ there are at least two values $v, w \in \mathcal{V}$ such that there are paths in $\mathcal{G}_x$ from I to both the input vector $I^v = (v, v, \cdots, v)$ and the input vector $I^w = (w, \ldots, w)$.*

Lemma 11 *Let $t < n-1$ and let $\rho = \text{FIP}(I, F)$. Let C be a legal condition such that $I, I' \in C$, where I is connected in $G_x(C)$ to I'. If ℓ is a premature round in F and $\ell \leq t+1-x$, then there is a run $\rho' = \text{FIP}(I', F')$, such that $\rho \overset{\ell}{\approx} \rho'$.*

Proof Denote by s the distance between I and I' in $\mathcal{G}_x(C, E_x)$. Notice that $0 \leq s < \infty$ (because by assumption I is connected to I' in $G_x(C, E_x)$, and so there is a finite path connecting I to I'). We prove the claim by induction on s. The base case is $s = 0$, for which the claim trivially holds for the choice $\rho' = \rho$ since we have then $I = I'$, $F = F'$, and because $t < n$ guarantees that $\rho \overset{\ell}{\approx} \rho$ for all runs.

For the inductive step, let $s > 0$ and assume that the claim holds for all runs with shortest $\mathcal{G}_x(C, E_x)$ distance $s - 1$ to I'. Since the distance between I and I' is $s > 0$, there is an initial vector $\widehat{I}$ whose distance from I in $\mathcal{G}_x(C, E_x)$ is 1 and whose distance from I' is $s - 1$. We first claim that $\rho \overset{\ell}{\approx} \widehat{\rho}$ for a run of the form $\widehat{\rho} = \textsc{Fip}(\widehat{I}, \widehat{F})$. Let $T = \{q_1, \ldots, q_x\}$ be a set of processes such that process p's initial value is the same in I and in $\widehat{I}$ for all $p \in \Pi \setminus T$. Denote by ρ^1 the run $\rho^1 = \textsc{Fip}(I, F_{q_1,0})$, which has initial vector I and in which q_1 crashes in round 0, , and no other process fails.

Since, by assumption, ℓ is premature in $\rho = \rho^0$, we have by Lemma 10 that $\rho \overset{\ell}{\approx} \textsc{Fip}(I, F_{q_1,0}) = \rho^1$. Let F^x be the failure pattern, and ρ^x the associated run, in which the x processes in T are silent from the start, and no other process fails. It is easy to check that $D(\rho^x) = x - 1$ since $d_r(\rho^x \leq x - 1$ for all rounds $r \geq 0$. Thus, $t + 1 - D = t + 2 - x$ in ρ^x. Since, by assumption, $\ell \leq t + 1 - x$, we have that ℓ is premature in ρ^x. Notice that $F^x_{q_1,0} = F_{q_1,0}$. Thus, using again Lemma 10, we have that $\rho^x \approx \textsc{Fip}(I, F_{q_1,0}) = \rho^1$. By symmetry and transitivity of $\overset{\ell}{\approx}$ it follows that $\rho \overset{\ell}{\approx} \rho^x$. Define $\widehat{\rho} = \textsc{Fip}(\widehat{I}, F^x)$. By assumption, $\widehat{I} \in C$, and so $\widehat{\rho}$ is a run of Fip under condition C. Since $t < n$ we have that F^x contains at least one correct process, which we denote w.l.o.g by p. Because ρ^x and $\widehat{\rho}$ differ only in the initial values of processes in T that are silent throughout F^x, and since $p \in S[\rho^x, \ell] \cap S[\widehat{\rho}, \ell]$, we have that $\rho^x \overset{\ell}{\sim}_p \widehat{\rho}$. It follows that $\rho^x \overset{\ell}{\approx} \widehat{\rho}$. Since $\rho \overset{\ell}{\approx} \rho^x$ we have by transitivity of $\overset{\ell}{\approx}$ that $\rho \overset{\ell}{\approx} \widehat{\rho}$.

Finally, since the distance between $\widehat{I}$ and I' in G_x is $s - 1$, we have by the inductive hypothesis that $\widehat{\rho} \overset{\ell}{\approx} \rho'$ for some run ρ' of the form $\rho' = \textsc{Fip}(I', F')$. As we have shown, $\rho \overset{\ell}{\approx} \widehat{\rho}$, and it follows by transitivity of $\overset{\ell}{\approx}$ that $\rho \overset{\ell}{\approx} \rho'$, which completes the inductive step. This completes the inductive step for s, and the lemma follows. $\square_{Lemma\ 11}$

The next corollary follows directly from Lemma 11.

Corollary 2 *Let $t < n - 1$ and let C be a legal x-coverable condition. Let $\rho = \textsc{Fip}(I, F)$, where $I \in C$, ℓ is premature in F, and $\ell \leq t + 1 - x$. Assume that I is connected in $\mathcal{G}_x$ to both the input vector $I^v = (v, \ldots, v)$ and the input vector vector $I^w = (w, \ldots, w)$. Then, $\rho^v \overset{\ell}{\approx} \rho \overset{\ell}{\approx} \rho^w$, where ρ^v and ρ^w are runs of* Fip *with input vectors I^v and I^w, respectively.*

Finally, the next theorem follows from the previous corollary.

Theorem 7 *Let P be a deterministic algorithm solving simultaneous condition-based consensus for condition C. Assume that C is x-coverable. Then, in no run of P can decision be reached before round $RS_{t,d,F} = \min(t + 1 - D, t + 1 - x)$.*

D Proof of Lemma 7

Lemma 7 Let P be a deterministic algorithm for simultaneous consensus. Let us assume that the runs σ and σ' of P correspond to the runs ρ and ρ' of Fip, respectively. Then (i) if $\rho \overset{r}{\sim}_q \rho'$ then $\sigma \overset{r}{\sim}_q \sigma'$, and (ii) if $\rho \overset{r}{\approx} \rho'$ then $\sigma \overset{r}{\approx} \sigma'$.

Proof Let us consider the runs σ and σ' of P and the corresponding runs ρ and ρ' of Fip. Since $\overset{r}{\approx}$ is the transitive closure of the relations $\{\overset{r}{\sim}_p\}_{p \in \Pi}$, claim (ii) follows from (i). Consequently, it suffices to prove claim (i). Let us observe that, due to the definition of "corresponds to", the fact that σ corresponds to ρ implies that $S[r, \sigma] = S[r, \rho]$ for all rounds r. The proof that $\rho \overset{r}{\sim}_q \rho' \Rightarrow \sigma \overset{r}{\sim}_q \sigma'$, is by induction on r.

Base case. For the base case, let us consider the initial configuration that corresponds to the fictitious round $r = 0$. Since the initial state of each process (under P as well as under Fip) is fully determined by its initial value, the fact that $\rho \overset{0}{\sim}_q \rho'$ implies that q has the same initial state in both. Since σ corresponds to ρ and σ' corresponds to ρ', it

follows that q has the same initial value in σ and σ', and so $\sigma \overset{0}{\sim}_q \sigma'$.

Induction case: $r > 0$. Let us assume that item (i) holds for every process at round $r-1$ (induction assumption). Moreover, let us assume that $\rho \overset{r}{\sim}_q \rho'$. As before, σ and σ' correspond to ρ and ρ', respectively. We need to show that $\sigma \overset{r}{\sim}_q \sigma'$.

Since q survives round r in both the runs ρ and ρ', and as this depends only on their failure patterns F and F', which are also the failure patterns in σ and σ', respectively, we have $q \in S[r,\sigma] \cap S[r,\sigma']$. Let us recall that the local state of a process q at the end of a round r of a run σ of a deterministic algorithm such as P (namely, the local state $ls(q,r,\sigma)$) is a function of its local state in round $r-1$ ($ls(q,r-1,\sigma)$) and the messages that it receives in round r. We have to show that the local states $ls(q,r,\sigma)$ and $ls(q,r,\sigma')$ are the same.

Since $\rho \overset{r}{\sim}_q \rho'$, it follows that the arrays Inp_q are the same in both the runs ρ and ρ' at the end of round r. So, $\mathsf{Inp}_q[q]$ in both runs have the same value at the end of r; let X be that value.

The fact that q survives round r in both ρ and ρ'means, in particular, that it receives its own round r message sent at line 04 of FIP in both ρ and ρ'. The value of this message in ρ is $ls(q,r-1,\rho)$ which is the value of $\mathsf{Inp}_q[q]$ at the end of r, i.e., $ls(q,r-1,\rho) = X$. A similar reasoning shows that $ls(q,r-1,\rho') = X$. It follows from $ls(q,r-1,\rho) = ls(q,r-1,\rho') = X$ that $\rho \overset{r-1}{\sim}_q \rho'$ and by the inductive hypothesis we obtain $\sigma \overset{r-1}{\sim}_q \sigma'$. Consequently, q has the same local state at the end of round $r-1$ in both σ and σ', i.e., $ls(q,r-1,\sigma) = ls(q,r-1,\sigma')$.

It remains to show that q receives exactly the same messages during round r in both σ and σ'. Suppose that q receives message μ from process $\widehat{q}$ in round r of σ. It follows from the failure pattern F that the message sent by $\widehat{q}$ to q during round r is received by q. Since ρ corresponds to σ and in FIP process $\widehat{q}$ sends messages to all processes in every round, we have that q receives a message from $\widehat{q}$ in round r of ρ as well. As above (case of the message that, at each round, a process sends to itself), this message in ρ contains $ls(\widehat{q},r-1,\rho)$ (the local state of $\widehat{q}$ at round $r-1$ of the run ρ). From $\rho \overset{r}{\sim}_q \rho'$ we have that q receives the same message from $\widehat{q}$ in ρ'. As a result, we have that $\rho \overset{r-1}{\sim}_{\widehat{q}} \rho'$, and by the inductive assumption for $r-1$ and $\widehat{q}$ we obtain that $\sigma \overset{r-1}{\sim}_{\widehat{q}} \sigma'$. Since the message μ is determined by P as a function of the local state of $\widehat{q}$ at $r-1$ in σ, we have that the same message μ is also sent by $\widehat{q}$ to q in round r of σ'. As the round r message sent by $\widehat{q}$ to q in ρ' is received by q, and σ' corresponds to ρ', we obtain that q receives μ from $\widehat{q}$ in σ' as well. It follows that every message received by q in round r of σ is received by it in the same round of σ'. By symmetry, the messages received in σ' are also received in σ. Finally, since q has the same local state in round $r-1$ of σ and σ' (namely, $ls(q,r-1,\sigma) = ls(q,r-1,\sigma')$), and receives the same messages in round r of both σ and σ', we obtain that $\sigma \overset{r}{\sim}_q \sigma'$, which concludes the proof of the lemma. $\square_{Lemma\ 7}$