

The Heard-Of model: computing in distributed systems with benign faults

Bernadette Charron-Bost · André Schiper

Received: 21 July 2006 / Accepted: 22 April 2009 / Published online: 8 July 2009
© Springer-Verlag 2009

Abstract Problems in fault-tolerant distributed computing have been studied in a variety of models. These models are structured around two central ideas: (1) degree of synchrony and failure model are two *independent* parameters that determine a particular type of system, (2) the notion of *faulty component* is helpful and even necessary for the analysis of distributed computations when faults occur. In this work, we question these two basic principles of fault-tolerant distributed computing, and show that it is both possible and worthy to renounce them in the context of benign faults: we present a computational model based only on the notion of *transmission faults*. In this model, computations evolve in rounds, and messages missed in a round are lost. Only information transmission is represented: for each round r and each process p , our model provides the set of processes that p “hears of” at round r (*heard-of set*), namely the processes from which p receives some message at round r . The features of a specific system are thus captured as a whole, just by a predicate over the collection of heard-of sets. We show that our model handles benign failures, be they static or dynamic, permanent or transient, in a unified framework. We demonstrate how this approach leads to shorter and simpler proofs of important results (non-solvability, lower bounds). In particular, we prove that the Consensus problem cannot be generally solved without an implicit and permanent consensus on heard-of

sets. We also examine Consensus algorithms in our model. In light of this specific agreement problem, we show how our approach allows us to devise new interesting solutions.

Keywords Consensus · Benign fault · Computational model · HO (Heard-Of) model · Transmission fault

1 Introduction

Problems in fault-tolerant distributed computing have been studied in a variety of models. Such models are structured around two central ideas:

1. Degree of synchrony and failure model are two *independent* parameters that determine a particular type of system.
2. The notion of *faulty component* is helpful and even necessary for the analysis of distributed computations when faults occur.

In this paper we question these two basic principles of fault-tolerant distributed computing, and show that it is both possible and worthy to renounce them in the context of benign faults: we present a computational model, suitable for systems with benign failures, which is based only on the notion of *transmission fault*.

Computations in our model are composed of *rounds*. In each round, a process sends a message to each process, waits to receive messages from some processes, and then computes a new state. Every message received at some round has been sent at that round. Consequently, any message missed in a round is definitely discarded. Using the terminology of Elrad and Francez [18], a round is a *communication-closed-layer*.

A. Schiper’s research funded by the Swiss National Science Foundation under grant number 200021-111701 and Hasler Foundation under grant number 2070.

B. Charron-Bost (✉)
École polytechnique, CNRS, 91128 Palaiseau Cédex, France
e-mail: Charron@lix.polytechnique.fr

A. Schiper
EPFL, 1015 Lausanne, Switzerland
e-mail: Andre.Schiper@epfl.ch

Most of the solutions to agreement problems that have been designed for synchronous message-passing systems as well as for partially synchronous or asynchronous ones are structured in rounds (e.g., [1, 5, 14, 16, 17, 41]). However, concerning impossibility results (lower bounds, non-solvability, ...) in message-passing systems, round-based computational models have been considered almost always for synchronous systems. The reason for that lies in the fact that it is an open question whether round-based models are equivalent to the ones in which late messages are not discarded.

To the best of our knowledge, Dwork et al. [17] were the first to define a round-based model for non synchronous computing. More precisely, they generalized the classical round-based computational model for synchronous systems to a large class of partially synchronous systems. Then Gafni [22] extended the round-based model to any type of systems. The basic idea in his model is to study how a system evolves round-by-round and to abstract away the implementation of the communication between processes, be it shared-memory or message-passing. The properties of the communication mechanisms and system guarantees are captured as a whole by a single module that is called *Round-by-Round Fault Detector* (for short *RRFD*) module. More precisely, at each round r and for each process p , the module provides a set of suspected processes from which p will not wait for a message (here, we call *messages* the pieces of information that are exchanged, whatever the medium of communication is). At this point, only non-transmission of information appears in the model: the reason why a process is suspected is not specified, whether it is due to the fact that the process is late or has crashed. In this way, *synchrony degree and failure model are encapsulated in the same abstract entity*.

The latter idea seems quite sound since separating synchrony degree and failure model breaks the continuum that naturally exists between them: for example, message asynchrony means that there is no bound on message delays, and message loss corresponds to infinite delays. Moreover, capturing synchrony degree and failures with the same abstraction gives hope for relating different types of systems, in particular synchronous and asynchronous systems.

Unfortunately, this idea is not followed through to the end in [22] since the notion of failure model is underhandedly reintroduced via the one of *faulty component*. Indeed, the communication medium is implicitly assumed to be reliable (no anomalous delay, no loss) and when process p receives no message from q , the latter process is considered to be responsible for the transmission failure (q is late or has crashed). The so-called *RRFD* modules only suspect processes, never links. Obviously, this impacts the design and correctness proofs of algorithms: for example, agreement problems are specified in [22] as usual, exempting faulty processes from making a decision.

The *RRFD* model and its general extension *GIRAF*¹ [25] are here influenced by the *component failure* models proposed in the literature, e.g., [5, 12, 19, 20, 30, 36]. As pointed out in [37], the component failure models “do not fully capture the reality of systems subject to (possibly transient) ubiquitous failures”, which are systems in which “faults occur anywhere in the system and, following a failure, normal functioning case resume after finite (although unpredictable) time”. Moreover, the principle of a priori blaming some components for transmission failures yields several major problems [37, 38]. First, it may lead to undesirable conclusions: for example, in the send-omission failure model, the entire system will be considered faulty even if only one message from each process is lost. Hence there is no algorithm in this traditional component failure model that can tolerate such transient failures, how few they may be. Second, it allows faulty processes to have deviant behaviors: in decision problems, it is very common that a faulty process is not obliged to make a decision. Indeed, even in their uniform versions [23] that require coordination among *all* the processes that decide, including those to which faults are ascribed, decision problems share the same restricted termination clause that exempts faulty processes from making a decision. For example, a process p that is blamed for the non-delivery of a single message— as this transmission fault is rightly or wrongly accounted for an omission from p —is allowed to make no decision even if p is blamed for nothing else. Finally, as already observed by Dolev [12], it appears that the real causes of transmission faults, namely sender failure, receiver failure, or link failure, may be actually unknown. Transmission faults are often ascribed to some components in an arbitrary manner that may not correspond to reality.

Moreover, there is no *prima facie* evidence that the notion of faulty component is really helpful in the analysis of fault-tolerant distributed algorithms. We show that our model leads to the development of new conditions guaranteeing the correctness of fault-tolerant algorithms, and to shorter and simpler proofs. This is due to the fact that the notion of faulty component unnecessarily overloads system analysis with non-operational details. In other words, it is sufficient that the model just specifies transmission faults (effects) without accounting for the faulty components (causes).

Santoro and Widmayer [37, 38, 40] clearly pointed out this issue. They introduced the *Transmission Faults model* that locates faults without specifying their cause. A transmission fault can be due to a link failure as well as a process failure. Contrary to classical models in which transmission faults involve only messages sent or received by an unknown but static set of processes (the so-called *faulty processes*), the *Transmission Faults model* is well-adapted to *dynamic*

¹ We would like to thank one of the reviewers for having pointed out the *GIRAF* model [25].

failures. However, this model is designed only for synchronous systems. Indeed, Santoro and Widmayer showed that dynamic failures in synchronous systems have the same negative effect as asynchronicity. This *de facto* reintroduces synchrony degree and failure model as two separate parameters of systems. The Transmission Faults model, also known as *Dynamic/Mobile Faults* model, has also been investigated in [9, 11, 21, 26, 39].

Contribution

Our aim is to develop a computational model for distributed systems that combines the advantages of the RRF model [22] and the Transmission Faults model [37, 38], but avoids their drawbacks. We propose a round-based model, called *Heard-Of* (HO for short) in which (1) synchrony degree and failure model are encapsulated in the same high-level abstraction, and (2) the notion of faulty component (process or link) has totally disappeared. As a result, the HO model accounts for transmission faults without specifying by whom nor why such faults occur. There is no evidence that this leads to a more powerful model than existing models from a computational viewpoint. However, we prove the semantic effectiveness of the HO model, and show that it is more elegant and simpler, specially for the design and correctness proofs of fault-tolerant distributed algorithms (see Sect. 5). Indeed, simplicity and conciseness is crucial for model checking [42] and formal verification of algorithms [8].

A computation in the HO model evolves in rounds. In each round, a process sends a message to all the others, and then waits to receive messages from the other processes. Communication missed in a round is lost. For each round r and each process p , $HO(p, r)$ denotes the set of processes that p has “heard of” at round r , namely the processes from which p receives some message at round r . A transmission fault from q to p at round r is characterized by the fact that q does not belong to $HO(p, r)$.

Exactly as in the RRF model [22], the features of a specific system are captured in the HO model as a whole, just by a predicate over the collection of the $HO(p, r)$'s, called a *communication predicate*.

The HO model handles benign failures, be they static or dynamic, permanent or transient, in a unified framework. In particular, the model can naturally represent link failures, contrary to models with failure detectors [5, 22]. Indeed, in such models, when the failure detector module indicates to some process p to stop waiting for a message from q , this is interpreted as “ q is (suspected to be) faulty”. Obviously, such an interpretation makes no sense if links may lose messages.

Another feature of the HO model is that contrary to the random model [1, 10] or the Failure Detector model, there is no notion of “augmenting” asynchronous systems with external devices (oracles) that processes may query: the communi-

cation predicate corresponding to an HO system is an integral part of the model and should be rather seen as defining the *environment* [7]. The weaker the predicate of an HO system is, the more freedom the environment allows the system, and the harder it is to solve problems. The HO abstraction (communication predicates) is supported only by the messages sent in the HO algorithm. In other words, we do not decouple predicates from the underlying algorithms in our approach. This is the reason why we encapsulate algorithm and communication predicate in the same structure that we shall call an *HO machine*.

Besides the construction of the HO model, we present various results about the Consensus problem that illustrate its semantic effectiveness. Our first result concerns systems that never partition; it characterizes the minimal communication predicate needed to solve Consensus in such systems. To do so, we first introduce the concept of *translation* of communication predicates. Informally, a communication predicate \mathcal{P} can be translated into another one \mathcal{P}' if there is a distributed algorithm that transforms heard-of sets satisfying \mathcal{P} into new ones satisfying \mathcal{P}' . Any problem that is solvable under \mathcal{P}' is then solvable under \mathcal{P} instead. The so-defined relation is reflexive and transitive, and thus orders communication predicates with respect to their abilities to solve problems. If \mathcal{P} can be translated into \mathcal{P}' , then we say that \mathcal{P} is *at least as strong as* \mathcal{P}' .

Of special interest is the communication predicate $\mathcal{P}_{sp_unif}^*$ which guarantees that at each round, all processes hear of the same non-empty subset of processes. Such a permanent operational agreement on heard-of sets clearly suffices to solve Consensus. Conversely and more surprisingly, we show that under the condition that there is no heard-of set partitioning—i.e., at each round, any two processes hear of at least one common process—if Consensus is solvable with the communication predicate \mathcal{P} , then \mathcal{P} is at least as strong as $\mathcal{P}_{sp_unif}^*$. In other words, Consensus cannot be solved without an implicit permanent agreement on the heard-of sets.

Then we describe four basic translations. Using these translations, we prove several results related to the communication predicate guaranteeing that every round has a non-empty *kernel*, i.e., at each round there is some process that is heard by all. We show that non-empty kernel rounds can be emulated by majority heard-of sets, and more generally, can be emulated in any system that never partitions. By means of these basic translations, we also give a simple direct proof of the reduction of the worst-case synchronous lower bounds [15, 31] to the general FLP asynchronous impossibility result [20] (this reduction has been previously established by Gafni for Atomic-Snapshot asynchronous systems [22]). This exemplifies how, by getting rid of the first principle which artificially separates synchrony degree and failure model, we can describe synchronous and asynchronous systems in a unified framework, and take advantage of this to

relate impossibility results that are traditionally considered as quite different in essence.²

Finally we study how to solve Consensus in systems prone to partitioning. The HO formalism enables us to describe well-known Consensus algorithms, and also to design new solutions. For each Consensus algorithm, we determine a simple communication predicate which guarantees correctness. Interestingly, all the communication predicates that we display express conditions that have to hold just sporadically, contrary to the perpetual correctness condition “ Ω ” in the Failure Detector model [4, 7]. Hence, the HO model seems to be a natural formalism for expressing fine-grained conditions with respect to time. Moreover, in many real systems, we observe series of “bad” and “good” periods with regard to both synchrony and faults. Since they just require sporadic conditions on heard-of sets, the algorithms that we examine are well-adapted to such systems, and so are quite realistic solutions to the Consensus problem.

This paper is structured as follows. In Sect. 2, we describe our model, and present many traditional systems in the HO framework. In Sect. 3, we define the notion of translation and propose a characterization of the communication predicates that make Consensus solvable (under certain transmission fault bounds). In Sect. 4, we give four basic translations, and highlight the key role played by the “no partitioning” assumption. In Sect. 5, we describe several Consensus algorithms, and determine HO conditions for their correctness. Section 6 concludes the paper.

2 HO model

As explained in the Introduction, computations in our model are composed of rounds, which are communication-closed layers in the sense that any message sent in a round can be received only at that round. The technical description of computations is similar to the ones in [17, 22], and so the model generalizes the classical notion of synchronized rounds developed for synchronous systems [29]. We introduce the notion of *kernel* of round r that represents what processes share during round r from the operational viewpoint. As we shall show, this notion plays a key role in solving Consensus.

² There is another approach to unify synchronous and asynchronous models, which consists in developing tools for model-independent analysis of decision problems, instead of using translations between system models. More specifically, [24] develops arguments from algebraic topology for synchronous, partially synchronous, and asynchronous systems as well, while [32] introduces the notion of *layering* as a tool for model-independent analysis of the Consensus problem. Note that the approaches in [24, 32] both use the notion of faulty component.

2.1 Heard-of sets and communication predicates

We suppose that we have a non-empty set Π of cardinality n , a set of messages M , including a *null* placeholder indicating the empty message. To each $p \in \Pi$, we associate a *process*, which consists of the following components: a set of states denoted by $states_p$, a subset $init_p$ of initial states, for each positive integer r called *round number*, a message-sending function S_p^r mapping $states_p \times \Pi$ to a unique (possibly *null*) message, and a state-transition function T_p^r mapping $states_p$ and partial vectors (indexed by Π) of elements of M to $states_p$. In each round r , process p first applies S_p^r to the current state, emits the “messages” to be sent to each process, and then, for a subset $HO(p, r)$ of Π (indicating the processes which p hears of), applies T_p^r to its current state and the partial vector of incoming messages whose support is $HO(p, r)$. The collection of processes is called an *algorithm on Π* .

Computation evolves in an infinite sequence of rounds. For each computation, we determine its *heard-of collection* which is the collection of subsets of Π indexed by $\Pi \times \mathbb{N}^*$:

$$(HO(p, r))_{p \in \Pi, r > 0}.$$

A *communication predicate* \mathcal{P} is defined to be a predicate over heard-of collections, that is a boolean function over the collections of subsets of Π indexed by $\Pi \times \mathbb{N}^*$:

$$\mathcal{P} : (2^\Pi)^{\Pi \times \mathbb{N}^*} \longrightarrow \{\text{false}, \text{true}\}.$$

Since we want to define a non-trivial notion of solvability (see Sect. 2.2), we only consider communication predicates different from the constant predicate “false”. Moreover, as we do not want the correctness of algorithms to depend on the time (or round) at which algorithms start to run (see Proposition 1), we restrict ourselves to the class of communication predicates \mathcal{P} whose truth-value is invariant under time translation, i.e., \mathcal{P} has the same truth-value for all the heard-of collections $(HO(p, r + i))_{p \in \Pi, r > 0}$, where i is any given integer. Such a communication predicate will be said to be *time invariant*.

For example, consider some condition \mathcal{C} over the heard-of sets in a given round, that is a predicate over collections of subset of Π indexed (only) by Π :

$$\mathcal{C} : (2^\Pi)^\Pi \longrightarrow \{\text{false}, \text{true}\}.$$

Then, the communication predicate

$$\mathcal{P}_{(\mathcal{C})} :: \exists r_0 \geq 0 : \mathcal{C} \text{ holds at } r_0$$

is not time invariant if \mathcal{C} is not constant, while

$$\mathcal{P}_{(\mathcal{C})}^\infty :: \forall r > 0, \exists r_0 \geq r : \mathcal{C} \text{ holds at } r_0,$$

which expresses that \mathcal{C} holds infinitely often, is a time invariant communication predicate. From now on, the term

communication predicate will only refer to a non-trivially false and time invariant predicate.

For any round r , its *kernel* is defined as the set of processes

$$K(r) = \bigcap_{p \in \Pi} HO(p, r).$$

Intuitively, it consists of the processes which are heard by all in round r . More generally, we introduce the kernel $K(\phi)$ of any set ϕ of rounds as:

$$K(\phi) = \bigcap_{r \in \phi} K(r).$$

When ϕ is the set of all the rounds in the computation, this defines the (*global*) *kernel* of the computation:

$$K = \bigcap_{r > 0} K(r).$$

It will be convenient to introduce the *cokernel* of some round, or more generally of some collection of rounds, as the complement in Π of the above defined kernels. Thus, with the same notation as above, we let

$$coK(r) = \Pi \setminus K(r), \quad coK(\phi) = \Pi \setminus K(\phi), \quad coK = \Pi \setminus K.$$

Round r is said to be *uniform* when, for any two processes p, q in Π ,

$$HO(p, r) = HO(q, r).$$

Round r is said to be a *nek* (for *non-empty kernel*) round if

$$K(r) \neq \emptyset,$$

and it is said to be *split* when there exist two processes p, q in Π such that

$$HO(p, r) \cap HO(q, r) = \emptyset.$$

Obviously, a nek round is not split, but the converse does not hold. Moreover, a non-trivial uniform round, that is a uniform round with a non-empty common heard-of set, is a nek round.

A *nek computation* is a computation whose global kernel is non-empty. In such a computation, there is at least one process from which every process hears during the whole computation. A computation is said to be *space uniform* when each of its rounds is uniform. It is said to be *time uniform* when the sets $HO(p, r)$ do not vary in time:

$$\forall r > 0, \forall p \in \Pi : HO(p, r) = HO(p, r + 1).$$

Finally, a computation is said to be *regular* when a process that is not heard by some process at some round is not heard by any process later:

$$\forall r > 0, \forall p \in \Pi : HO(p, r + 1) \subseteq K(r).$$

Note that regularity is a weak form of the combination of space and time uniformity.

Equivalently, we could rather consider *talked-to* sets, denoted $TT(p, r)$, and defined by

$$TT(p, r) = \{q \in \Pi : p \in HO(q, r)\},$$

which are the dual notion of heard-of sets. Contrary to $HO(p, r)$, process p cannot know $TT(p, r)$ at the end of round r , and this is the reason why we have preferred to express the communication properties of computations in terms of their heard-of collections instead of their talked-to collections.

2.2 HO machines

A *Heard-Of machine* (or *HO machine* for short) for Π is a pair $M = (A, \mathcal{P})$ where A is an algorithm on Π and \mathcal{P} is a communication predicate. For example, we shall consider the HO machines with the communication predicate:

$$\mathcal{P}_{sp_unif} :: \forall r > 0, \forall p, q \in \Pi^2 : HO(p, r) = HO(q, r),$$

that is HO machines with space uniform computations, and those with regular computations:

$$\mathcal{P}_{reg} :: \forall r > 0, \forall p \in \Pi : HO(p, r + 1) \subseteq K(r).$$

We shall also consider the class of HO machines that share the “no split” predicate:

$$\mathcal{P}_{nosplit} :: \forall r > 0, \forall p, q \in \Pi^2 :$$

$$HO(p, r) \cap HO(q, r) \neq \emptyset,$$

the subclass of HO machines with the communication predicate:

$$\mathcal{P}_{nekrounds} :: \forall r > 0 : K(r) \neq \emptyset,$$

and the one with the stronger communication predicate:

$$\mathcal{P}_{nek} :: K \neq \emptyset.$$

More generally, we introduce the communication predicate:

$$\mathcal{P}_K^f :: |coK| \leq f$$

which is equivalent to

$$|K| \geq n - f.$$

We shall also consider the weaker predicate:

$$\mathcal{P}_{HO}^f :: \forall r > 0, \forall p \in \Pi : |HO(p, r)| \geq n - f$$

and more specifically $\mathcal{P}_{HO}^{maj} = \mathcal{P}_{HO}^{\lfloor \frac{n-1}{2} \rfloor}$ that asserts every heard-of set is a majority set.

A *run* of (A, \mathcal{P}) is totally determined by a set of initial states (one per process) and a heard-of collection that satisfies \mathcal{P} . To each run corresponds the collection of the states (one per process and per round) reached by processes during the run; we denote p 's state at the end of round r by $\sigma_p^{(r)}$. By extension, the initial state of p is denoted by $\sigma_p^{(0)}$. In the

sequel, given a run of M , for any variable X_p of process p , $X_p^{(r)}$ will denote the value of X_p after r rounds of this run.

A problem Σ for Π is a predicate over state collections indexed by $\Pi \times \mathbb{N}^*$:

$$\Sigma : \left(s_p^r \right)_{p \in \Pi, r > 0} \longrightarrow \Sigma \left((s_p^r)_{p \in \Pi, r > 0} \right) \in \{\text{false}, \text{true}\},$$

where $s_p^r \in \text{states}_p$ for every $p \in \Pi$. An HO machine $M = (A, \mathcal{P})$ solves a problem Σ if the state collection in each of its runs satisfies Σ ; then we say that problem Σ is *solvable under \mathcal{P}* .

Since communication predicates are invariant under time translation, and round numbers are not part of process states, correctness of algorithms does not depend on the time at which algorithms start to run. Formally, for any integer i and any HO algorithm A with the message-sending and state-transition functions S_p^r and T_p^r respectively, let ${}^i A$ denote the algorithm defined by the message-sending functions ${}^i S_p^r$ and the state-transition functions ${}^i T_p^r$ such that ${}^i S_p^r$ and ${}^i T_p^r$ are trivial for the first i rounds, and for any round $r > i$,

$${}^i S_p^r = S_p^{r-i} \quad \text{and} \quad {}^i T_p^r = T_p^{r-i}.$$

Proposition 1 *If the HO machine $M = (A, \mathcal{P})$ solves Σ , then for any integer i , the HO machine ${}^i M = ({}^i A, \mathcal{P})$ also solves Σ .*

In this paper, we concentrate on the well-known agreement problem, called *Consensus*. In this problem, each process p has an initial value v_p from a fixed set V , and must reach an irrevocable decision on one of the initial values. Thus each value v in V corresponds to an initial state s_p^v of process p , signifying that p 's initial value is v : $\sigma_p^{(0)} = s_p^v$.

Process p has also disjoint sets of *decision states* Σ_p^v , one per value v in V , meaning that p has decided v . Then the *Consensus problem in V* is defined as the conjunction of the following requirements:

Irrevocability. Once a process decides a value, it remains decided on that value:

$$\forall p \in \Pi, \forall v \in V, \forall r > 0 : \\ \sigma_p^{(r)} \in \Sigma_p^v \Rightarrow \forall r' \geq r : \sigma_p^{(r')} \in \Sigma_p^v.$$

Agreement. No two processes decide differently:³

$$\forall p, q \in \Pi, \forall v, w \in V, \forall r, r' > 0 : \\ \sigma_p^{(r)} \in \Sigma_p^v \wedge \sigma_q^{(r')} \in \Sigma_q^w \Rightarrow v = w.$$

³ Note that under agreement, irrevocability is equivalent to the following weaker condition:

$$\forall p \in \Pi, \forall v \in V, \forall r > 0 : \sigma_p^{(r)} \in \Sigma_p^v \Rightarrow \forall r' \geq r, \exists v' \in V : \sigma_p^{(r')} \in \Sigma_p^{v'}.$$

This remark will be used in several places in the sequel.

Integrity. Any decision value is the initial value of some process:

$$\forall v \in V, \forall p \in \Pi, \forall r > 0 : \\ \sigma_p^{(r)} \in \Sigma_p^v \Rightarrow \exists q \in \Pi : \sigma_q^{(0)} = s_q^v.$$

Termination. All processes eventually decide:

$$\forall p \in \Pi, \exists r_p > 0, \exists v \in V : \sigma_p^{(r_p)} \in \Sigma_p^v.$$

Since there is no notion of faulty process in the HO model, a process is never exempted from making a decision. Such a strong liveness requirement may seem unreasonable in two respects. Firstly, it may make Consensus needlessly unsolvable in the sense that the resulting Consensus specification might be unsolvable under some communication predicate \mathcal{P} whereas the classical Consensus problem (with the non-uniform Termination condition) is solvable in the type of systems corresponding to \mathcal{P} (see Table 1). The paper shows that this objection does not hold.

Secondly, one may wonder whether an HO algorithm in which *all processes* decide can be implemented in systems with processes that are prone to crash failures. The answer is yes. Of course, a process that has crashed takes no steps, and so can make no decision. However, the corresponding HO process is not heard of any more, and so has no impact on the rest of the computation.

More precisely, let \mathcal{S} be a system with possible process crash failures, and let \mathcal{P} be a communication predicate corresponding to \mathcal{S} (cf. Table 1). Suppose that there is a round-based computational model for \mathcal{S} with similar notions of algorithm, run of an algorithm, problem, and solvability (e.g., the computational model described in [29] for synchronous systems with crash failures). Let $\Sigma^{\mathcal{S}}$ be a problem in this model, and Σ its “entirely uniform” version for the HO model. Assume that $M = (A, \mathcal{P})$ is an HO machine solving Σ . Like any HO algorithm, A is also an algorithm in the model for \mathcal{S} . Since a process in \mathcal{S} that has crashed can take no further step, Σ is trivially not solvable in \mathcal{S} , and in particular A does not solve Σ in \mathcal{S} . However, we claim that A solves $\Sigma^{\mathcal{S}}$:

Proposition 2 *If the HO machine (A, \mathcal{P}) solves Σ , then A solves $\Sigma^{\mathcal{S}}$ in \mathcal{S} .*

We give only the intuitive argument, for lacking of a formal correspondence between \mathcal{S} and \mathcal{P} , and between $\Sigma^{\mathcal{S}}$ and Σ .

Proof (sketched) Let $\rho^{\mathcal{S}}$ be any run of A in \mathcal{S} . By mapping any process in \mathcal{S} that crashes at some point t in round r onto an HO process that is no more heard of after t , we easily construct a run ρ of A that extends $\rho^{\mathcal{S}}$, i.e., $\rho^{\mathcal{S}}$ is a subsequence of ρ . Since (A, \mathcal{P}) solves Σ , ρ satisfies Σ , and so

the projection ρ^S satisfies Σ^S , which is some “projection” of Σ on the set of correct processes. \square

In the particular case of Consensus, the above argument shows that there is no problem of transposing an HO machine solving the entirely uniform Consensus specification in a system \mathcal{S} with possible crash failures: the capability of an HO process to make a decision (in ρ) is just not implemented (in ρ^S) if the corresponding process in \mathcal{S} has crashed.

2.3 How to guarantee communication predicates

Obviously, an HO machine is implementable in a system as soon as the corresponding communication predicate can be guaranteed by the system. In Table 1, we go over various classical types of message-passing systems of interest, and we examine the communication predicates that they can guarantee. For each type of system listed in Table 1 except asynchronous systems with initial crash failures, we use several results previously established in [13, 17, 22]. As for asynchronous systems with at most f initial crash failures, they clearly support the communication predicate $\mathcal{P}_{\diamond unif}^f$ defined by:

$$\mathcal{P}_{\diamond unif}^f :: \exists r_0 > 0, \exists \Pi_0 \in 2^\Pi \text{ s.t. } |\Pi_0| \geq n - f, \\ \forall p \in \Pi, \forall r \geq r_0 : HO(p, r) = \Pi_0.$$

Moreover, the positive result by Fischer et al. [20] for initial crash failures shows that in the case of a majority of correct processes ($2f < n$), space-time uniformity of the heard-of sets can be achieved from the beginning.⁴ That is, HO machines with the predicate

$$\mathcal{P}_{unif}^f :: \exists \Pi_0 \in 2^\Pi \text{ s.t. } |\Pi_0| \geq n - f, \\ \forall p \in \Pi, \forall r > 0 : HO(p, r) = \Pi_0$$

can be implemented in any asynchronous system provided a majority of processes is correct.

3 Communication predicates to solve Consensus

In this section, we address the fundamental question of determining the computational models in which Consensus is solvable. In terms of the HO model, it consists in identifying the communication predicates for which Consensus is solvable.

We partially answer the question by restricting attention to the class of communication predicates from which nek rounds can be emulated: in this class of HO models, we prove that permanent space uniformity is a necessary and sufficient condition for solving Consensus. In other words, *Consensus*

⁴ The algorithm in [20] ensures agreement on the membership of the initial clique.

cannot be solved without an implicit and permanent consensus on heard-of sets.

We start by formalizing what it means for an HO machine $M = (A, \mathcal{P})$ to emulate a communication predicate \mathcal{P}' . To do that, we first define the notion of a k -round translation from \mathcal{P} to \mathcal{P}' , and then its generalization to translations that take a non constant number of rounds. Translations of the first type are called *uniform translations*.

3.1 Uniform translations

Let k be any positive integer, and let A be an algorithm that maintains a variable $NewHO_p$ at every process p , which contains a subset of Π . We call *macro-round* ρ the sequence of the k consecutive rounds $k(\rho - 1) + 1, \dots, k\rho$. The value of $NewHO_p$ at the end of macro-round ρ is denoted $NewHO_p^{(\rho)}$. We say that the HO machine $M = (A, \mathcal{P})$ *emulates* the communication predicate \mathcal{P}' in k rounds if for any run of M , the following holds:

- E1: If process q belongs to $NewHO_p^{(\rho)}$, then there exist an integer l in $\{1, \dots, k\}$, a chain of $l + 1$ processes p_0, p_1, \dots, p_l from $p_0 = q$ to $p_l = p$, and a subsequence of l increasing round numbers r_1, \dots, r_l in macro-round ρ such that for any index $i, 1 \leq i \leq l$, we have $p_{i-1} \in HO(p_i, r_i)$.
- E2: The collection $(NewHO_p^{(\rho)})_{p \in \Pi, \rho > 0}$ satisfies predicate \mathcal{P}' .

Condition E1 states that if q is in $NewHO_p$ at macro-round ρ , then p has actually heard of q during this macro-round through some intermediate processes p_1, \dots, p_{l-1} . Hence this condition excludes trivial emulations of \mathcal{P}' . Condition E2 states that the variables $NewHO_p$ simulate heard-of sets satisfying \mathcal{P}' . If there exists an algorithm A such that the HO machine (A, \mathcal{P}) emulates \mathcal{P}' in k rounds, then we write $\mathcal{P} \succeq_k \mathcal{P}'$, and we say that A is a *k round translation* of \mathcal{P} into \mathcal{P}' .

Note that if $\mathcal{P} \Rightarrow \mathcal{P}'$, the trivial algorithm in which each process p writes the value of $HO(p, r)$ into $NewHO_p$ at the end of each round r is a one round translation of \mathcal{P} into \mathcal{P}' , and so $\mathcal{P} \succeq_1 \mathcal{P}'$.

3.2 General translations

Now we generalize the previous definition to translations that take a non-constant number of rounds in time and space. For that, each process p maintains an additional variable $MacroRound_p$ initialized to 0. Upon updating $NewHO_p$, process p increments $MacroRound_p$ by 1. When p sends a basic message m , it tags m with the current value of $MacroRound_p$. Moreover, p ignores any message tagged by

Table 1 Classical system types and communication predicates

System type	Communication predicate
Synchronous system, reliable links, at most f faulty senders	$ K \geq n - f$
Synchronous system, at most f omission transmission faults per round [38]	$\forall r > 0 : \sum_{p \in \Pi} HO(p, r) \geq n^2 - f$
Synchronous system, a block of at most f omission transmission faults [38]	$\forall r > 0 : K(r) \geq n - f$
Synchronous system, reliable links, at most f crash failures	$ K \geq n - f$ \wedge $\forall p \in \Pi, \forall r > 0 : HO(p, r + 1) \subseteq K(r)$
Synchronous communication, asynchronous processes, reliable links, atomic send to all, at most f crash failures [13]	$\forall p \in \Pi, \forall r > 0 : HO(p, r) \geq n - f$ \wedge $\forall p, q \in \Pi^2, \forall r > 0 : HO(p, r) = HO(q, r)$
Synchronous communication, asynchronous processes, reliable links, at most 1 crash failure [13]	$\forall p \in \Pi, \forall r > 0 : 1 \leq HO(p, r) \leq 2$ \wedge $\forall p, q \in \Pi^2, \forall r > 0 : HO(p, r) = HO(q, r)$
Asynchronous system, reliable links, at most f crash failures	$\forall p \in \Pi, \forall r > 0 : HO(p, r) \geq n - f$
Asynchronous system, reliable links, at most f initial crash failures	$\forall p \in \Pi : HO(p, 1) \geq n - f$ $\wedge (\forall p \in \Pi, \forall r > 0 : HO(p, r) \subseteq HO(p, r + 1)) \wedge$ $\exists \Pi_0 \subseteq \Pi, \exists r_0 > 0, \forall p \in \Pi, \forall r > r_0 : HO(p, r) = \Pi_0$
Same with $f < n/2$ [20]	$\exists \Pi_0 \subseteq \Pi$ s.t. $ \Pi_0 \geq n - f, \forall p \in \Pi, \forall r > 0 :$ $HO(p, r) = \Pi_0$
Partially synchronous system, eventual reliable links, at most f crash failures [17]	$\exists \Pi_0 \subseteq \Pi$ s.t. $ \Pi_0 \geq n - f, \exists r_0 > 0, \forall p \in \Pi, \forall r > r_0 :$ $HO(p, r) = \Pi_0$

an integer different to the current value of $MacroRound_p$. Then rephrasing the condition E1 as follows

E1: If process q belongs to $NewHO_p^{(\rho)}$, then there exist a chain of processes p_0, p_1, \dots, p_l from $p_0 = q$ to $p_l = p$, and a subsequence of l increasing round numbers r_1, \dots, r_l such that for any index $i, 1 \leq i \leq l$, we have

$$r_i < r_{i+1}, MacroRound_{p_i}^{(r_i)} = \rho - 1, \text{ and} \\ p_{i-1} \in HO(p_i, r_i).$$

yields a general definition of translation.

If there exists an algorithm A such that the HO machine (A, \mathcal{P}) emulates \mathcal{P}' , we write $\mathcal{P} \succeq \mathcal{P}'$, and we say that A translates \mathcal{P} into \mathcal{P}' . Obviously, the relation \succeq contains all the relations \succeq_k .

Given an emulation of \mathcal{P}' by an HO machine (A, \mathcal{P}) , any problem that can be solved with \mathcal{P}' , can be solved with \mathcal{P} instead. To see this, suppose that the HO machine (B, \mathcal{P}') solves a problem Σ . We compose A and B in the following way: each process p executes B with rounds that are “split” by A . More precisely, concurrently with B , every process p runs A , and so (locally) determines what we call “A macro-

rounds” and maintains the variable $A.NewHO_p$. The algorithm B at process p is then modified as follows: messages of A during an A macro-round ρ piggyback messages sent by B at round ρ , and p computes its new state at the end of macro-round ρ by applying B 's state-transition function at round ρ to (1) its state (with respect to B) at the beginning of ρ and (2) the partial vector of B 's messages indexed by $A.NewHO_p^{(\rho)}$.

Proposition 3 *If Σ is solvable under \mathcal{P}' and $\mathcal{P} \succeq \mathcal{P}'$, then Σ is solvable under \mathcal{P} .*

The relation \succeq is clearly transitive; thus it orders communication predicates with respect to their ability to solve problems. If both $\mathcal{P} \succeq \mathcal{P}'$ and $\mathcal{P}' \succeq \mathcal{P}$ hold, then we say that \mathcal{P} and \mathcal{P}' are *equivalent*, and we denote $\mathcal{P} \simeq \mathcal{P}'$.

As we shall see below, an important class of translations are those that preserve kernels. More precisely, we introduce the notion of a *kernel preserving translation* from \mathcal{P} to \mathcal{P}' which is defined as an emulation of \mathcal{P}' with an HO machine $M = (A, \mathcal{P})$ such that for any run of M , we have:

$$\bigcap_{p \in \Pi, r \in \rho_p} HO(p, r) \subseteq \bigcap_{p \in \Pi} NewHO_p^{(\rho)},$$

where ρ_p denotes the set of rounds that together form the macro-round ρ on p .

3.3 Consensus and nek rounds

Let $\mathcal{P}_{sp_unif}^*$ denote the communication predicate that guarantees any round to be uniform and non-trivial, that is $\mathcal{P}_{sp_unif}^* = \mathcal{P}_{sp_unif} \wedge \mathcal{P}^*$ with

$$\mathcal{P}^* :: \forall r > 0, \forall p \in \Pi : HO(p, r) \neq \emptyset.$$

Proposition 4 *Let \mathcal{P} be a communication predicate such that $\mathcal{P} \succeq \mathcal{P}_{sp_unif}^*$. Then for any set of values V , there exists an algorithm A such that the HO machine (A, \mathcal{P}) solves Consensus in V .*

Proof Let A_0 be an algorithm on Π such that (A_0, \mathcal{P}) emulates $\mathcal{P}_{sp_unif}^*$, and let us fix an arbitrary order p_1, \dots, p_n on Π . Let A be identical to A_0 , except that

1. at each round, each process sends its knowledge about initial values to all;
2. at the end of the first macro-round, each process decides the initial value of the first process in $NewHO_p$, according to the order p_1, \dots, p_n .

Note that thanks to E2, every $NewHO_p$ is non-empty at the end of macro-round 1. Moreover, from E1 it follows that the decision rule is well-defined since each process knows the initial values of all the processes in the set $NewHO_p$. Hence the decision rule is well-defined, and termination is satisfied. Integrity is a straightforward consequence of item 2. Agreement follows from E2. □

This result can be interestingly compared with the impossibility of Consensus with the communication predicate $\mathcal{P}_{\diamond_unif}^*$ (cf. Sect. 2.3). It turns out that eventual space-time uniformity is not sufficient for Consensus, whereas space uniformity alone makes Consensus solvable provided it holds from the beginning.

Conversely, the following proposition shows that in the class of HO machines with nek rounds, space uniformity can be achieved permanently if binary Consensus is solvable. In other words, *Consensus is solvable only if there is an implicit permanent agreement on the heard-of sets.*

Proposition 5 *Let \mathcal{P} be a communication predicate such that $\mathcal{P} \succeq \mathcal{P}_{nekrounds}$. If there is an HO machine (A, \mathcal{P}) that solves binary Consensus, then $\mathcal{P} \succeq \mathcal{P}_{sp_unif}^*$.*

Proof Let B be an algorithm that emulates $\mathcal{P}_{nekrounds}$ from \mathcal{P} . From A and B , we design an algorithm C and prove that (C, \mathcal{P}) emulates $\mathcal{P}_{sp_unif}^*$.

To simulate a macro-round with C , every process p first executes one macro-round of B and records the value of

$B.NewHO_p$ at the end of the macro-round in some variable $Propose_p$. Then it executes n instances of A in parallel, where each solves Consensus (cf. Proposition 1). The initial value of p for the i th instance of A is the truth-value of “ $p_i \in Propose_p$ ”. From the decision values, p sets

$$C.NewHO_p := \{p_i \in \Pi : p \text{ decides “true” for the } i\text{-th Consensus}\}.$$

By the agreement condition of Consensus, the emulated macro-round is uniform. Moreover, since B emulates $\mathcal{P}_{nekrounds}$, there is at least one process p_i that belongs to all the $B.NewHO_p$'s, and so all the initial values for the i th Consensus are equal to “true”. By the integrity condition of Consensus, the only possible decision value is “true”. In other words, we have $p_i \in \bigcap_{p \in \Pi} C.NewHO_p$. This shows that the emulated uniform macro-round is non-trivial, and so E2 is satisfied.

We now argue E1. Consider a C macro-round (made up of a B macro-round and n executions of A in parallel), and let p_i in $C.NewHO_p$ at the end of the C macro-round. By the integrity condition of Consensus, there is some processes x such that $p_i \in Propose_x$. By the Knowledge Transfer theorem [6], for one of them, say x_1 , there is a finite sequence of processes $x_2, \dots, x_k = p$ such that during the execution of the i th instance of A , x_1 sends a message m_1 to x_2 , x_2 sends a message m_2 to x_3 after receiving m_1 , ..., x_{k-1} sends a message m_{k-1} to $x_k = p$ after receiving m_{k-2} . Moreover, since E1 holds for B macro-rounds, there is a communication path from p_i to x_1 during the first part of the C macro-round. Hence there is a connection from p_i to p during the whole C macro-round. □

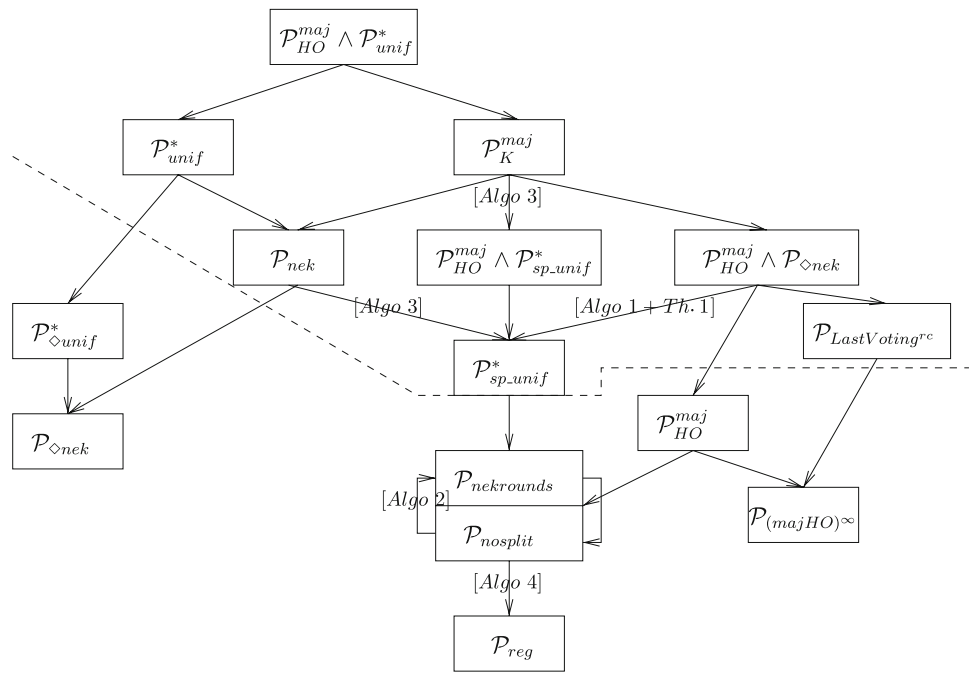
Since binary consensus is trivially reducible to multi-valued Consensus, Propositions 4 and 5 provide a characterization of the HO machines with nek rounds that solve Consensus:

Theorem 1 *In the class of communication predicates which are at least as strong as $\mathcal{P}_{nekrounds}$, the following assertions are equivalent:*

1. For any set V , there is an algorithm A such that (A, \mathcal{P}) solves Consensus in V ;
2. $\mathcal{P} \succeq \mathcal{P}_{sp_unif}^*$.

Note that $\mathcal{P}_{sp_unif}^*$ is the HO counterpart of one of the system types described in [13], namely the one with asynchronous processes, synchronous communication, reliable links, an atomic send-to-all primitive, and at most $n - 1$ crashes. Consequently Theorem 1 shows that, among the system types in [13] for which Consensus is solvable, this special one is the weakest; indeed all of them can emulate nek rounds.

Fig. 1 Relationships among some basic communication predicates and solvability of Consensus



4 Basic communication predicate translations

In this section, our aim is to establish some relationships among communication predicates, and to outline a first (partial) map of various classes of these predicates that play a key role for solving Consensus. To do so, we describe several fundamental translations that are all uniform. Such translations simply handle union and intersection of heard-of sets. Interestingly, some of them allow us to amplify our characterization of nek round communication predicates that make Consensus solvable. We compare these translations with other ones that have been given in the literature in the context of the classical taxonomy of system types. Our main results are summarized in Fig. 1.

4.1 A two round translation for increasing kernels

First we present a two round translation and prove a lower bound on the membership of the (new) kernels of macro-rounds. As a result, the translation increases kernels in some significant cases. In particular, it transforms \mathcal{P}_{HO}^{maj} into $\mathcal{P}_{nekrounds}$, that is $\mathcal{P}_{HO}^{maj} \succeq \mathcal{P}_{nekrounds}$. This translation also provides a direct proof of a very interesting result established by Gafni [22] relating synchronous and asynchronous systems.

The translation computes $NewHO_p^{(\rho)}$ from the collection of heard-of sets in rounds $2\rho - 1$ and 2ρ as follows (see Algorithm 1):

$$NewHO_p^{(\rho)} := \bigcup_{q \in HO(p, 2\rho)} HO(q, 2\rho - 1).$$

Algorithm 1 Translation for increasing kernels

- 1: **Initialization:**
- 2: $NewHO_p \in 2^V$, initially empty
- 3: **Round r :**
- 4: S_p^r :
- 5: **if** $r = 2\rho$ **then**
- 6: send $\langle HO(p, r - 1) \rangle$ to all processes
- 7: T_p^r :
- 8: **if** $r = 2\rho$ **then**
- 9: $NewHO_p := \bigcup_{q \in HO(p, r)} HO(q, r - 1)$

In this way, we emulate a macro-round ρ whose kernel satisfies the following key property:

Proposition 6 *If all heard-of sets at rounds $2\rho - 1$ and 2ρ contain at least $n - f_1$ and $n - f_2$ processes, respectively, then*

$$|\tilde{K}^{(\rho)}| \geq n - f_1 \left(1 + \frac{f_2}{n - f_2} \right),$$

where $\tilde{K}^{(\rho)} = \bigcap_{p \in \Pi} NewHO_p^{(\rho)}$.

Proof Consider the directed graph G_ρ whose vertices are the processes in Π , and there is an edge from p to q if and only if p belongs to $HO(q, 2\rho - 1)$. For any vertex x in G_ρ , let $nbIn(x)$ and $nbOut(x)$ be the numbers of in-neighbors and out-neighbors of x , respectively. The number of edges in G_ρ is equal to

$$E(G_\rho) = \sum_{x \in \Pi} nbIn(x) = \sum_{y \in \Pi} nbOut(y),$$

and since $nbIn(x) = |HO(x, 2\rho - 1)|$, we have

$$E(G_\rho) \geq n(n - f_1). \tag{1}$$

Let us separate the summation $\sum_{y \in \Pi} nbOut(y)$ into those y 's in $\tilde{K}^{(\rho)}$ and those not in $\tilde{K}^{(\rho)}$, and let \tilde{k}_ρ denote the cardinality of $\tilde{K}^{(\rho)}$. Clearly, we have

$$\sum_{y \in \tilde{K}^{(\rho)}} nbOut(y) \leq n\tilde{k}_\rho. \tag{2}$$

For the other term in the sum, we show that for any y that is not in $\tilde{K}^{(\rho)}$, we have

$$nbOut(y) \leq f_2. \tag{3}$$

This is true because if y is not in $\tilde{K}^{(\rho)}$, then there exists some process p such that

$$y \notin \bigcup_{q \in HO(p, 2\rho)} HO(q, 2\rho - 1),$$

that is for any q in $HO(p, 2\rho)$, y is not in $HO(q, 2\rho - 1)$. In other words, none of the out-neighbors of y in G_ρ belongs to $HO(p, 2\rho)$. Since any heard-of set $HO(p, 2\rho)$ has at least $n - f_2$ elements, $nbOut(y)$ is at most f_2 . From (1), (2) and (3) it follows that

$$n(n - f_1) \leq n\tilde{k}_\rho + (n - \tilde{k}_\rho)f_2,$$

and so

$$\tilde{k}_\rho \geq n - f_1 \left(1 + \frac{f_2}{n - f_2}\right).$$

□

With the communication predicate $\mathcal{P}_{HO}^{maj} = \mathcal{P}_{HO}^{\lceil \frac{n-1}{2} \rceil}$, Proposition 6 can be specialized as follows:

Corollary 1 *There is a two round translation of \mathcal{P}_{HO}^{maj} into $\mathcal{P}_{nekrounds}$, and so $\mathcal{P}_{HO}^{maj} \geq \mathcal{P}_{nekrounds}$.*

Proof Take $f_1 = f_2 = \lceil \frac{n-1}{2} \rceil$, which leads to $\tilde{k}_\rho \geq 1$. □

Another interesting corollary of Proposition 6 is obtained with $f_2 = 1$: in this case, Proposition 6 gives

$$\tilde{k}_\rho \geq n - f_1 - \frac{f_1}{n - 1}.$$

Therefore, if $f_1 \leq n - 2$, then we have $\tilde{k}_\rho \geq n - f_1$. In particular, in a system with at least 3 processes and heard-of sets of cardinality $n - 1$ ($f_1 = f_2 = 1$), we can emulate macro-rounds with kernels of size at least $n - 1$, and so the global kernel of f macro-rounds has a membership of over $n - f$ processes. Considering that the communication predicates defined by $\forall r > 0, \forall p \in \Pi : |HO(p, r)| \geq n - 1$ and $|K| \geq n - f$ are the HO counterparts of asynchronous systems with at most one crash failure and synchronous systems with at most f send omission failures, respectively, we derive the following result relating synchronous and asynchronous systems:

Corollary 2 *Asynchronous message-passing systems with at most one crash failure can implement the first f rounds of a synchronous system with at most f send omission failures among $n \geq 3$ processes.*

A similar result is shown by Gafni [22] for asynchronous atomic-snapshot shared memory systems with at most one crash failure. Note that the very elegant reduction of the omission failure lower bound to the asynchronous impossibility result [20] that Gafni derives from his result can also be spun off from Corollary 2.

4.2 Translating no split rounds into nek rounds

We now show that $\mathcal{P}_{nosplit}$ and $\mathcal{P}_{nekrounds}$ are actually equivalent. Clearly, $\mathcal{P}_{nekrounds}$ implies $\mathcal{P}_{nosplit}$, and so we have $\mathcal{P}_{nekrounds} \geq \mathcal{P}_{nosplit}$. To prove that $\mathcal{P}_{nosplit} \geq \mathcal{P}_{nekrounds}$, we present a $\lambda(n)$ round translation, where $\lambda(n)$ is the integer satisfying $2^{\lambda(n)-1} < n \leq 2^{\lambda(n)}$, which emulates nek macro-rounds from no split rounds. This translation, which appears in Algorithm 2, is an extension from 2 to $\lambda(n)$ of Algorithm 1.

Each macro-round consists of $\lambda(n)$ consecutive rounds. We fix such a macro-round ρ , and we denote $r_1, \dots, r_{\lambda(n)}$ the sequence of rounds that form ρ .⁵ Each process p maintains a variable $Listen_p$, which is contained in Π and is equal to $HO(p, r_1)$ at the end of round r_1 . In the following rounds, p sends the current value of $Listen_p$ to all and then computes the new $Listen_p$ as the union of the $Listen_q$'s it has just received. That is, at each round $r, r_2 \leq r \leq r_{\lambda(n)}$, p sets

$$Listen_p := \bigcup_{q \in HO(p, r)} Listen_q.$$

Theorem 2 *Algorithm 2 is a $\lambda(n)$ round translation of $\mathcal{P}_{nosplit}$ into $\mathcal{P}_{nekrounds}$, and so we have $\mathcal{P}_{nosplit} \simeq \mathcal{P}_{nekrounds}$.*

Algorithm 2 Translating no split rounds into nek rounds

- 1: **Initialization:**
 - 2: $Listen_p \in 2^V$, initially empty
 - 3: $NewHO_p \in 2^V$, initially empty
 - 4: **Round r :**
 - 5: S_p^r :
 - 6: send $\langle Listen_p \rangle$ to all processes
 - 7: T_p^r :
 - 8: **if** $r \equiv 1 \pmod{\lambda(n)}$ **then**
 - 9: $Listen_p := HO(p, r)$
 - 10: **else**
 - 11: $Listen_p := \bigcup_{q \in HO(p, r)} Listen_q$
 - 12: **if** $r \equiv 0 \pmod{\lambda(n)}$ **then**
 - 13: $NewHO_p := Listen_p$
-

⁵ Precisely, we have $r_1 = \lambda(n)(\rho - 1) + 1, \dots, r_{\lambda(n)} = \lambda(n)\rho$.

Proof Condition E1 trivially follows from the code of Algorithm 2 (lines 9 and 11). We now prove E2. For that, consider the directed graphs G_i induced by the heard-of sets at round r_i . Let G_i^* denote the directed graph whose vertices are the processes in Π , and there is an edge from p to q iff there exists a chain of $i + 1$ processes x_1, \dots, x_{i+1} from $x_1 = p$ to $x_{i+1} = q$ such that

$$x_2 \in HO(x_1, r_{\lambda(n)}), x_3 \in HO(x_2, r_{\lambda(n)-1}), \dots, \\ \text{and } x_{i+1} \in HO(x_i, r_{\lambda(n)-i+1}).$$

Clearly, $G_1^* = G_{\lambda(n)}$, and $Listen_p$ at the end of round $r_{\lambda(n)}$ is the set of p 's in-neighbours in $G_{\lambda(n)}^*$:

$$Listen_p^{(r_{\lambda(n)})} = \{q \in \Pi : (q, p) \text{ is an edge of } G_{\lambda(n)}^*\}.$$

We now prove the following lemma:

Lemma 1 *For any index $i \in \{1, \dots, \lambda(n)\}$, there is at least one common in-neighbour to any subset of 2^i processes in the graph G_i^* .*

Proof By induction on i .

Basis: $i = 1$. We have $G_1^* = G_{\lambda(n)}$, and the lemma coincides with the no split predicate.

Inductive step: Suppose that $i \geq 2$ and the lemma holds in G_{i-1}^* . Let $\{p_1, \dots, p_{2^i}\}$ be any subset of 2^i processes. By inductive hypothesis, $p_1, \dots, p_{2^{i-1}}$ have a common in-neighbour x_1 in G_{i-1}^* , and $p_{2^{i-1}+1}, \dots, p_{2^i}$ have a common in-neighbour x_2 in G_{i-1}^* . Since the no split predicate holds at each round, x_1 and x_2 have a common in-neighbour in $G_{\lambda(n)-i+1}$, no matter whether $x_1 = x_2$ or not; let x denote this node. By definition of G_i^* , x is a common in-neighbour to p_1, \dots, p_{2^i} in this graph. \square Lemma 1

Condition E2 directly follows from Lemma 1 as the special case $i = \lambda(n)$ since $n \leq 2^{\lambda(n)}$. \square

Note that since \mathcal{P}_{HO}^{maj} implies $\mathcal{P}_{nosplit}$, Algorithm 2 is a $\lambda(n)$ round translation of \mathcal{P}_{HO}^{maj} into $\mathcal{P}_{nekrounds}$. Thus we get another proof of Corollary 1, but the translation requires $\lambda(n)$ rounds instead of two rounds in Algorithm 1.

Combining Theorems 1 and 2, we get the following corollary:

Corollary 3 *Let \mathcal{P} be a communication predicate satisfying $\mathcal{P} \succeq \mathcal{P}_{nosplit}$. Then the following assertions are equivalent:*

1. *There is an algorithm A such that $M = (A, \mathcal{P})$ solves Consensus;*
2. $\mathcal{P} \succeq \mathcal{P}_{sp_unif}^*$.

4.3 A translation achieving space uniformity

Our third translation achieves space uniformity under the condition of original non-empty global kernels. More precisely, we give a $f + 1$ round translation of \mathcal{P}_K^f into $\mathcal{P}_{sp_unif}^f \wedge \mathcal{P}_K^f$.

Processes propagate and collect all the heard-of sets that they have ever seen during $f + 1$ consecutive rounds. At the end of the macro-round, p 's new heard-of set is the intersection of the sets of process names that p has just collected at the last round. Formally, each macro-round consists of $f + 1$ consecutive rounds. Each process p maintains three variables $Listen_p$, $Known_p$, and $NewHO_p$, which are all contained in Π and are equal to Π , $\{p\}$, and \emptyset at the beginning of each macro-round, respectively. At each round, p listens to process q only if it hears of q at all the previous rounds of the macro-round, and so p sets:

$$Listen_p := Listen_p \cap HO(p, r).$$

Moreover, during the f first rounds of any macro-round, each process p collects the names of all the processes it hears of in its variable $Known_p$; for that, it sends $Known_p$ to all processes and then sets:

$$Known_p := Known_p \cup \left(\bigcup_{q \in Listen_p} Known_q \right).$$

At the last round of any macro-round, p computes the intersection (instead of the union as in the previous rounds of the macro-round) of the sets $Known_q$ it has just collected. The code of the translation is given below (see Algorithm 3).

Algorithm 3 Translation for space uniformity

```

1: Initialization:
2:    $Listen_p \in 2^V$ , initially  $\Pi$ 
3:    $NewHO_p \in 2^V$ , initially  $\emptyset$ 
4:    $Known_p \in 2^V$ , initially  $\{p\}$ 

5: Round  $r$ :
6:    $S_p^r$ :
7:     send  $\langle Known_p \rangle$  to all processes
8:    $T_p^r$ :
9:      $Listen_p := Listen_p \cap HO(p, r)$ 
10:    if  $r \not\equiv 0 \pmod{f+1}$  then
11:       $Known_p := Known_p \cup \left( \bigcup_{q \in Listen_p} Known_q \right)$ 
12:    else
13:       $NewHO_p := \bigcap_{q \in Listen_p} Known_q$ 
14:       $Listen_p := \Pi$ 
15:       $Known_p := \{p\}$ 

```

We fix a macro-round ρ and introduce some notation relative to ρ . Let r_1, \dots, r_{f+1} denote the sequence of the $f + 1$ rounds that form ρ . Recall that $K(\rho)$ denotes the kernel of macro-round ρ , i.e., $K(\rho) = \bigcap_{r=r_1}^{r_{f+1}} K(r)$.

We say that process p *knows process s at round r* if $s \in Known_p^{(r)}$. If $s \in Known_p^{(r)} \setminus Known_p^{(r-1)}$, $q \in Listen_p^{(r)}$, and $s \in Known_q^{(r-1)}$, then we say that p *hears of s from q at round r* . Finally, process s is said to be *good* (at macro-round

ρ) if s is known by all processes at round r_f ; otherwise s is bad. In other words, the set of good processes is defined by

$$Good = \bigcap_{p \in \Pi} Known_p^{(r_f)}.$$

Thus at line (13), every process computes a local approximation of the set of good processes.

We are going to prove that if $K(\rho)$ contains at least $n - f$ processes, then at the end of any macro-round the *NewHO*'s are all identical and contain $K(\rho)$. For that, we start with some preliminary assertions, where as previously, $X_p^{(r)}$ denotes the value of X_p just at the end of round r , except for $X_p = Listen_p$, $X_p = Known_p$, and $r = r_{f+1}$: in this case, $Listen_p^{(r_{f+1})}$ and $Known_p^{(r_{f+1})}$ denote the values of $Listen_p$ and $Known_p$ in round r_{f+1} just before lines 14 and 15, respectively.

Lemma 2 $K(\rho) = \bigcap_{p \in \Pi, r \in \{r_1, \dots, r_{f+1}\}} Listen_p^{(r)}$.

Proof Immediate from the definition of $Listen_p^{(r)}$. □

Lemma 3 Any process p in $K(\rho)$ is a good process.

Proof Let p be any process in $K(\rho)$. By lines (15) and (11), it follows that all processes know p at the end of round r_1 . Moreover, each $Known_p$ variable is non decreasing in each macro-round. This shows that p is a good process. □

Lemma 4 If process p hears of some process s at round r_k , then there exist $k - 1$ processes p_1, \dots, p_{k-1} , each different from p and s , such that p_1 hears of s from s at round r_1 , p_2 hears of s from p_1 at round r_2 , ..., p_{k-1} hears of s from p_{k-2} at round r_{k-1} , and p hears of s from p_{k-1} at round r_k . Moreover, processes p_1, \dots, p_{k-2} , and s are all in $coK(\rho)$.

Proof Since p hears of process s at round r_k , there exists some process p_{k-1} such that $p_{k-1} \in Listen_p^{(r_k)}$ and $s \in Known_{p_{k-1}}^{(r_{k-1})}$. Since $Listen_p$ is non-increasing, $p_{k-1} \in Listen_p^{(r_{k-1})}$. This implies that p_{k-1} hears of s at round r_{k-1} since p does not know s at this round. In turn, there exists some process p_{k-2} such that $p_{k-2} \in Listen_{p_{k-1}}^{(r_{k-1})}$, and $s \in Known_{p_{k-2}}^{(r_{k-2})}$. From $s \in Known_{p_{k-2}}^{(r_{k-2})}$ and $s \notin Known_p^{(r_{k-1})}$, we deduce that $p_{k-2} \notin Listen_p^{(r_{k-1})}$. By Lemma 2, we have $p_{k-2} \in coK(\rho)$.

Step by step, we exhibit $k - 1$ processes p_1, \dots, p_{k-1} such that for any index i , $1 \leq i \leq k - 1$, $s \notin Known_{p_i}^{(r_{i-1})}$, $s \in Known_{p_i}^{(r_i)}$, and $p_{i-1} \in Listen_{p_i}^{(r_i)}$.

For any index i such that $2 \leq i \leq k - 2$, we have both $s \in Known_{p_{i-1}}^{(r_{i-1})}$ and $s \notin Known_{p_{i+1}}^{(r_i)}$.

Therefore, $p_{i-1} \notin Listen_{p_{i+1}}^{(r_i)}$. By Lemma 2, we deduce that p_{i-1} belongs to $coK(\rho)$. Similarly, we have $s \notin Known_{p_2}^{(r_1)}$, and so s belongs to $coK(\rho)$, too. From $p_{k-2} \in coK(\rho)$, it follows that all the processes p_1, \dots, p_{k-2} , and s are in $coK(\rho)$. □

Lemma 5 If process p knows some bad process s at the end of round r_{f+1} , then p has heard of s by the end of the round r_f , i.e.,

$$s \in Known_p^{(r_{f+1})} \wedge s \notin Good \Rightarrow s \in Known_p^{(r_f)}.$$

Proof Let s be a bad process; so there exists some process q such that $s \notin Known_q^{(r_f)}$. Suppose for contradiction that p hears of s at round r_{f+1} . By Lemma 4, there are f processes p_1, \dots, p_f each different from both p and s such that p hears of s from p_f at round r_{f+1} , and processes p_1, \dots, p_{f-1} , and s are all in $coK(\rho)$. Since $s \notin Known_q^{(r_f)}$, $Listen_q$ contains neither p nor p_f at this round. Therefore, p and p_f are also in $coK(\rho)$, which contradicts the fact that $coK(\rho)$ is of size at most f . □

Lemma 6 A process is good iff it is known at the end of round r_f by some process in the kernel, i.e., $s \in Good \Leftrightarrow \exists p \in K(\rho) : s \in Known_p^{(r_f)}$.

Proof By definition, a good process is known by all processes at round r_f .

Conversely, let s be any process known by some process p in $K(\rho)$ at round r_f . Assume, for the sake of contradiction, that s is bad. Since p is in $K(\rho)$, every process q receives a message from p at round r_{f+1} , and so $Known_q$ contains s at the end of round r_{f+1} . By Lemma 5, we deduce that every process already knows s at round r_f . This contradicts that s is a bad process. □

Lemma 7 For any process p , at the end of round r_{f+1} , $NewHO_p$ is composed of all the good processes, i.e., $NewHO_p^{(r_{f+1})} = Good$.

Proof Obviously, we have $Good \subseteq NewHO_p^{(r_{f+1})}$.

Conversely, let s be any process in $NewHO_p^{(r_{f+1})}$; s is known at round r_f by all the processes in $Listen_p^{(r_{f+1})}$, and in particular by those in $K(\rho)$ by Lemma 2. From Lemma 6, it follows that s is a good process since $K(\rho)$ is non-empty. □

Lemma 7 says that all the $NewHO_p$'s are equal after $f + 1$ rounds, and so the collection of the *NewHO*'s satisfies \mathcal{P}_{sp_unif} at the end of each macro-round. Moreover, Lemma 3 implies that the translation preserves kernels, and so emulates $\mathcal{P}_K^f \wedge \mathcal{P}_{sp_unif}$. Since E1 is clearly guaranteed, we have proved the following theorem:

Theorem 3 Algorithm 3 is a $f + 1$ translation of \mathcal{P}_K^f into $\mathcal{P}_K^f \wedge \mathcal{P}_{sp_unif}$, and so we have $\mathcal{P}_K^f \simeq \mathcal{P}_K^f \wedge \mathcal{P}_{sp_unif}$.

Combining the latter theorem with Proposition 4, we derive a $f + 1$ round algorithm A such that the HO machine (A, \mathcal{P}_K^f) solves Consensus. Thus we check that at least for nek machines, the strong termination requirement, namely “every process eventually decides”, does not make the Consensus specification harder to solve. Note that the first f rounds of A are identical to those of the *FloodSet* algorithm [29] which is a well-known Consensus algorithm devised for synchronous systems with at most f crash failures. These two algorithms only differ in round $f + 1$: Algorithm 3 computes the intersection of the $Known_p$'s instead of union in *FloodSet*. Hence, substituting intersection for union just at the last round is sufficient to guarantee a general agreement among *all* processes under the only communication predicate \mathcal{P}_K^f (without regularity).

Interestingly, if we substitute

$$NewHO_p := NewHO_p \cap (\bigcap_{q \in Listen_p} Known_q)$$

for

$$NewHO_p := \bigcap_{q \in Listen_p} Known_q$$

at line (13) in Algorithm 3, the resulting algorithm translates \mathcal{P}_K^f into $\mathcal{P}_K^f \wedge \mathcal{P}_{sp_unif} \wedge \mathcal{P}_{reg}$. Considering that the communication predicates \mathcal{P}_K^f and $\mathcal{P}_K^f \wedge \mathcal{P}_{reg}$ are the HO counterparts of synchronous systems with at most f send omission failures and with at most f crash failures respectively, we get an automatic procedure which both guarantees space uniformity and masks send omissions into crash failures.

4.4 A two round translation for increasing time uniformity

We now describe a two round translation that increases time uniformity in the sense that it emulates regular runs. This translation can be viewed as a refinement of our first translation (Algorithm 1), with in addition a mechanism for the transition from a macro-round to the next one which guarantees regularity. The basic idea of this mechanism is at each macro-round and for each process to compute an approximation of the kernel of the previous macro-round.

More precisely, each process p maintains a variable $ApproxK_p$ whose initial value is Π . At the end of round 2ρ , p sets

$$ApproxK_p := \bigcap_{q \in HO(p, 2\rho)} \left(HO(q, 2\rho - 1) \cap ApproxK_q^{(\rho-1)} \right),$$

where $ApproxK_q^{(\rho-1)}$ denotes the value of $ApproxK_q$ at the end of the macro-round $\rho - 1$. The heard-of set at macro-

round ρ for process p is now defined by:

$$NewHO_p := \bigcup_{q \in HO(p, 2\rho)} \left(HO(q, 2\rho - 1) \cap ApproxK_q^{(\rho-1)} \right).$$

The resulting algorithm is called Algorithm 4.

Algorithm 4 Translating no split rounds into regular rounds

- 1: **Initialization:**
 - 2: $ApproxK_p \in 2^V$, initially equal to Π
 - 3: $NewHO_p \in 2^V$, initially empty
 - 4: **Round** $r = 2\rho$:
 - 5: S_p^r :
 - 6: send $(ApproxK_p \cap HO(p, r - 1))$ to all processes
 - 7: T_p^r :
 - 8: $ApproxK_p := ApproxK_p \cap \left(\bigcap_{q \in HO(p, r)} ApproxK_q \cap HO(q, r - 1) \right)$
 - 9: $NewHO_p := \bigcup_{q \in HO(p, r)} ApproxK_q \cap HO(q, r - 1)$
-

Theorem 4 Algorithm 4 translates $\mathcal{P}_{nosplit}$ into \mathcal{P}_{reg} , and so we have $\mathcal{P}_{nosplit} \succeq \mathcal{P}_{reg}$. Moreover, it preserves global kernels.

Proof Condition E1 immediately follows from the code of Algorithm 4, line 9. For the same reason, the algorithm preserves kernels. We now prove E2. According to the code of the algorithm, if $x \notin NewHO_p^{(\rho)}$, then for any s in $HO(q, 2\rho)$, we have $x \notin HO(s, 2\rho - 1) \cap ApproxK_s^{(\rho-1)}$.

Because of the no split predicate, for any process y , $HO(y, 2\rho)$ intersects $HO(q, 2\rho)$, and so there exists $s \in HO(y, 2\rho)$ such that $x \notin HO(s, 2\rho - 1) \cap ApproxK_s^{(\rho-1)}$. Hence $x \notin ApproxK_y^{(\rho)}$. It follows that for any process p , $x \notin NewHO_p^{(\rho+1)}$. In other words, we have showed that

$$\forall p \in \Pi : NewHO_p^{(\rho+1)} \subseteq \bigcap_{q \in \Pi} NewHO_q^{(\rho)},$$

i.e., regularity holds.

Moreover, by definition of $ApproxK_p$, we easily get

$$\forall p \in \Pi : \bigcap_{r=1}^{\rho} K(2r - 1) \subseteq ApproxK_p^{(\rho)}.$$

It follows that Algorithm 4 preserves global kernels. \square

From the latter point, we derive the following corollary:

Corollary 4 $\forall f \in \{1, \dots, n - 1\} : \mathcal{P}_K^f \simeq \mathcal{P}_K^f \wedge \mathcal{P}_{reg}$.

Since the communication predicates \mathcal{P}_K^f and $\mathcal{P}_K^f \wedge \mathcal{P}_{reg}$ are the HO counterparts of synchronous systems with at most f send omission failures, and synchronous systems with at most f crash failures, respectively [22], Algorithm 4 provides a general method to convert synchronous algorithms

tolerant of f crash failures into ones tolerant of f omission senders.

Algorithm 4 is similar to the two round translation given by Neiger and Toueg [33] to mask send omission into crash failures. The only difference between both lies in the processes that are in charge of stopping information transmission: In Neiger and Toueg’s algorithm, upon learning it is faulty at some point, process p self censors for the rest of the computation whereas in Algorithm 4, p sends the same messages but the other processes that detect p is faulty do not hear of p anymore.

Algorithms 1 and 4 can be combined into a three round translation that increases both space and time uniformity. In this new translation, each process p inductively computes $NewHO_p$ and $ApproxK_p$ as follows:

$$NewHO_p^{(\rho)} := \bigcup_{q \in HO(p, 3\rho)} \left(\bigcup_{s \in HO(q, 3\rho-1)} HO(s, 3\rho - 2) \cap ApproxK_s^{(\rho-1)} \right),$$

and

$$ApproxK_p := \bigcap_{q \in HO(p, 3\rho)} \left(\bigcup_{s \in HO(q, 3\rho-1)} HO(s, 3\rho - 2) \cap ApproxK_s^{(\rho-1)} \right).$$

In this way, the first f rounds of a synchronous system with at most f crash failures can be implemented from an asynchronous message-passing system with at most one crash failure. This three round simulation is identical to the one given by Gafni [22] from asynchronous atomic-snapshot shared memory systems: the last two rounds of the simulation actually correspond to the two round *adopt-commit* protocol in [22].

4.5 Comparing communication predicates

In the previous sections, we saw several interrelationships between some basic communication predicates. These relations and those that are directly derived from the implication relations are illustrated in Fig. 1 as follows: there is a direct edge from \mathcal{P} to \mathcal{P}' if $\mathcal{P} \geq \mathcal{P}'$. We adopt the following notation: $\mathcal{P}_K^{maj} = \mathcal{P}_K^{\lfloor \frac{n-1}{2} \rfloor}$, and

$$\mathcal{P}_{(majHO)^\infty} :: \forall r > 0, \exists r_0 \geq r, \forall p \in \Pi : |HO(p, r_0)| > n/2.$$

The dashed line represents the “Consensus line”: a communication predicate \mathcal{P} is above the line iff Consensus is solvable under \mathcal{P} . The figure is completed with the predicate $\mathcal{P}_{LastVoting^{rc}}$ defined in Sect. 5.5.

5 Consensus and general HO machines

We now examine general HO machines, some with empty kernel rounds, that solve Consensus. To do so, we first revisit various classical Consensus algorithms devised for asynchronous or partially synchronous systems. For coordinator-based algorithms, we introduce a generalization of HO machines, the *Coordinated HO machines* (or *CHO machines* for short).

HO and CHO machine formalisms enable us to express well-known Consensus algorithms in a fairly concise and elegant way, and so to extract the algorithmic schemes on which they are based. This not only gives some new insights into these Consensus algorithms, but also allows us to design new ones that are interesting in practice since they are correct under quite realistic conditions. Moreover, it is striking to see how easy it is to determine simple conditions that ensure the correctness of these algorithms from their HO or CHO counterparts.

5.1 A Consensus algorithm à la Ben-Or: the *UniformVoting* algorithm

First, we present a Consensus algorithm that to the best of our knowledge, has not yet been described in the literature. It can be viewed as a deterministic version of the Ben-Or algorithm [1,35]. We call it the *UniformVoting* algorithm, see Algorithm 5.

As for all the other algorithms described in Sect. 5, *UniformVoting* is organized into *phases*.⁶ A *UniformVoting* phase consists of two rounds. Every process p maintains a variable x_p containing a value in V , initially equal to p ’s initial value. Process p broadcasts x_p at the first round of each phase, and then adopts the smallest value it has just received. Then, p votes for value v if it has not heard that some process has started the phase with another value; otherwise, p does not cast a vote. At the second round, p sends v or “?” to all, accordingly. In the same message, it sends again the current value of x_p . If each message that p receives at the second round contains a vote for v , then p decides v (this is why we call this algorithm *UniformVoting*). If p receives some values v different from “?”, then it chooses one such value arbitrarily and adopts it for the next phase; otherwise, p adopts the smallest value of the x_q ’s it has just received.

We now argue that if no round is split, then no process can make a bad decision (agreement). Then we prove that termination is enforced by just one uniform round, which is

⁶ A phase consists of a fixed number of consecutive rounds. Basically, there is no difference between a phase and a macro-round. We have preferred the term “macro-round” for translations because it seems us more suggestive in this context, but here we use the classical terminology of “phase”.

Algorithm 5 The *UniformVoting* algorithm

```

1: Initialization:
2:    $x_p := v_p$   ( $v_p$  is the initial value of  $p$ )
3:    $vote_p \in V \cup \{?\}$ , initially ?
4: Round  $r = 2\phi - 1$ :
5:    $S_p^r$ :
6:   send  $\langle x_p \rangle$  to all processes
7:    $T_p^r$ :
8:    $x_p :=$  smallest  $v$  received
9:   if all the values received are equal to  $v$  then
10:     $vote_p := v$ 
11: Round  $r = 2\phi$ :
12:    $S_p^r$ :
13:   send  $\langle x_p, vote_p \rangle$  to all processes
14:    $T_p^r$ :
15:   if at least one  $\langle *, v \rangle$  with  $v \neq ?$  is received then
16:      $x_p := v$ 
17:   else
18:      $x_p :=$  smallest  $w$  from  $\langle w, ? \rangle$  received
19:   if all the messages received are
20:     equal to  $\langle *, v \rangle$  with  $v \neq ?$  then
21:     DECIDE( $v$ )
22:    $vote_p := ?$ 

```

guaranteed by the communication predicate:

$$\mathcal{P}_{(unif)^\infty} :: \forall r > 0, \exists r_0 \geq r, \forall p, q \in \Pi^2 : \\ HO(p, r_0) = HO(q, r_0).$$

Theorem 5 The HO machine consisting of the *UniformVoting* algorithm and the predicate $\mathcal{P}_{nosplit} \wedge \mathcal{P}_{(unif)^\infty}$ solves Consensus.

Proof Integrity is trivially satisfied.

The proof of the agreement condition relies on the fact that if two processes p and q vote for v and v' at the same phase, then predicate $\mathcal{P}_{nosplit}$ ensures that $v = v'$. Moreover, predicate $\mathcal{P}_{nosplit}$ also guarantees that if some process decides v at round $r = 2\phi$, then all the x_p 's remain equal to v from round r .

For termination, let r_0 be a uniform round. There are two cases to consider.

1. Round r_0 is the first round of some phase ϕ_0 , i.e., $r_0 = 2\phi_0 - 1$. Therefore at round r_0 , either all processes vote for the same value v or no process votes.
2. Round r_0 is the second round of some phase ϕ_0 , i.e., $r_0 = 2\phi_0$.

In both cases, all the x_p 's are equal at the end of round $2\phi_0$, and every process has decided at the end of round $2\phi_0 + 2$. It follows that $\mathcal{P}_{(unif)^\infty}$, which is invariant by time translation (cf. Sect. 2.1) and guarantees one uniform round, enforces termination of the *UniformVoting* algorithm. \square

5.2 Coordinated HO machines

Numerous algorithms for Consensus are coordinator-based (e.g., the Consensus algorithms proposed by Dwork

et al. [17], Chandra and Toueg's algorithm [5], Paxos [27]). The correctness of these algorithms is guaranteed by some properties on coordinators: for example, termination in Paxos requires that during some phase, all processes hear of the coordinator of the phase. For such algorithms, we introduce the *Coordinated HO machine* (or *CHO machine* for short) for which algorithms refer to the notion of coordinators, and predicates deal not only with heard-of sets, but also with coordinators.

A CHO machine is a pair $M^c = (A, P)$ much like the ordinary HO machine. Reflecting the fact that the messages sent by a process p in a round of a CHO machine do not uniquely depend on the current state, but also on the identity of a *coordinator*, the message-sending function S_p^r is no longer a function from $states_p \times \Pi$ to M but instead a function

$$S_p^r : \Pi \times states_p \times \Pi \longrightarrow M.$$

Similarly, the state of process p at the end of a round does not only depend on its current state and the collection of the messages it has just received, but also on the identity of its coordinator. So, the transition function T_p^r is now a function

$$T_p^r : states_p \times (M^\Pi)^* \times \Pi \longrightarrow states_p$$

where $(M^\Pi)^*$ denotes the set of partial vectors of elements of M , indexed by Π . The functions $(S_p^r)_{r>0}$ and $(T_p^r)_{r>0}$ define the *coordinated process* p , and the collection of coordinated processes is called a *coordinated algorithm*.

As for HO machines, at every round r , each process p (1) applies the message-sending function S_p^r to the current coordinator and the current state to generate the messages to be sent, and (2) applies the state-transition T_p^r to the current state and the incoming messages. The combination of the two steps is called a *coordinated round*, and p 's coordinator at r is denoted $Coord(p, r)$. Process p sets out to be its coordinator if $p = Coord(p, r)$.

We say that r is a *uniformly coordinated round* if

$$\forall p, q \in \Pi : Coord(p, r) = Coord(q, r)$$

and r is *well coordinated* if

$$\forall p \in \Pi : Coord(p, r) \in HO(p, r).$$

As we shall see in the next sections, uniformly and well coordinated rounds play a key role for guaranteeing correctness of coordinated Consensus algorithms.

A computation of a CHO machine is *uniformly coordinated from round* r_0 if any round r , $r \geq r_0$, is uniformly coordinated; a computation is *uniformly coordinated* if it is uniformly coordinated from the first round.

Usually, when algorithms are decomposed into phases, every process keeps the same coordinator during each whole

phase. The coordinator of process p during phase ϕ can thus be denoted by $Coord(p, \phi)$.

Given a set of initial states, one per process, each run of a CHO machine is not uniquely determined by the heard-of set collection, but also by the *coordinator collection* according to space and time, namely $(Coord(p, r))_{p \in \Pi, r > 0}$. A *CHO machine for Π* consists of a coordinated algorithm A and a predicate over both heard-of sets and coordinator collections, called a *communication-coordinator predicate*, which is invariant by time translation. For example, we shall consider CHO machines with the predicate:

$$\forall r > 0, \exists r_0 \geq r, \forall p, q \in \Pi^2 : \\ Coord(p, r_0) = Coord(q, r_0) \wedge Coord(p, r_0) \in HO(p, r_0)$$

or equivalently,

$$\forall r > 0, \exists r_0 \geq r, \forall p, q \in \Pi^2 : \\ Coord(p, r_0) = Coord(q, r_0) \wedge Coord(p, r_0) \in K(r_0).$$

Finally, the notion of what it means for a CHO machine to solve a problem is similar to the one for an HO machine.

Similarly to the heard-of sets, the way processes determine the name of their coordinators is not specified in the HO formalism: it may be the result of some computation (in other words, the CHO machine is emulated by an ordinary HO machine), or processes may use some external devices (physical devices or oracles) that are capable of reporting the name of a coordinator to each process. Most of the CHO machines that we shall consider can be simulated by ordinary HO machines, and so this generalization does not seem to lead to a more powerful computational model (as explained above, the basic motivation for introducing CHO machines is to devise Consensus algorithms and to state conditions for their correctness in a simpler way). In particular, we can adopt an “off-line” strategy, usually called the *rotating coordinator* strategy, which consists in selecting $Coord(p, r) = p_{1+r \bmod n}$ for every process p in Π .

When $\Pi = \{p_1, \dots, p_n\}$. Note that fixing the rotating coordinator strategy, any CHO machine reduces to an HO machine.

With the rotating coordinator strategy, agreement on the name of a coordinator is for free, that is every round is uniformly coordinated. On the other hand, the on-line strategy that consists in selecting p 's coordinator in its heard-of set provides well coordinated rounds for free (in the case heard-of sets do not vary “too much” in time). A critical point is to achieve rounds which are both uniformly and well coordinated.

5.3 The *CoordUniformVoting* machine for Consensus

When looking closer at the *UniformVoting* machine, we may think to ensure uniformity of one round, and so termination,

by the help of coordinators: at the beginning of each phase, coordinators are in charge to make the x_p 's values uniform. More precisely, to each phase ϕ , we add a preliminary round in which every process p that sets out to be coordinator of phase ϕ (i.e., $p = Coord(p, \phi)$) broadcasts the value of its variable x_p . Upon receiving a message with value v from $Coord(q, \phi)$, process q adopts this value for x_q . Actually, the additional round allows us to simplify the two rounds of *UniformVoting*: each process p just sends its vote instead of sending both its vote and the value of x_p . This yields an algorithm that we call *CoordUniformVoting* (see Algorithm 6).

Since the decision of one process at some round 2ϕ of *UniformVoting* entails that all the x_p 's are equal, *CoordUniformVoting* still satisfies integrity and agreement. If at some phase ϕ_0 , all processes agree on some coordinator's name c , and this coordinator is in the kernel of ϕ_0 , then every process hears of and adopts x_c 's value. In that case, all processes decide on this value at the end of phase ϕ_0 . This proves the following theorem:

Theorem 6 *The CHO machine consisting of the CoordUniformVoting algorithm and the predicate that guarantees no split round and uniformly and well coordinated phases infinitely often solves Consensus.*

This exemplifies how the use of coordinators transforms the requirement of a non-trivial uniform round into the one of agreeing on the name of some process in the kernel. Note that agreement on the coordinator of a phase ϕ may be achieved by a leader election algorithm. At that point, the question is whether the elected process is actually in the kernel of round $3\phi - 2$.

Instead of using a leader election algorithm, we can adopt the rotating coordinator strategy. We denote by *CoordUniformVoting^c* the resulting algorithm. With such a coordinator strategy, having the leader in the kernel at some point in computation is ensured by

$$\exists \phi_0 > 0 : \bigcap_{i=1}^n K(\phi_0 + i) \neq \emptyset.$$

Thus using the rotating coordinator strategy, we substitute uniformity of one round in the *UniformVoting* machine for some “temporal stability” guaranteeing a sufficiently long period with a non-empty kernel.

Algorithm 6 can be simplified by observing that lines 9 to 16 can be removed, if at line 17 the value received is assigned to $vote_p$ (see Algorithm 7).

5.4 The *DLS* algorithm

The algorithms described up to now work correctly only if some invariant properties for the *HO*'s are satisfied (e.g., $\mathcal{P}_{n\text{ekrounds}}$ or $\mathcal{P}_{\text{nosplit}}$). Taking a closer look at these algorithms, it turns out that the safety conditions of Consensus

Algorithm 6 The *CoordUniformVoting* algorithm

```

1: Initialization:
2:    $x_p := v_p$    $\{v_p \text{ is the initial value of } p\}$ 
3:    $vote_p \in V \cup \{?\}$ , initially ?
4: Round  $r = 3\phi - 2$ :
5:    $S_p^r$ :
6:   if  $p = \text{Coord}(p, \phi)$  then
7:     send  $\langle x_p \rangle$  to all processes
8:    $T_p^r$ :
9:   if some message  $\langle v \rangle$  is received
10:    from  $\text{Coord}(p, \phi)$  then
11:      $x_p := v$ 
12: Round  $r = 3\phi - 1$ :
13:    $S_p^r$ :
14:     send  $\langle x_p \rangle$  to all processes
15:    $T_p^r$ :
16:   if all the values received are equal to  $v$  then
17:      $vote_p := v$ 
18: Round  $r = 3\phi$ :
19:    $S_p^r$ :
20:     send  $\langle vote_p \rangle$  to all processes
21:    $T_p^r$ :
22:   if at least one  $\langle v \rangle$  with  $v \neq ?$  is received then
23:      $x_p := v$ 
24:   if all the messages received are
25:     equal to  $\langle v \rangle$  with  $v \neq ?$  then
26:     DECIDE( $v$ )
27:      $vote_p := ?$ 

```

Algorithm 7 Simplified *CoordUniformVoting* algorithm

```

1: Initialization:
2:    $x_p := v_p$    $\{v_p \text{ is the initial value of } p\}$ 
3:    $vote_p \in V \cup \{?\}$ , initially ?
4: Round  $r = 2\phi - 1$ :
5:    $S_p^r$ :
6:   if  $p = \text{Coord}(p, \phi)$  then
7:     send  $\langle x_p \rangle$  to all processes
8:    $T_p^r$ :
9:   if received  $\langle v \rangle$  from  $\text{Coord}(p, \phi)$  then
10:     $vote_p := v$ 
11: Round  $r = 2\phi$ :
12:    $S_p^r$ :
13:     send  $\langle vote_p \rangle$  to all processes
14:    $T_p^r$ :
15:   if at least one  $\langle v \rangle$  with  $v \neq ?$  is received then
16:      $x_p := v$ 
17:   if all the messages received are
18:     equal to  $\langle v \rangle$  with  $v \neq ?$  then
19:     DECIDE( $v$ )
20:      $vote_p := ?$ 

```

may be violated if there are some “bad” periods during which these predicates do not hold. Thereby, such algorithms cannot be used in systems with message losses (even very rare), which considerably limits the scope of these Consensus algorithms.

In a seminal paper [17], Dwork, Lynch, and Stockmeyer showed how to cope with such bad periods, and designed an algorithm, that we call *DLS*, which solves Consensus if a “sufficiently long” good period occurs. The basic idea of this algorithm is to satisfy safety conditions no matter how

badly processes communicate, that is even if many failures occur in the system.

The *DLS* algorithm was originally described in an HO-like style [17]. Rounds are grouped into phases, where each phase ϕ consists of four rounds. The algorithm includes the rotating coordinator strategy, and so each phase ϕ is led by a unique coordinator. We refer the reader to [17] for the complete description of *DLS*.

Dwork et al. [17] proved that their algorithm never violates integrity and agreement. Moreover, they showed that it terminates if “there is a majority of correct processes, and there exists some round GST, such that all messages sent from correct processors at round GST or afterwards are delivered during the round at which they were sent.” When shifting GST after the last time a process crashes, the above condition actually corresponds to eventual space-time uniformity:

$\exists \text{ GST} > 0, \exists \Pi_0$ s.t. $|\Pi_0| > n/2$:

$$\forall p \in \Pi, \forall r \geq \text{GST} : HO(p, r) = \Pi_0$$

which in terms of failure detectors coincides with $\diamond\mathcal{P}$ [5].

As already mentioned in [17], this termination requirement can be drastically weakened: a single well coordinated phase without “too many” transmission failures entails termination. Formally, the *DLS* algorithm is still correct when replacing eventual space-time uniformity by the following communication predicate:

$\forall \phi > 0, \exists \phi_0 \geq \phi, \exists \Pi_0$ s.t. $|\Pi_0| > n/2$:

$$(\forall p \in \Pi, \forall r \in \phi_0 : HO(p, r) = \Pi_0) \wedge (p_{1+\phi_0 \bmod n} \in \Pi_0).$$

Variant of DLS: Interestingly, the safety conditions of Consensus, namely integrity and agreement, still hold for any coordinator strategy, even when several processes lead the same phase. In other words, the CHO extension of *DLS*, that we denote *CoordDLS*, also satisfies integrity and agreement whatever the communication-coordinator predicate we consider. For termination, we just have to substitute the condition

$$(\forall p, q \in \Pi^2 : \text{Coord}(p, \phi_0) = \text{Coord}(q, \phi_0)) \wedge$$

$$(\forall p \in \Pi : \text{Coord}(p, \phi_0) \in \Pi_0).$$

for the condition $p_{1+\phi_0 \bmod n} \in \Pi_0$ in the above communication predicate. Thus this variant of *DLS* solves Consensus under the condition that there exists some uniform phase⁷ ϕ_0 whose kernel $K(\phi_0)$ is a majority set, and which is led by a single process (coordinator) in $K(\phi_0)$.

⁷ By *uniform phase*, we mean that each round of this phase is uniform.

5.5 A CHO algorithm “à la Paxos”

The *DLS* algorithm is based on the rotating coordinator paradigm, which ensures permanent agreement on the coordinator, but as mentioned above, it supports a more flexible coordinator strategy. The idea of using various policies for determining coordinators has been introduced by Lamport in the *Paxos* algorithm [27]. However, the idea is not followed through to the end in the latter algorithm: the first round of *Paxos* enforces the choice of a unique coordinator for the remaining rounds of the phase, and so the “Consensus core” in *Paxos* actually manages a single coordinator per phase.

We have observed here that *Paxos* is still safe even in the presence of multiple coordinators in the same phase. Indeed, multiple coordinators in the same phase can only prevent casting a vote, since this requires a coordinator to get a majority of (non-null) messages. By the majority condition, at most one coordinator per phase is able to cast a vote. Thus we design a new CHO algorithm, called *LastVoting* (Algorithm 8), which follows the basic line of *Paxos*, but manages possible multiple coordinators per phase. *LastVoting* is structured as *Paxos*, except the first round that is removed. This is because *LastVoting* proceeds in numerically consecutive phases, and so a phase automatically gets a chance to complete. Note that the first two rounds of every phase in *Paxos* are needed only when the coordinator changes. We always remove the first of these two rounds. The second of these two rounds is needed in *LastVoting* only when the coordinator changes. Agreement on a single coordinator (also called *leader*) is not achieved by the algorithm anymore, but is part of the conditions that guarantee termination.

The Consensus core in *Paxos* shares many common features with *DLS*: as for *DLS*, the coordinator of a *Paxos* phase does not cast a vote and misses its turn if it does not receive conclusive information from enough (namely a majority of the) processes. This is the basic reason why *DLS* and *Paxos* both tolerate link failures. On the other hand, the two algorithms differ in the values of the coordinators’ votes. In *DLS*, the coordinator of a phase votes for some value v if v is a *majority value* (i.e., it has heard that at least $n/2$ processes find v acceptable) whereas the coordinator of a *Paxos* phase votes for the *most recent value* it has heard of. This is why the coordinator of a *Paxos* phase can cast a vote (and so can make a decision) even if the preceding round is not uniform. In that respect, our *LastVoting* algorithm is similar to *Paxos*.

In the following theorem, we show that *LastVoting* is always safe (even in the presence of multiple coordinators at some phases), and we exhibit a very simple condition that enforces termination. Interestingly, the latter condition only involves *one* phase, and the corresponding communication-coordinator predicate is non-stable, contrary to the “ Ω condition”—classically supposed for *Paxos* termination—that

Algorithm 8 The LastVoting algorithm

```

1: Initialization:
2:  $x_p \in V$ , initially  $v_p$  ( $v_p$  is the initial value of  $p$ )
3:  $vote_p \in V \cup \{?\}$ , initially ?
4:  $commit_p$  a Boolean, initially false
5:  $ready_p$  a Boolean, initially false
6:  $ts_p \in \mathbb{N}$ , initially 0

7: Round  $r = 4\phi - 3$ :
8:  $S_p^r$  :
9:   send  $(x_p, ts_p)$  to  $Coord(p, \phi)$ 

10:  $T_p^r$  :
11:   if  $p = Coord(p, \phi)$  and
      number of  $(v, \theta)$  received  $> n/2$  then
12:     let  $\bar{\theta}$  be the largest  $\theta$  from  $(v, \theta)$  received
13:      $vote_p :=$  one  $v$  such that  $(v, \bar{\theta})$  is received
14:      $commit_p :=$  true

15: Round  $r = 4\phi - 2$ :
16:  $S_p^r$  :
17:   if  $p = Coord(p, \phi)$  and  $commit_p$  then
18:     send  $(vote_p)$  to all processes

19:  $T_p^r$  :
20:   if received  $(v)$  from  $Coord(p, \phi)$  then
21:      $x_p := v$ ;  $ts_p := \phi$ 

22: Round  $r = 4\phi - 1$ :
23:  $S_p^r$  :
24:   if  $ts_p = \phi$  then
25:     send  $(ack)$  to  $Coord(p, \phi)$ 

26:  $T_p^r$  :
27:   if  $p = Coord(p, \phi)$  and
      number of  $(ack)$  received  $> n/2$  then
28:      $ready_p :=$  true

29: Round  $r = 4\phi$ :
30:  $S_p^r$  :
31:   if  $p = Coord(p, \phi)$  and  $ready_p$  then
32:     send  $(vote_p)$  to all processes

33:  $T_p^r$  :
34:   if received  $(v)$  from  $Coord(p, \phi)$  then
35:     DECIDE( $v$ )
36:   if  $p = Coord(p, \phi)$  then
37:      $ready_p :=$  false
38:      $commit_p :=$  false

```

requires uniformly and well coordinated phases *permanently* from some point in the computation [7].

Theorem 7 *The HO machine that consists of the LastVoting algorithm and the communication-coordinator predicate:*

$$\forall \phi > 0, \exists \phi_0 \geq \phi, \exists c_0 \in \Pi, \forall p \in \Pi :$$

$$\begin{cases} |HO(c_0, 4\phi_0 - 3)| > n/2 \wedge |HO(c_0, 4\phi_0 - 1)| > n/2 \\ \wedge \\ (Coord(p, \phi_0) = c_0) \wedge (Coord(p, \phi_0) \in K(\phi_0)) \end{cases}$$

solves Consensus.

Proof The above communication-coordinator predicate enforces termination of the *LastVoting* algorithm. Indeed, if the predicate holds, the conditions of lines 17, 27, and 34 are all true at phase ϕ_0 .

Integrity follows from the fact that the decision is on $vote_p$, that $vote_p$ is always set to one of the x_p values received (line 13), and that x_p is only updated with $vote_p$ received (line 21).

For agreement, let ϕ_1 be the first phase at which some process makes a decision. Let p be such a process and let v be its decision value. Lines 27 and 31 imply that p 's coordinator at phase ϕ_1 , denoted c , has received more than $n/2$ acknowledgements at round $4\phi_1 - 1$ and $\text{vote}_c^{(4\phi_1-1)} = v$. Moreover, c has received more than $n/2$ non-null messages at round $4\phi_1 - 3$.

For any phase $\phi \geq \phi_1$, let Π_ϕ denote the set of processes that have updated their timestamp variables at least once since phase ϕ_1 :

$$\Pi_\phi = \{q \in \Pi : ts_q^{(\phi)} \geq \phi_1\}.$$

The heart of the proof is the following lemma, which says that from phase ϕ_1 , each process q may barter the value of x_q only for v .

Lemma 8 *At any phase $\phi \geq \phi_1$, the following holds:*

1. Π_ϕ is a majority set, i.e., $|\Pi_\phi| > n/2$.
2. For any process q in Π_ϕ , we have $x_q^{(4\phi-2)} = v$.

Proof By induction on $\phi - \phi_1$.

Basis: $\phi = \phi_1$. Any process q in Π_{ϕ_1} executes line (21), and so has received a vote from its coordinator $c' = \text{Coord}(q, \phi_1)$. Hence c' casts a vote at round $4\phi_1 - 3$, and so c' receives more than $n/2$ non-null messages at this round. Since each process sends at most one non-null message at round $4\phi_1 - 3$, we have $c' = c$. It follows that

$$x_q^{(4\phi_1-2)} = \text{vote}_c^{(4\phi_1-3)} = v.$$

Moreover, c receives more than $n/2$ acknowledgements, and so more than $n/2$ processes execute line 21 at phase ϕ_1 . This shows that Π_{ϕ_1} is a majority set.

Inductive step: Suppose that $\phi > \phi_1$, $|\Pi_{\phi-1}| > n/2$, and for any $q \in \Pi_{\phi-1}$, $x_q^{(4\phi-5)} = v$. At phase ϕ , any process q in $\Pi_{\phi-1}$ either lets ts_q unchanged or sets ts_q to ϕ . It follows that Π_ϕ contains $\Pi_{\phi-1}$, and so Π_ϕ is a majority set.

Let q be any process in Π_ϕ . We consider two cases.

1. Process q does not execute line 21 at phase ϕ . Then $ts_q^{(\phi)} = ts_q^{(\phi-1)}$ and $x_q^{(\phi)} = x_q^{(\phi-1)}$. It follows that q belongs also to $\Pi_{\phi-1}$. The inductive hypothesis implies that $x_q^{(\phi-1)} = v$.
2. Process q updates ts_q and x_q at phase ϕ . Let $c' = \text{Coord}(q, \phi)$. From lines 17, 18, and 20, it follows that c' has casted a vote and $\text{vote}_{c'}^{(4\phi-3)} = x_q^{(4\phi-2)}$. Therefore c' has received more than $n/2$ non-null messages at round $4\phi - 3$. Since each process sends at most one non-null message at this round, one of them has been sent by a process in the majority set $\Pi_{\phi-1}$. Line 12 implies that the largest timestamp received by c' at round $4\phi - 3$ is at least equal to ϕ_1 . From the inductive hypothesis we derive that $\text{vote}_{c'}^{(4\phi_1-3)} = v$. Hence $x_q^{(4\phi-2)} = v$, as needed. $\square_{\text{Lemma 8}}$

Let p' be a process that decides v' at phase ϕ , and let c' denote the coordinator of p' at phase ϕ . By definition of ϕ_1 , we have $\phi \geq \phi_1$. We are going to prove that $v = v'$. For that, we proceed by induction on $\phi - \phi_1$.

Basis: $\phi = \phi_1$. Process c' has necessarily received more than $n/2$ acknowledgements at phase $\phi = \phi_1$. Since each process sends at most one such message per phase, we have $c = c'$, and so

$$v' = \text{vote}_{c'}^{(4\phi_1-3)} = \text{vote}_c^{(4\phi_1-3)} = v.$$

Inductive step: Let $\phi > \phi_1$ and assume that any decision value at phases $\phi_1, \dots, \phi - 1$ is equal to v . Process c' has definitely casted a vote for v' at round $4\phi - 1$, and so more than $n/2$ processes q set x_q to v' and ts_q to ϕ at round $4\phi - 2$. Such processes share the same coordinator, namely c' . It follows that c' has received more than $n/2$ non-null messages at round $4\phi - 3$. By point (1) in Lemma 8, at least one of them has been sent by some process in Π_ϕ . From line 12 and point (2) in Lemma 8, it follows that $\text{vote}_{c'}^{(4\phi-3)} = v$. \square

Rotating coordinator: Similarly to *CoordUniformVoting*, we can use the rotating coordinator strategy to determine coordinators in *LastVoting*: we denote by *LastVoting*^{rc} the resulting algorithm. With such a coordinator strategy, the existence of a uniformly and well coordinated phase is ensured by the following communication predicate $\mathcal{P}_{\text{LastVoting}^{rc}}$:

$$\mathcal{P}_{\text{LastVoting}^{rc}} :: \exists \phi_0 > 0 : \bigcap_{i=1}^n K(\phi_0 + i) \neq \emptyset.$$

Observe that when (i) choosing the off-line strategy of the rotating coordinator in *LastVoting*, and (ii) requiring that the condition at line 11 always holds (which means that the coordinator sends a vote in every phase, see line 18), the resulting algorithm, denoted *CT*, corresponds to the *Rotating Coordinator* algorithm described in [5] for solving Consensus with the failure detector $\diamond S$.

Because of point (ii), it turns out that *CT* is safe only under some non-trivial invariant property of the heard-of sets, namely the no split predicate $\mathcal{P}_{\text{nosplit}}$. More precisely, agreement may be violated if two coordinators receive messages from disjoint sets of processes (i.e., there is no process heard by both). This point is not discussed in [5] because the authors assume no link failure and a majority of correct processes, which guarantees $\mathcal{P}_{\text{HO}}^{\text{maj}}$, and so $\mathcal{P}_{\text{nosplit}}$. If this assumption does not hold, then the *Rotating Coordinator* algorithm blocks forever, which is translated in the HO model by the fact *CT* is not safe.⁸ The failure detector $\diamond S$ or

⁸ At two places in each phase, processes in the *Rotating Coordinator* algorithm wait for at least $n/2$ messages to advance to the next round. In the HO model, advancing from one round to the next is automatic,

the predicate $\mathcal{P}_{\diamond nek}$ play a role only for the termination condition of Consensus. Obviously, the notion of failure detectors makes no sense in the context of link failures. However, our remark shows that basically, the *Rotating Coordinator* algorithm does not tolerate link failures, and more generally dynamic failures. To make it safe in the presence of such failures—which is a quite reasonable requirement—it is sufficient just to add the test “number of $\langle v, \theta \rangle$ received $> n/2$ ” at line 11.

5.6 A non-coordinated algorithm without HO invariant

The *DLS* and the *LastVoting* algorithms show that, if we resort to coordinators, Consensus can be solved without invariant properties for the heard-of sets, i.e., communication predicates of the form:

$$\mathcal{P}_{(\square \mathcal{C})} :: \forall r \geq 0 : \mathcal{C} \text{ holds at } r,$$

where \mathcal{C} is a predicate over the heard-of sets at a given round (e.g., $\mathcal{P}_{nekrounds}$ or $\mathcal{P}_{nosplit}$). This naturally leads us to question whether Consensus is solvable without both (i) invariant properties for the HO’s and (ii) coordinators. As we shall show below, the answer is yes if there exist rounds in which heard-of sets have a membership larger than $2n/3$ (Algorithm 9). We leave the question open in the case heard-of sets are only majority sets.

For that, we design an HO algorithm that we call *OneThirdRule* (Algorithm 9). A similar algorithmical schema is used in the first round of [3], in [35], and in the fast rounds of *Fast Paxos* [28]. Each phase of the *OneThirdRule* algorithm consists of one single round. Safety conditions of Consensus, namely integrity and agreement, are always satisfied: if some process decides v at line 10 of round r , then in any round $r' \geq r$, only v can be assigned to any x_p , and hence only v can be decided. Liveness is ensured by the following condition:

$$\exists r_0 > 0, \exists \Pi_0 \text{ s.t. } |\Pi_0| > 2n/3, \forall p \in \Pi : \\ (HO(p, r_0) = \Pi_0) \wedge (\exists r_p > r_0 : |HO(p, r_p)| > 2n/3).$$

The first part, namely the existence of some uniform round r_0 with an enough large kernel, makes the system “space uniform” in the sense that at the end of round r_0 , all processes adopt the same value for x_p . The second part of the condition forces every process p to make a decision at the end of round r_p . These observations establish the following result:

Theorem 8 *The HO machine consisting of the OneThirdRule algorithm and the communication predicate $\mathcal{P}_{(\mathcal{C}_0)\infty}$,*

and so is not under the control of processes. This is why executions of the *Rotating Coordinator* algorithm that block (because some process does not eventually hear of a majority of processes) are translated in the HO model into unsafe executions of the *CT* algorithm.

where \mathcal{C}_0 holds at round r_0 if

$$\exists \Pi_0 \text{ s.t. } |\Pi_0| > 2n/3, \forall p \in \Pi : HO(p, r_0) = \Pi_0, \\ \text{solves Consensus.}$$

Algorithm 9 The *OneThirdRule* algorithm

```

1: Initialization:
2:    $x_p := v_p$            {  $v_p$  is the initial value of  $p$  }
3: Round  $r$ :
4:    $S_p^r$  :
5:     send  $\langle x_p \rangle$  to all processes
6:    $T_p^r$  :
7:     if  $|HO(p, r)| > 2n/3$  then
8:        $x_p :=$  the smallest most often received value
9:     if more than  $2n/3$  values received are equal to  $\bar{x}$  then
10:      DECIDE( $\bar{x}$ )

```

Note that, contrary to all the algorithms we have described up to now, a decision is possible in just one round: if all the initial values are identical⁹ and few transmission faults occur at the beginning (i.e., \mathcal{C}_0 holds at round 1), then every process decides at the end of the first round. Hence, the *OneThirdRule* algorithm which is a very simple algorithm that does not require any coordinator election procedure, may be quite efficient. Furthermore, the condition for its correctness is satisfied in many real systems. Indeed, one uniform round and “not too many” transmission faults (i.e., heard-of sets with a cardinality greater than $2n/3$) in some subsequent round is sufficient to entail decision. This may seem to be a strong condition in the classical context of permanent and static failures, such as crash failures, but it is a realistic assumption in systems with transient failures (e.g., crash recovery). For all these reasons, we think that it would be very interesting to develop this solution in many real applications requiring Consensus among processes.

5.7 Summary

Table 2 summarizes the various HO and CHO Consensus algorithms described in this section, with their correctness requirements. Because of time invariance (cf. Sect. 2.1), the corresponding communication predicates guarantee conditions on rounds that must hold infinitely often, but not necessarily permanently, at least for the last four algorithms. The latter solutions are quite relevant in practice, since one can observe that real systems alternate between “bad” and “good” periods. Note that some of these requirements capture conditions that in the past have been expressed in terms of failure detectors, requiring conditions to eventually hold forever, and thus suitable for permanent failures only.

⁹ In the context of Atomic Broadcast, messages can sometimes be spontaneously ordered, which translates into identical initial values for Consensus.

Table 2 HO and CHO Consensus algorithms and their correctness conditions with communication predicates

Algorithm	Predicate for safety	Predicate for liveness
<i>UniformVoting</i>	no split rounds	$\forall r > 0, \exists r_0 \geq r : r_0$ is uniform
<i>CoordUniformVoting</i>	no split rounds	$\forall \phi > 0, \exists \phi_0 \geq \phi : \phi_0$ is uniformly and well coordinated
<i>CoordUniformVoting^{rc}</i>	no split rounds	$\forall \phi > 0, \exists \phi_0 \geq \phi : \bigcap_{i=1}^n K(\phi_0 + i) \neq \emptyset$
<i>CoordDLS</i>	none	$\forall \phi > 0, \exists \phi_0 \geq \phi :$ $\begin{cases} \phi_0 \text{ is uniform} \\ K(\phi_0) \text{ is a majority set} \\ \phi_0 \text{ is uniformly and well coordinated} \end{cases}$
<i>LastVoting</i>	none	$\forall \phi > 0, \exists \phi_0 \geq \phi :$ $\begin{cases} \forall p \in \Pi, \forall r, \text{ s.t. } 4\phi_0 - 3 \leq r \leq 4\phi_0 : HO(p, r) > n/2 \\ \phi_0 \text{ is uniformly and well coordinated} \end{cases}$
<i>LastVoting^{rc}</i>	none	$\forall \phi > 0, \exists \phi_0 \geq \phi :$ $\begin{cases} \forall p \in \Pi, \forall r, \text{ s.t. } 4\phi_0 < r < 4(\phi_0 + n) : HO(p, r) > n/2 \\ \bigcap_{i=1}^n K(\phi_0 + i) \neq \emptyset \end{cases}$
<i>OneThirdRule</i>	none	$\forall r > 0, \exists r_0 \geq r : r_0$ is uniform and $ K(r_0) > 2n/3$

6 Conclusion

The paper proposes a new computational model for fault-tolerant distributed systems, which is suitable for describing benign failures in a unified framework. Apart from allowing concise expressions of Consensus algorithms, the model overcomes the limitations of the traditional approaches by getting rid of two basic principles—*independence of synchrony degree and failure model*; *notion of faulty component*—on which previous models are all based. In particular, our approach allows us to handle (1) transient failures and (2) failures that hit all the components of the system (links and processes). By contrast, classical models are limited to static faults both in time and space.

As we have observed, a second key point of the HO model is to make *sporadic* conditions sufficient to solve Consensus, in contrast to the classical models obtained by “augmenting” asynchronous systems with external devices (like failure detectors [5], or other oracles). Indeed, in such augmented asynchronous systems, only *stable* properties—i.e., properties that once they hold, hold forever—can be useful for solving problems like Consensus. In the HO formalism, we can give a precise meaning to the statement “*the system works correctly for long enough*”, and we prove that such sporadic conditions are *sufficient* to make Consensus solvable whereas in augmented asynchronous models, Consensus *requires* stable properties of the type “*eventually and forever the system behaves correctly*”.

Besides, it is striking to see how, by removing the barrier between synchrony degree and failure model, the HO formalism enables us to give direct proofs of important results in fault-tolerant distributed computing. In this way, we can unify results for synchronous and asynchronous systems, and give a simple proof of the weakest predicate that makes Consensus solvable under some fault bounds.

In this paper, we dealt with benign failures only, but the HO model can be extended to handle more severe faults. Indeed, we pursue our approach in a sequel paper [2] where we show how to cope with *value faults*: messages may be corrupted, i.e., at any round r , the message received by process q from p may be different of the message that p ought to send to q . This novel framework covers the classical *Byzantine failures* [34] as well as the dynamic transmission faults studied in [37,38]. Thus, we derive new Consensus algorithms tolerating both benign and value faults, be they static or dynamic, permanent or transient. Besides, it would be worth extending the notion of translation introduced in Sect. 3 to the HO model with value failures, and studying the connection with the translations of Neiger and Toueg [33] from Byzantine to benign failures in synchronous systems.

Acknowledgments We would like to thank M. Biely, M. Hutle, U. Schmid and J. Widder for many helpful discussions related to the HO model. We also thank L. Lamport, J. Welch, O. Rütli, K. Peters, Ph. Kufner, and U. Nestmann for their comments on an earlier version of the paper, and the anonymous referees.

References

1. Ben-Or, M.: Another advantage of free choice: Completely asynchronous agreement protocols. In: Proceedings of the Second ACM Symposium on Principles of Distributed Computing, pp. 27–30 (1983)
2. Biely, M., Charron-Bost, B., Gaillard, A., Hutle, M., Schiper, A., Widder, J.: Tolerating corrupted communications. In: Proceedings of the Twentysixth ACM Symposium on Principles of Distributed Computing, pp. 244–253 (2007)
3. Brasileiro, F., Greve, F., Mostéfaoui, A., Raynal, M.: Consensus in one communication step. In: 6th International Conference Parallel Computing Technologies (PaCT), pp. 42–50. Springer, LNCS 2127 (2001)
4. Chandra, T.D., Hadzilacos, V., Toueg, S.: The weakest failure detector for solving consensus. J. ACM 43(4), 685–722 (1996)

5. Chandra, T.D., Toueg, S.: Unreliable failure detectors for asynchronous systems. *J. ACM* **43**(2), 225–267 (1996)
6. Chandy, K.M., Misra, J.: How processes learn. *Distrib. Comput.* **1**(1), 40–52 (1986)
7. Charron-Bost, B., Hutle, M., Widder, J.: In search of lost time. Technical Report LSR/2008-006, EPFL (2008)
8. Charron-Bost, B., Merz, S.: Formal verification of a Consensus algorithm in the Heard-Of model. TR 20009/07, LIX (2009)
9. Chlebus, B.S., Dicks, K., Pelc, A.: Broadcasting in synchronous networks with dynamic faults. *Networks* **27**, 309–318 (1996)
10. Chor, B., Dwork, C.: Randomization in byzantine agreement. *Adv. Comput. Res.* **5**, 443–497 (1989)
11. Dobrev, S.: Computing input multiplicity in anonymous synchronous networks with dynamic faults. *J. Discrete Algorithms* **2**(4), 425–438 (2004)
12. Dolev, D.: The Byzantine generals strike again. *J. Algorithms* **3**(1), 14–30 (1982)
13. Dolev, D., Dwork, C., Stockmeyer, L.: On the minimal synchronism needed for distributed consensus. *J. ACM* **34**(1), 77–97 (1987)
14. Dolev, D., Reischuk, R., Strong, H.R.: Early stopping in Byzantine agreement. *J. ACM* **37**(4), 720–741 (1990)
15. Dolev, D., Strong, H.R.: Polynomial algorithms for multiple processor agreement. In: *Proceedings of the Fourteenth ACM Symposium on Theory of Computing*, pp. 401–407. ACM Press, New York (1982)
16. Dolev, D., Strong, H.R.: Authenticated algorithms for Byzantine agreement. *SIAM J. Comput.* **12**(4), 656–666 (1983)
17. Dwork, C., Lynch, N.A., Stockmeyer, L.: Consensus in the presence of partial synchrony. *J. ACM* **35**(2), 288–323 (1988)
18. Elrad, T.E., Francez, N.: Decomposition of distributed programs into communication-closed-layers. *Sci. Comput. Progr.* **2**(3), April 1982
19. Fischer, M.J., Lynch, N.A.: A lower bound for the time to assure interactive consistency. *Inf. Process. Lett.* **14**, 183–186 (1982)
20. Fischer, M.J., Lynch, N.A., Paterson, M.S.: Impossibility of distributed consensus with one faulty process. *J. ACM* **32**(2), 374–382 (1985)
21. Fraigniaud, P., Peyrat, C.: Broadcasting in a hypercube when some calls fail. *Inf. Process. Lett.* **39**(3), 115–119 (1991)
22. Gafni, E.: Round-by-round fault detectors: unifying synchrony and asynchrony. In: *Proceedings of the Seventeenth ACM Symposium on Principles of Distributed Computing*, pp. 143–152 (1998)
23. Gopal, A., Toueg, S.: Reliable broadcast in synchronous and asynchronous environments (preliminary version). In: Bermond, J.-C., Raynal, M. (eds.) *Proceedings of the Third International Workshop on Distributed Algorithms*. Lecture Notes on Computer Science, vol. 392, pp. 110–123. Springer, Heidelberg (1989)
24. Herlihy, M., Rajsbaum, S., Tuttle, M.: Unifying synchronous and asynchronous message-passing models. In: *Proceedings of the Seventeenth ACM Symposium on Principles of Distributed Computing*, pp. 123–132 (1998)
25. Keidar, I., Shraer, A.: Timeliness, failure-detectors, and consensus performance. In: *ACM Symposium on Principles of Distributed Computing*, pp. 169–178 (2006)
26. Kralovic, R., Kralovic, R., Ruzicka, P.: Broadcasting with many faulty links. In: *SIROCCO*, pp. 211–222 (2003)
27. Lamport, L.: The part-time parliament. *ACM Trans. Comput. Syst.* **16**(2), 133–169 (1998)
28. Lamport, L.: Fast Paxos. *Distrib. Comput.* **19**(2), 79–103 (2006)
29. Lynch, N.A.: *Distributed Algorithms*. Morgan Kaufmann, Menlo Park (1996)
30. Malkhi, D., Oprea, F., Zhou, L.: Ω meets Paxos: leader election and stability without eventual timely links. In: *International Conference on Distributed Computing (DISC)*, pp. 199–213 (2005)
31. Merritt, M.J.: Unpublished Notes (1985)
32. Moses, Y., Rajsbaum, S.: A layered analysis of consensus. *SIAM J. Comput.* **31**(4), 989–1021 (2002)
33. Neiger, G., Toueg, S.: Automatically increasing the fault-tolerance of distributed algorithms. *J. Algorithms* **11**(3), 374–419 (1990)
34. Pease, M., Shostak, R., Lamport, L.: Reaching agreement in the presence of faults. *J. ACM* **27**(2), 228–234 (1980)
35. Pedone, F., Schiper, A., Urban, P., Cavin, D.: Solving agreement problems with weak ordering oracles. In: *Proceedings of the 4th European Dependable Computing Conference (EDCC-4)*, LNCS-2485, pp. 44–61, Toulouse, France. Springer, Heidelberg (2002)
36. Perry, K.J., Toueg, S.: Distributed agreement in the presence of processor and communication faults. *IEEE Trans. Softw. Eng.* **12**(3), 477–482 (1986)
37. Santoro, N., Widmayer, P.: Agreement in synchronous networks with ubiquitous faults. *Theor. Comput. Sci.* **384**, 232–249 (2007)
38. Santoro, N., Widmayer, P.: Time is not a healer. In: *Proceedings of the 6th Symposium on Theor. Aspects of Computer Science*, pp. 304–313, Paderborn, Germany (1989)
39. Santoro, N., Widmayer, P.: Distributed function evaluation in the presence of transmission faults. In: *SIGAL International Symposium on Algorithms*, pp. 358–367 (1990)
40. Santoro, N., Widmayer, P.: Majority and unanimity in synchronous networks with ubiquitous dynamic faults. In: *Proceedings of the 12th International Colloquium SIROCCO*. Lecture Notes on Computer Science, vol. 3499, pp. 262–276, Mont Saint Michel, France. Springer, Heidelberg (2005)
41. Schmid, U.: How to model link failures: A perception-based fault model. In: *IEEE International Conference on Dependable Systems and Networks (DSN)*, pp. 57–66 (2001)
42. Tsuchiya, T., Schiper, A.: Using bounded model checking to verify consensus algorithms. In: *International Conference on Distributed Computing (DISC)*, pp. 466–480 (2008)