

Computing in Distributed Systems in the Presence of Benign Failures

Bernadette Charron-Bost
Ecole polytechnique, France

André Schiper
EPFL, Switzerland

1 Two principles of fault-tolerant distributed computing

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 failures occur.

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

The new idea in our model is to study *round*-based computations and abstract away the implementation of the communication exchange between processes, be it shared memory, message-passing, or any other mechanism. In the whole paper, we call “*messages*” the pieces of information that are exchanged, whatever the medium of communication is. In each round, a process emits messages to the other processes, waits to receive messages from some processes, and then computes a new state. Every message received at some round must have been sent at that round. Consequently, any message missed at a round is definitely discarded. Using the terminology of Elrad and Francez [14], a round is a *communication-closed layer*.

Most of the fault-tolerant distributed algorithms that have been designed for synchronous systems, or for partially synchronous or asynchronous ones, are structured in rounds (e.g., [12, 1, 13, 9, 4]). However, concerning impossibility results (lower bounds, non-solvability, ...), 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, Lynch, and Stockmeyer [13] 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 [16] extended the round-based model to any type of systems. The basic idea in his model is to capture the properties of the communication mechanisms and of the system guarantees as a whole by a single module that is called *Round-by-Round Failure 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. 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 [16] 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 *Round-by-Round Failure Detector modules* only suspect processes, never links.

The RRFD model is here influenced by the whole literature on fault-tolerant distributed computing — with one single exception, namely the work by Santoro and Widmayer [24, 25] discussed in more details below — which models benign failures in terms of faulty components: for example, the loss of a message is attributed to a faulty behavior of either the sending process (send omission), the receiving process (receive omission), or the link. Unfortunately, the principle of *a priori* blaming some components for transmission failures yields several major problems. 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, so few they are. Second, in problem specifications, it naturally leads to exempt faulty processes to satisfy some conditions since the behaviors of faulty processes is out of control. In particular, in all decision problems, a faulty process is never obliged to make a decision. Indeed, even in their uniform versions [17] 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 to make a decision. For example, a process p that is blamed for the non-delivery of a single message — as this failure transmission is rightly or wrongly accounted for a receive omission from p — is allowed to make no decision even if p is blamed for nothing else. Finally, as already observed by Dolev [10], it appears that the real causes of transmission failures, namely sender failure, receiver failure, or link failure, may be actually unknown. Failure transmissions 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, and indeed we show in [7] 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 failures (effects) without accounting for the faulty components (causes).

Santoro and Widmayer [24, 25] clearly pointed out this issue. They introduced the *Transmission Faults model* that locates failures without specifying their cause. A transmission failure can represent link failure as well as process failure. Contrary to classical models in which transmission failures 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* failures. However, it is designed only for synchronous systems. Indeed, Santoro and Widmayer showed that dynamic transmission 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.

Contribution

Our aim is to develop a computational model for distributed systems that combines the advantages of the RRF model [16] and the Transmission Faults model [24], 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 failures without specifying by whom nor why such failures occur.*

More precisely, 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 at 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 failure from q to p at round r is characterized by the fact that q does not belong to $HO(p, r)$. 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 [4, 16]. 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 [23, 1] or the failure detector approach, there is no notion of “augmenting” asynchronous systems with external devices (oracles) that processes may query: the communication predicate corresponding to an HO system is an integral part of the model and should be rather seen as defining the *environment*. 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 cannot decouple predicates from the underlying algorithms. This is the reason why we encapsulate algorithm and communication predicate in the same structure that we shall call an *HO machine*.

In this paper, besides the construction of the HO model, we present one central result for fault-tolerance that illustrates its semantic effectiveness. This 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 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 two 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 [20, 11] to the general FLP asynchronous impossibility result [15] (this reduction has been previously established by Gafni for Atomic-Snapshot asynchronous systems [16]). 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.¹

This paper is structured as follows. In Section 2, we describe our model, and present many traditional systems in the HO framework. In Section 3, we define the notion of translation and propose a characterization of the communication predicates that make Consensus solvable (under certain transmission failure bounds). In Section 4, we give two basic translations, and highlight the key role played by the “no partitioning” assumption. Section 5 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 [13] and [16], and so the model generalizes the classical notion of synchronized rounds developed for synchronous systems [19]. We introduce the notion of *kernel* at 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.

2.1 Heard-Of sets and communication predicates

We suppose that we have a non-empty set Π of cardinality n , a set of messages M , and a *null* placeholder indicating the empty message. To each p in Π , 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

¹Interestingly, 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, [18] develops arguments from algebraic topology for synchronous, partially synchronous, and asynchronous systems as well, while [21] introduces the notion of *layering* as a tool for model-independent analysis of the Consensus problem. Note that both are faulty component based approaches.

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 $M \cup \{null\}$ 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 collections of subsets of Π (representing heard-of collections) that is not the constant predicate “False”, and that is invariant under time translation, i.e., \mathcal{P} has the same truth-value for any heard-of collection $(HO(p, r + i))_{p \in \Pi, r > 0}$, where $i \in \mathbb{N}$.² Note that if \mathcal{C} is a condition over the heard-of sets at some round, then the natural communication predicate that guarantees \mathcal{C} eventually holds at some round is the following:

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

which expresses that \mathcal{C} holds infinitely often.

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 at 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

$$coK(r) = \Pi \setminus K(r), \quad coK(\phi) = \Pi \setminus K(\phi), \quad \text{and} \quad coK = \Pi \setminus K.$$

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

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

²The latter condition is due to the fact that we do not want correctness of algorithms depends on the time at which algorithms start to run (see Proposition 2.1).

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 according to 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 would 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 $M = (A, \mathcal{P})$ 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 all 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; p 's state at the end of

A *problem* Σ for Π is a predicate over state collections. 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 2.1 *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 . Then the Consensus problem is defined as the conjunction of the following run predicates:

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:

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

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:

$$\forall p \in \Pi, \exists r_p > 0, \exists v \in V : \sigma_{p,r} \in \Sigma_p^v.$$

Note that as stated above, irrevocability is implied by agreement, and so we can only consider the last three conditions for the Consensus specification.

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 differs from the classical Termination condition of Consensus that only requires *non faulty* processes to make a decision (for example, cf. the specifications given in [19]). The point is that one may wonder whether an algorithm in which *all processes* decide can be implemented in real systems with *processors*³ that are prone to crash failures. The answer is yes. Of course, a processor that has crashed takes no steps, and so can make no decision (this is the reason why faulty components are traditionally exempt from deciding). However, the corresponding HO process is not heard of any more, and so has no impact on the rest of the computation. So there is no problem of transposing an HO machine solving the entirely uniform Consensus specification in a real system with possible crash failures: the capability of an HO process to make a decision is just not implemented if the corresponding processor has crashed.

With such a completely uniform specification, the artefact of requirements that depend on the way components are blamed for the failures disappears. Indeed, consider a real system with no failure except for a processor p that does not crash, but fails to send a message to each of its neighbors just once. There are three component failure models handling this scenario, namely the send omission, the receive omission, and the link failure models. Then with the classical specification of the Consensus problem, either (a) all processes except p , (b) only p , or (c) all processes must make a decision, accordingly. Hence, the requirement for Consensus in the traditional approach highly depends on the failure model, that is, on the way components are blamed for failures. This illustrates how the same syntactic specification may lead to various semantic requirements as soon as the specification is referring to the failure model. We avoid this problem in the HO formalism by considering completely uniform specifications with the same requirement for all processes.

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*⁴ 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 [8, 13, 16]. As for asynchronous systems with at most f initial crash failures, they clearly support the communication predicate $\mathcal{P}_{\diamond_{unif}}^f$ defined by:

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

³We use the term “processors” to designate the system entities that implement HO processes in a real system, in order to make the two notions clearly distinguishable from each other in the discussion below.

⁴In Table 1, we limit us to message-passing systems, but similar correspondences hold for shared-memory systems as well, and we examine them in more detail in a sequel paper.

Moreover, the positive result by Fischer, Lynch, and Paterson [15] for initial crash failures shows that in the case of a majority of correct processors ($2f < n$), space-time uniformity of the heard-of sets can be achieved from the beginning.⁵ 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 processors is correct.

2.4 HO counterpart of a system

At this point, we are naturally led to examine the notion of the *HO counterpart of a system* introduced by Gafni [16]. Roughly speaking, the HO counterpart of a system S is a communication predicate \mathcal{P} which exactly captures all the properties (synchrony degree, failure model, ...) of S . With respect to the partial order that we shall formally define in Section 3.1, \mathcal{P} corresponds to the strongest communication predicate that can be implemented from S . The notion of HO counterpart of a system raises two fundamental questions:

1. Does any system S have an HO counterpart \mathcal{P} ?
2. If it exists, does \mathcal{P} really capture all the properties of S ? And what is more, is it possible to give a rigorous meaning to the latter question?

We leave these questions open,⁶ and in the sequel we shall use the notion of HO counterpart informally. In Table 1, we give what we reasonably think to be the HO counterparts of various system types (i.e., the strongest implementability results).

As an example, $\mathcal{P}_{\diamond unif}^f$ is the HO counterpart of the partially synchronous systems with at most f crash failures described in [13]. Note that $\mathcal{P}_{\diamond unif}^f$ implies the HO counterpart of asynchronous systems with at most f initial crash failures when $f \geq n/2$. Interestingly, this explains why all these systems share the same “partitioning argument” to prove the impossibility of Consensus [15, 13, 4].

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 HO models, it consists in identifying the communication predicates of HO machines that solve Consensus.

We partially answer the question by limiting us 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 cannot be solved without an implicit and permanent consensus on heard-of sets*. We then compare our result with the one established by Chandra, Hadzilacos, and Toueg [3] for asynchronous systems augmented with failure detector oracles.

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*.

⁵The algorithm in [15] ensures agreement on the membership of the initial clique.

⁶The same question arises for models with failure detectors: for example, does the model with the *eventual perfect failure detector* $\diamond\mathcal{P}$ entirely capture the computational power of the partially synchronous systems in [13]? Except in [6], which provides a negative answer in the case of the perfect failure detector and synchronous systems, this question has not yet been addressed, and is left open.

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 rounds 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 $M = (A, \mathcal{P})$ 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 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 rounds r_1, \dots, r_l such that for any index i , $1 \leq i \leq l$, we have

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

yields a general definition of translation.

If there exists an algorithm A such that the HO machine $M = (A, \mathcal{P})$ 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 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 A 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 3.1 *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 3.2 *Let \mathcal{P} be a communication predicate such that $\mathcal{P} \succeq \mathcal{P}_{sp_unif}^*$. Then there exists an algorithm A such that the HO machine $M = (A, \mathcal{P})$ solves Consensus.*

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. \square

This result can be interestingly compared with the impossibility of Consensus with the communication predicate $\mathcal{P}_{\diamond_unif}^*$ (cf. Section 2.3). This shows 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 Consensus is solvable. In other words, *Consensus is solvable only if there is an implicit permanent agreement on the first heard-of sets*.

Proposition 3.3 *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 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 2.1). 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 [5], 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 , \dots , 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. \square

Propositions 3.2 and 3.3 provide a characterization of the HO machines with nek rounds that solves Consensus:

Theorem 3.4 *In the class of communication predicates which are at least as strong as $\mathcal{P}_{nekrounds}$, the following assertions are equivalent:*

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

Note that $\mathcal{P}_{sp_unif}^*$ is the HO counterpart of one of the system types described in [8], namely systems with asynchronous processes, synchronous and reliable links, atomic send-to-all primitive, and at most $n - 1$ crashes (cf. Table 1). Hence, Theorem 3.4 shows that this system type is the weakest one for solving Consensus among the 32 system types described in [8], since each of them can implement nek rounds.

In [3], Chandra, Hadzilacos and Toueg characterize the failure detectors that make Consensus solvable in a system with reliable links and a minority of crash failures. Such a system can emulate the predicate \mathcal{P}_{HO}^{maj} . In Section 4.1, we prove that \mathcal{P}_{HO}^{maj} can be translated into $\mathcal{P}_{nekrounds}$, for which Theorem 3.4 applies. Therefore our result, whose proof is quite simple and direct, has the same scope as the one in [3].

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 two basic translations that are both uniform. Such translations simply handle union and intersection of heard-of sets. Interestingly, they allow us to amplify our characterization of nek round communication predicates that make Consensus solvable.

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 [16] relating synchronous and asynchronous models.

The translation computes $NewHO_p^{(\rho)}$ from the collection of heard-of sets at 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 4.1 *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 $\tilde{K}^{(\rho)} = NewHO_p^{(\rho)}$.

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

$$E(G_\rho) \geq n(n - f_1). \quad (1)$$

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

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

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

$$nbOut(y) \leq f_2. \quad (3)$$

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 4.1 can be specialized as follows:

Corollary 4.2 *There is a two round translation of \mathcal{P}_{HO}^{maj} into $\mathcal{P}_{nekrounds}$, and so*

$$\mathcal{P}_{HO}^{maj} \succeq \mathcal{P}_{nekrounds}.$$

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 4.1 is obtained with $f_2 = 1$: in this case, Proposition 4.1 gives

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

Therefore, if $f_1 \leq n - 1$, 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 4.3 *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.*

A similar result is shown by Gafni [16] 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 [15] that Gafni derives from his result can also be span off from Corollary 4.3.

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 $\mathcal{P}_{nekrounds} \succeq \mathcal{P}_{nosplit}$. To prove that $\mathcal{P}_{nosplit} \succeq \mathcal{P}_{nekrounds}$, we present a $\lambda(n)$ round translation, where $\lambda(n)$ is the integer defined by

$$2^{\lambda(n)-1} < n \leq 2^{\lambda(n)},$$

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 .⁷ 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 . At 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

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

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:       $NewHO_p := Listen_p$ 

```

Theorem 4.4 *Algorithm 2 is a $\lambda(n)$ round translation of $\mathcal{P}_{nosplit}$ into $\mathcal{P}_{nekrounds}$, and so we have $\mathcal{P}_{nosplit} \simeq \mathcal{P}_{nekrounds}$.*

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

⁷Precisely, we have $r_1 = \lambda(n)(\rho - 1) + 1, \dots, r_{\lambda(n)} = \lambda(n)\rho$.

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 $G_{\lambda(n)}^*$:

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

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

Lemma 4.5 *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 $G_1^* = G_{n-1}$, and the lemma coincides with the no split predicate.

Inductive step: Suppose $i \geq 2$ 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 $x_1 = x_2$ or not; let x denote this node. By definition of G_i^* , x is a common in-neighbour to p_1, \dots, p_{i+1} in this graph. \square *Lemma 4.5*

By definition of the $NewHO_p$'s, the translation preserves kernels, which completes the proof that Algorithm 2 translates $\mathcal{P}_{nosplit}$ into $\mathcal{P}_{nekrunds}$. \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 $\mathcal{P}_{nekrunds}$. Thus we get another proof of Corollary 4.2, but the translation requires $\lambda(n)$ rounds instead of two rounds in Algorithm 1.

Combining Theorems 3.4 and 4.4, we get the following corollary:

Corollary 4.6 *In the class of communication predicates which are at least as strong as $\mathcal{P}_{nosplit}$, the following assertions are equivalent:*

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

5 Conclusion

The paper proposes a new computational model for fault-tolerant distributed systems, which is suitable for describing benign failures in a unified framework. 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 failures both in time and space.

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 failure bounds.

In this paper, we dealt with benign failures only, but the HO model can be extended to handle more severe failures. Indeed, we pursue our approach in a sequel paper [2] where we show how to cope with *value failures*: 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* [22] as well as the dynamic transmission faults studied in [24], and leads to new interesting solutions in the presence of corrupted informations.

Acknowledgments. We would like to thank M. Biely, M. Huttler, U. Schmid and J. Widder for discussions related to the HO model. We also thank L. Lamport and J. Welch for their comments on an earlier version of the paper.

References

- [1] M. Ben-Or. Another advantage of free choice: Completely asynchronous agreement protocols. In *Proceedings of the Second ACM Symposium on Principles of Distributed Computing*, pages 27–30, August 1983.
- [2] M. Biely, B. Charron-Bost, A. Gaillard, M. Huttler, A. Schiper, and J. Widder. Tolerating corrupted communications. In *Proceedings of the Twentysixth ACM Symposium on Principles of Distributed Computing*, August 2007. To appear.
- [3] T. D. Chandra, V. Hadzilacos, and S. Toueg. The weakest failure detector for solving consensus. *Journal of the ACM*, 43(4):685–722, July 1996.
- [4] T. D. Chandra and S. Toueg. Unreliable failure detectors for asynchronous systems. *Journal of the ACM*, 43(2):225–267, March 1996.
- [5] K. Mani Chandy and Jayadev Misra. How processes learn. *Distributed Computing*, 1(1):40–52, 1986.
- [6] B. Charron-Bost, R. Guerraoui, and A. Schiper. Synchronous systems and perfect failure detectors: Solvability and efficiency issues. In *Proceedings of the International Conference on Dependable Systems and Networks*, pages 523–532. IEEE, June 2000.
- [7] B. Charron-Bost and A. Schiper. The heard-Of model: Unifying all benign failures. Technical Report LSR/2006-004, Département Systèmes de Communication, EPFL, July 2006.
- [8] D. Dolev, C. Dwork, and L. Stockmeyer. On the minimal synchronism needed for distributed consensus. *Journal of the ACM*, 34(1):77–97, January 1987.
- [9] D. Dolev, R. Reischuk, and H. R. Strong. Early stopping in Byzantine agreement. *Journal of the ACM*, 37(4):720–741, October 1990.
- [10] Danny Dolev. The Byzantine generals strike again. *Journal of Algorithms*, 3(1):14–30, 1982.
- [11] Danny Dolev, Rüdiger Reischuk, and H. Raymond Strong. Early stopping in Byzantine agreement. Technical Report RJ5406, IBM Research Laboratory, December 1986.
- [12] Danny Dolev and H. Raymond Strong. Authenticated algorithms for Byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, November 1983.

- [13] C. Dwork, N. A. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, April 1988.
- [14] T.E. Elrad and N. Francez. Decomposition of distributed programs into communication-closed-layers. *Science of Computer Programming*, 2(3), April 1982.
- [15] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.
- [16] E. Gafni. Rounds-by-rounds fault detectors: unifying synchrony and asynchrony. In *Proceedings of the Seventeenth ACM Symposium on Principles of Distributed Computing*, pages 143–152, August 1998.
- [17] Ajei Gopal and Sam Toueg. Reliable broadcast in synchronous and asynchronous environments (preliminary version). In J.-C. Bermond and M. Raynal, editors, *Proceedings of the Third International Workshop on Distributed Algorithms*, volume 392 of *Lecture Notes on Computer Science*, pages 110–123. Springer Verlag, September 1989.
- [18] Maurice Herlihy, Sergio Rajsbaum, and Mark Tuttle. Unifying synchronous and asynchronous message-passing models. In *Proceedings of the Seventeenth ACM Symposium on Principles of Distributed Computing*, pages 123–132, August 1998.
- [19] N. A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
- [20] M. J. Merritt. Unpublished Notes, 1985.
- [21] Y. Moses and S. Rajsbaum. A layered analysis of consensus. *SIAM Journal of Computing*, 31(4):989–1021, 2002.
- [22] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *Journal of the ACM*, 27(2):228–234, April 1980.
- [23] Michael Rabin. Randomized Byzantine generals. In *Proceedings of the Twenty-Fourth Symposium on Foundations of Computer Science*, pages 403–409. IEEE Computer Society Press, November 1983.
- [24] N. Santoro and P. Widmayer. Time is not a healer. In *Proceedings of the 6th Symposium on Theor. Aspects of Computer Science*, pages 304–313, Paderborn, Germany, 1989.
- [25] N. Santoro and P. Widmayer. Majority and unanimity in synchronous networks with ubiquitous dynamic faults. In *Proceedings of the 12th International Colloquium SIROCCO*, volume 3499 of *Lecture Notes on Computer Science*, pages 262–276, Mont Saint Michel, France, May 2005. Springer Verlag.

SYSTEM TYPE	COMMUNICATION PREDICATE
Synchronous, reliable links at most f faulty senders	$ K \geq n - f$
Synchronous, at most f omission transmission faults ([24])	$\forall r > 0 : \sum_{p \in \Pi} HO(p, r) \geq n^2 - f$
Synchronous, a block of at most f omission transmission faults ([24])	$\forall r > 0 : K(r) \geq n - f$
Synchronous, 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, reliable links asynchronous processes, atomic send to all at most f crash failures ([8])	$\forall p \in \Pi, \forall r > 0 : HO(p, r) \geq n - f$ \wedge $\forall p, q \in \Pi^2, \forall r > 0 : HO(p, r) = HO(q, r)$
Synchronous, reliable links asynchronous processes, at most 1 crash failure ([8])	$\forall p \in \Pi, \forall r > 0 : 1 \leq HO(p, r) \leq 2$ \wedge $\forall p, q \in \Pi^2, \forall r > 0 : HO(p, r) = HO(q, r)$
Asynchronous, reliable links at most f crash failures	$\forall p \in \Pi, \forall r > 0 : HO(p, r) \geq n - f$
Asynchronous, 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 $f < n/2$	$\exists \Pi_0 \subseteq \Pi$ s.t. $ \Pi_0 \geq n - f, \forall p \in \Pi, \forall r > 0 :$ $HO(p, r) = \Pi_0$
Partially synchronous ([13]) Eventual reliable links at most f crash failures	$\exists \Pi_0 \subseteq \Pi$ s.t. $ \Pi_0 \geq n - f, \exists r_0 > 0, \forall p \in \Pi, \forall r > r_0 :$ $HO(p, r) = \Pi_0$

Table 1: System types and their HO counterpart