On the Strongest Message Adversary for Consensus in Directed Dynamic Networks

Ulrich Schmid
ECS, TU Wien
s@ecs.tuwien.ac.at

Manfred Schwarz
ECS, TU Wien
mschwarz@ecs.tuwien.ac.at

Abstract

Inspired by the successful chase for the weakest failure detector in asynchronous message passing systems with crash failures and surprising relations to synchronous directed dynamic networks with message adversaries established by Raynal and Stainer [PODC'13], we[1] introduce the concept of message adversary simulations and use it for defining a notion for strongest message adversary for solving distributed computing problems like consensus and k-set agreement. We prove that every message adversary that admits all graph sequences consisting of perpetual star graphs and is strong enough for solving multi-valued consensus is a strongest one. We elaborate on seemingly paradoxical consequences of our results, which also shed some light on the fundamental difference between crash-prone asynchronous systems with failure detectors and synchronous dynamic networks with message adversaries.

Eligible for best student paper award: yes.

1 Introduction

Synchronous distributed systems consisting of a possibly unknown number n of processes that never fail but where a *message adversary* (MA) [1] controls the ability to communicate is a well-established model for dynamic networks [24]. Runs are determined by sequences of communication graphs $\mathcal{G}^1, \mathcal{G}^2, \ldots$ here, where a directed edge (p, q) is in $\mathcal{G}^r$ iff the message adversary does not suppress the message sent by p to q in round r; we will use the notation $p \rightarrow_r q$ to concisely express this. A message adversary can be identified by the set of (infinite) graph sequences it may generate, which are called *admissible* graph sequences. Research has provided various possibility and impossibility results for agreement problems, in particular, (deterministic) consensus, in this setting, both in undirected [25] and directed dynamic networks [7–9, 31, 32]. Albeit these results enclose the impossibility/possibility border of consensus quite tightly, no "strongest" message adversary for consensus is known yet.

Raynal and Stainer [28] established an interesting relation between synchronous systems (abbreviated $\mathcal{SMP}$) with message adversaries and asynchronous message-passing systems (abbreviated $\mathcal{AMP}$) with process crashes augmented by failure detectors [12]. Among other results, they showed that $\mathcal{AMP}$ in conjunction with the weakest failure detector (Ω, Σ) for consensus with an arbitrary number of crashes [17] can be simulated in $\mathcal{SMP}$ with the message adversary (SOURCE, QUORUM) (and *vice versa*). This message adversary guarantees communication graphs

[1]This work has been supported by the Austrian Science Fund FWF under the projects ADynNet (P28182) and RiSE/SHiNE (S11405).

that are all rooted[2] and where, for every pair of processes p, q and every pair of rounds r, r', there is some process ℓ such that $\ell \to_r p$ and $\ell \to_{r'} q$. Additionally, there is a process p and some round r_0 such that for all processes q and all rounds $r > r_0$ we have $p \to_r q$. Consequently, every (Ω, Σ)-based consensus algorithms for $\mathcal{AMP}$ can be employed atop of this simulation in $\mathcal{SMP}$ with (SOURCE, QUORUM).

However, the question arises whether failure detector-based consensus algorithms on top of failure detector implementations can indeed compete with specifically designed consensus algorithms for $\mathcal{SMP}$ for some given MA. In particular, is it always possible to implement the weakest failure detector (Ω, Σ) atop of $\mathcal{SMP}$ with a message adversary that admits a consensus algorithm? Conversely, given that (Ω, Σ) is a weakest failure detector for consensus, is it somehow possible to simulate $\mathcal{SMP}$ with a strong(est) message adversary for consensus atop of $\mathcal{AMP}$ augmented with (Ω, Σ)? Interestingly, it follows from our results in [9] that neither is possible: Σ cannot be implemented in $\mathcal{SMP}$ with some message adversary $\mathrm{VSSC}_{D,E}(\infty)$ that admits a consensus algorithm, and the latter cannot be implemented in $\mathcal{AMP}$ equipped with (Ω, Σ) either. Essentially, the reason is that all properties achievable in $\mathcal{AMP}$ with failure detectors are inherently time-free, i.e., of eventual-type, whereas $\mathcal{SMP}$ with message adversaries facilitates time-dependent properties: The latter are sometimes too short-lived to guarantee eventual properties, however, and, conversely, cannot be extracted from eventual properties either.

In this paper, we avoid the detour via failure detectors and introduce the simple concept of *message adversary simulations* in $\mathcal{SMP}$, using the HO model [15] as a basis. Inspired by the definition of a weakest failure detector, we also define a notion for a *strongest message adversary*: A strongest MA S for a given problem P is such that (i) it admits a solution algorithm for the problem P in $\mathcal{SMP}$ equipped with S, and (ii) every MA A that admits a solution algorithm for P in $\mathcal{SMP}$ with A allows to simulate $\mathcal{SMP}$ equipped with S.

Using MA simulations, we prove that the message adversary STAR (which generates an infinite sequence of identical star graphs $\mathcal{G}, \mathcal{G}, \dots$) is a strongest message adversary for multi-valued consensus. Moreover, we show that every message adversary A that satisfies $\mathrm{STAR} \subseteq A$ and allows to solve multi-valued consensus is also a strongest message adversary. It hence turns out that both (SOURCE, QUORUM) and $\mathrm{VSSC}_{D,E}(\infty)$ are strongest message adversaries, even though neither $(\mathrm{SOURCE}, \mathrm{QUORUM}) \subseteq \mathrm{VSSC}_{D,E}(\infty)$ nor $(\mathrm{SOURCE}, \mathrm{QUORUM}) \supseteq \mathrm{VSSC}_{D,E}(\infty)$ holds. Moreover, there are interesting and sometimes apparently paradoxical consequences resulting from our findings, like the one that $\mathcal{AMP}$ with (Ω, Σ) allows to simulate $\mathcal{SMP}$ with the strongest message adversary (SOURCE, QUORUM), which in turn can be transformed into $\mathcal{SMP}$ with the strongest message adversary $\mathrm{VSSC}_{D,E}(\infty)$, which in turn does not allow to simulate $\mathcal{AMP}$ with (Ω, Σ)!

The remainder of our paper is organized as follows: After a short account of related work in Section 2, we define our synchronous message passing models $\mathcal{SMP}$ with message adversaries in Section 3. Asynchronous message passing models $\mathcal{AMP}$ with failure detectors are introduced in Section 4, along with a collection of failure detector-related definitions and results. In Section 5, we introduce a simple simulation relation between message adversaries and prove that the message adversary STAR is strongest for solving consensus in $\mathcal{SMP}$. We also show that this result generalizes to a fairly large class of strongest message adversaries. In Section 6, we discuss the consequences arising from the fact that STAR is a strongest message adversary. Some conclusions in Section 7 complete our paper.

[2] A graph is rooted if it has a rooted directed spanning tree.

2 Related work

There is a huge amount of work on relations between different distributed computing models. The implementability of failure detectors in timing-based models of computation has already been addressed in Chandra and Touegs seminal work [12], and received quite some attention in the chase for the weakest system model for implementing Ω [2–5, 19, 21, 26, 27].

There are also several papers that establish more abstract relations between various synchronous models and asynchronous models with failure detectors. Charron-Bost, Guerraoui and Schiper showed that the partially synchronous model [18] with $\Phi = 1$ and $\Delta \geq 1$ is not equivalent to the asynchronous model with perfect failure detectors in terms of problem solvability. A more general relation between various eventually synchronous models and asynchronous models with stabilizing failure detectors, which also considers efficiency of the algorithmic transformations, has been established by Biely et. al. in [6]: Among other results, it establishes that Ω is essentially equivalent to models with an eventually timely source, as well as to eventual lock-step rounds. [14, 22] shed some light on limitations of the failure detector abstraction in timing-based models of computation, which are relevant in our context.

Research on consensus in synchronous message passing systems subject to communication failures dates back at least to the seminal paper [29] by Santoro and Widmayer; generalizations have been provided in [10, 13, 15, 16, 30]. The term message adversary was coined by Afek and Gafni in [1]. Whereas the message adversaries in all the work above are oblivious, in the sense that they may choose the graph for a round arbitrarily from a fixed set of graphs, [7] and some follow-up work [8, 31] allows arbitrary sequences of communication graphs (that can also model stabilizing behavior, for example). In [28], Raynal and Stainer related stabilizing message adversaries to asynchronous systems with failure detectors, which actually stimulated our interest in the problem addressed in this paper. As our first step, we showed in [9] that Σ cannot be implemented in $\mathcal{SMP}$ with message adversary $\mathrm{VSSC}_{D,E}(\infty)$ that admits a consensus algorithm, and the latter cannot be implemented in $\mathcal{AMP}$ equipped with (Ω, Σ) either. In this paper, we will show that these results also hold for a simpler message adversary.

Researchers have also developed several "round-by-round" frameworks, which allow to relate models of computation with different degrees of synchrony and failures. Examples are round-by-round fault detectors by Gafni [20], the GIRAF framework by Keidar and Shraer [23], and the the HO model by Charron-Bost and Schiper [15].

3 The Model $\mathcal{SMP}$

The model for $\mathcal{SMP}$ used in this paper will be based on the HO model introduced in [15], which provides all the features needed for defining our MA simulations. It consists of a non-empty set $\Pi = \{p_1, \ldots, p_n\}$ of n processes with unique ids, and a set of messages M, which includes a null placeholder indicating the empty message. Each process $p \in \Pi$ consists of the following components: a set of states denoted by $states_p$, a subset $init_p$ of initial states, for each positive integer $r \in \mathbb{N}*$, called round number, a message sending function S_p^r mapping states $p \times \Pi$ to a unique (possibly null) message m_p, and a state-transition function T_p^r mapping $states_p$ and partial vectors (indexed by Π) z_p of elements of M to $states_p$. The collection of the pairs of message sending function and state-transition function of the processes for every round $r > 0$ is called an algorithm on Π.

Computations in the HO model are composed of infinitely many rounds, which are communication-closed layers in the sense that any message sent in a round can be received only at that round.

In each round r, process p first applies S_p^r to the current state $\mathtt{s}_p^{r-1} \in states_p$, 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 $HO(p,r)$ to compute $\mathtt{s}_p^r$.

A communication predicate 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 $\Pi \times \mathbb{N}*$:

$$P : (2^{\Pi})^{\Pi \times \mathbb{N}*} \Rightarrow \{true, false\}$$

Rather than directly using communication predicates for describing our message adversaries, however, we will exploit the fact that $\cup_{p\in\Pi} HO(p,r) = G^r$. Consequently, we will usually stick to the admissible graph sequences of a given MA, and silently assume that they are translated to the according communication predicate.

In order to describe information propagation in a sequence $\mathcal{G}_r, \mathcal{G}_{r+1} \ldots, \mathcal{G}_{r+\ell}$ of communication graphs, the notion of edges in the compound graph $\mathcal{G}_r \circ \mathcal{G}_{r+1} \circ \cdots \circ G_{r+\ell}$ becomes useful: Given two graphs $\mathcal{G} = \langle V, E\rangle$, $\mathcal{G}' = \langle V, E'\rangle$ with the same vertex-set V, the *compound graph* $\mathcal{G} \circ \mathcal{G}' := \langle V, E''\rangle$ where $p \to q \in E''$ if and only if for some $p' \in V : p \to p' \in E$ and $p' \to q \in E'$. Since we assume self-loops in every communication graph, the compound graph can be expressed by the product of the adjacency matrices of $\mathcal{G}$ and $\mathcal{G}'$. By $\mathtt{s}_p^r \rightsquigarrow \mathtt{s}_q^{r'}$, we express the fact that (the state $\mathtt{s}_p^r$ of) p at the end of round r *influences* (the state $\mathtt{s}_q^{r'}$ of) q at the end of round r', which obviously requires a chain of messages from p to q. Consequently, $\mathtt{s}_p^r \rightsquigarrow \mathtt{s}_q^{r'}$ if and only if $p \to q$ in $\mathcal{G}_r \circ \cdots \circ G_{r'}$.

In the appendix, we will also use some of the convenient notation introduced in [9] for reasoning about high-level properties of the graphs and graph sequences, like source components and vertex-stable source components.

Following [28], the abbreviation used for such models in the sequel is $\mathcal{SMP}_n[adv : MA]$, where n is the number of processes and MA is the name assigned to a set of admissible graph sequences. For example, $\mathcal{SMP}_n[adv : \text{SOURCE}, \text{QUORUM}]$ denotes the synchronous model with message adversary (SOURCE, QUORUM) defined below.

We will restrict our attention to the (uniform[3]) consensus problem in this paper, which is defined as follows: Every process $p \in \Pi$ has an input value $x_p \in V$ from some domain V ($V = \{0,1\}$ for binary consensus) and a decision value y_p, initially undefined $y_p = \perp$. Uniform consensus requires every process p that does not crash before it decides to irrevocably assign a value from V to y_p according to the following properties:

(V) Validity: y_p has to be equal to one of the x_q's.
(A) Agreement: $y_p = y_q$ for every pair of processes $p, q \in \Pi$ that decide.
(T) Termination: y_p has to be assigned a value in finite time at every process p that does not crash in the run.

A generalization/relaxation of consensus is k-set agreement [11], which allows at most k different decision values system-wide; 1-set agreement is equivalent to consensus.

We say that a problem like consensus is *impossible* under some model $\mathcal{SMP}_n[adv : MA]$, if there is no deterministic algorithm that solves the problem for every admissible communication graph sequence of MA. For example, every problem that requires at least some communication among the processes is impossible under the *unrestricted* message adversary ∞, as the sequence $\mathcal{G}, \mathcal{G}, \ldots$ where $\mathcal{G}$ does not contain even a single edge is also admissible here.

[3]Note that we will also study consensus in asynchronous systems with crash failures later on. In $\mathcal{SMP}$, no process ever crashes.

We will now specify four message adversaries, which are primarily used in this paper. The first two, Definition 3.1 and 3.2, are simplified versions of MAs introduced in [9], where we strengthened the properties as much as possible, albeit in a way that neither sacrificed their sufficiency for solving consensus resp. k-set agreement nor their insufficiency for implementing certain failure detectors.

Definition 3.1 (Message adversary VSSC(∞))**.** *The message adversary* VSSC(∞) *is the set of all sequences of communication graphs* $(\mathcal{G}^r)_{r>0}$*, where the following holds:*

(i) For every round $r > 0$*,* $\exists p \in \Pi, \forall q \in \Pi$*: there exists a (directed) path from* p *to* q *in every* $\mathcal{G}^r$*.*

(ii) There exists a round $r > 0$ *such* $\forall r' > r : S_r = S_{r'}$*, where the set* S_r *is such that* $p \in S_r$ *if* $\forall q \in \Pi$*: there exists a path from* p *to* q *in* $\mathcal{G}^r$*.*

The property that the set S consists of the same vertices for some duration or even, as in (ii) above, forever is called *vertex stability* [9].

Definition 3.2 (k-set message adversary VSSC$_k$(∞))**.** *The message adversary* VSSC$_k$(∞) *is the set of all sequences of communication graphs* $(\mathcal{G}^r)_{r>0}$*, where*

(i) there is some $k > 0$ *and* $P_1 \cup \cdots \cup P_k = \Pi$*, such that, in every* $\mathcal{G}_r$*,* $r > 0$*, every* P_i *is an isolated, weakly connected component,*

(ii) VSSC(∞) *holds independently in every partition* P_i*.*

Since it turns out that VSSC(∞) $\subset$ VSSC$_{D,E}(d)$ for $D = E = n - 1$, where VSSC$_{D,E}(d)$ is a message adversary introduced in [9] that allows to solve consensus, it follows that consensus can also be solved in $\mathcal{SMP}_n[adv : \text{VSSC}(\infty)]$ (by using the consensus algorithm for VSSC$_{D,E}(d)$), and that k-set agreement can be solved in $\mathcal{SMP}_n[adv : \text{VSSC}_k(\infty)]$ (by the same algorithm). Details can be found in the appendix.

The next two message adversaries have been introduced in [28]: SOURCE requires that, eventually, there is a round after which some process successfully sends a message to every process in the system:

Definition 3.3 (Message adversary SOURCE)**.** *The message adversary* SOURCE *is the set of all sequences of communication graphs* $(\mathcal{G}^r)_{r>0}$*, where* $\exists p \in \Pi : \exists r_0 \geq 1 : \forall r \geq r_0 : \forall q \in \Pi : p \rightarrow_r q$*.*

QUORUM requires that every two processes $p, q \in \Pi$ eventually hear from some common process $\ell \in \Pi$, in every pair of rounds r_p, r_q. Moreover, it requires that the set of processes that appear strongly correct (formally introduced in Definition 4.5) is non-empty:

Definition 3.4 (Message adversary QUORUM)**.** *The message adversary* QUORUM *is the set of all sequences of communication graphs* $(\mathcal{G}^r)_{r>0}$*, where* $\forall p, q \in \Pi : \forall r_p, r_q : (\{k : \ell \rightarrow_{r_p} p\} \cap \{\ell : \ell \rightarrow_{r_q} q\} \neq 0)]$*, and the set of strongly correct processes is not emtpy.*

$\mathcal{SMP}_n[adv : \text{SOURCE}, \text{QUORUM}]$ and $\mathcal{SMP}_n[adv : \text{VSSC}(\infty)]$ allow a solution algorithm for consensus according to [28] and [9], respectively.

4 Failure Detectors in Asynchronous Systems

In asynchronous systems ($\mathcal{AMP}$), processes may take steps at any time, and the time between two steps must be finite and larger than 0. Given some initial configuration $C^0 = (s_1^0, \ldots, s_n^0)$ consisting of the initial states of all processes, a run (also called execution) in $\mathcal{AMP}$ is a sequence of infinitely

many steps of every process starting from C^0, where every message sent must eventually be received and processed in finite time (except when failures occur, see below). Note that the end-to-end delay of a single message, from the time it is broadcast to the time it is received and processed, can be different for different recipients. Conceptually, we assume a (non-observable) clock with domain $\mathcal{T} = \{0, 1, 2, \dots\}$ and require all computing steps in a run to occur synchronized with this clock.

A convenient way to characterize consensus and k-set solvability in distributed systems where processes are (usually) subject to crash failures are (weakest) *failure detectors* [12].

Definition 4.1 (Process crashes)**.** *We say that process p_i crashes at time $t \geq 0$, if it stops executing its computing step at time t (possibly leaving it incomplete) and does not execute further steps at time $t' > t$.*

A failure detector [12] is an oracle that can be queried by any process in any computing step. Formally, a failure detector $\mathcal{D}$ with range $\mathcal{R}$ maps each failure pattern F to a non-empty set of histories with range $\mathcal{R}$, where a history H with range $\mathcal{R}$ is a function $H : \Pi \times \mathcal{T} \to \mathcal{R}$. The failure pattern is a function $F : \mathcal{T} \to 2^{\Pi}$ that maps each $t \in \mathcal{T}$ to the processes that have crashed by t. The set of all possible failure patterns is called the environment. Finally, $\mathcal{D}(F)$ denotes the set of possible failure detector histories permitted by $\mathcal{D}$ for the failure pattern F.

Definition 4.2. *$\mathcal{AMP}_{n,x}[\mathrm{fd} : FD]$ denotes the asynchronous message passing model consisting of n processes, at most x of which may crash in a run, augmented by failure detectors FD.*

Two important failure detectors for consensus in $\mathcal{AMP}$ are Σ and Ω, as their combination (Ω, Σ) is known to be a weakest failure detector in the wait-free environment [17].

Definition 4.3. *The eventual leader failure detector Ω has range Π. For each failure pattern F, for every history $H \in \Omega(F)$, there is a time $t \in \mathcal{T}$ and a correct process s.t. for every process p, $H(p, t) = q$.*

Definition 4.4. *The quorum failure detector Σ has range 2^{Π}. For each failure pattern F, for every $H \in \Sigma(F)$, two properties hold: (1) for every $t, t' \in \mathcal{T}$ and $p, q \in \Pi$ we have $H(p, t) \cap H(q, t') \neq \emptyset$ and (2) there is a time $t \in \mathcal{T}$ s.t. for every process p, $H(p, t) \subseteq \Pi \setminus \bigcup_{t \in \mathcal{T}} F(t)$.*

In order to relate such failure detector models to our message adversaries, we use the simple observation that the externally visible effect of a process crash can be expressed in our setting: Since correct processes in asynchronous message passing systems perform an infinite number of steps, we can assume that they send an infinite number of (possibly empty) messages that are eventually received by all correct processes. As in [28], we hence assume that the correct (= non-crashing) processes in the simulated $\mathcal{AMP}$ are the *strongly correct processes*. Informally, a strongly correct process is able to disseminate its state to all other processes infinitely often.

Definition 4.5 (Faulty and strongly correct processes)**.** *Given an infinite sequence of communication graphs σ, process p is* faulty *in a run with σ if there is a round r s.t., for some process q, for all $r' > r$: $\mathtt{s}_p^r \not\rightsquigarrow \mathtt{s}_q^{r'}$.*
Let $\mathcal{C}(\sigma) = \left\{ p \in \Pi \mid \forall q \in \Pi, \forall r \in \mathbb{N}, \exists r' > r \colon \mathtt{s}_p^r \rightsquigarrow \mathtt{s}_q^{r'} \right\}$ denote the strongly correct (= non-faulty) processes in any run with σ.

If a given process influences just one strongly correct process infinitely often, it would transitively influence all processes in the system, hence would also be strongly correct. Therefore, in order not to be strongly correct, a faulty process must not influence *any* strongly correct process infinitely often. We can hence define failure patterns as follows:

Definition 4.6 (Failure Pattern)**.** *The failure pattern associated with communication graph sequence σ is a function $F_\sigma : \mathbb{N} \to 2^\Pi$ s.t. $p \in F_\sigma(r)$ if, and only if, for all processes $q \in \mathcal{C}(\sigma)$, for all $r' > r$: $\mathtt{s}_p^r \not\rightsquigarrow \mathtt{s}_q^{r'}$.*

Note that $F_\sigma(r) \subseteq F_\sigma(r+1)$ as required.

The following lemmas have originally been proven for the message adversary $\mathrm{VSSC}_{D,E}(d)$ in [9].[4] In the appendix, we will adopt these proofs for the weaker message adversaries given in Definition 3.1 and 3.2.

Lemma 4.7. *$\mathcal{SMP}_n[adv : \mathrm{VSSC}(\infty)]$ does not allow to implement $\mathcal{AMP}_{n,n-1}[fd : \Sigma]$.*

Proof. [9, Proof of Lemma 28] in the appendix □

Lemma 4.8. *$\mathcal{AMP}_{n,n-1}[fd : \Sigma_k]$ cannot be implemented in $\mathcal{SMP}_n[adv : \mathrm{VSSC}_k(\infty)]$.*

Proof. [9, Proof of Lemma 30] in the appendix □

Lemma 4.9. *For $k > 1$, $\mathcal{AMP}_{n,n-1}[fd : \Omega_k]$ cannot be implemented in $\mathcal{SMP}_n[adv : \mathrm{VSSC}_k(\infty)]$.*

Proof. [9, Proof of Lemma 31] in the appendix □

Lemma 4.8 may come as a surprise, since the proof of the necessity of Σ_k for k-set agreement (hence the necessity of $\Sigma = \Sigma_1$ for consensus) developed by Raynal et. al. [11] only relies on the availability of a correct k-set agreement algorithm. However, their reduction proof works only in $\mathcal{AMP}_{n,n-1}$, i.e., crash-prone asynchronous message passing systems: It relies crucially on the fact that there cannot be a safety violation (i.e., a decision on a value that eventually leads to a violation of k-agreement) in any finite prefix of a run. This is not the case in the simulation running atop of $\mathcal{SMP}_{n,0}[adv : \mathrm{VSSC}(\infty)]$, however, as we cannot ensure the crash failure semantics of faulty processes (that is needed for ensuring safety in arbitrary prefixes) here. Hence, we cannot apply their result (or adapt their proof) in our setting.

From these negative results, we conclude that, given $\mathcal{SMP}$ with some message adversary, looking out for simulations of $\mathcal{AMP}_{n,n-1}[fd : (\Sigma, \Omega)]$ in order to be able to run standard failure detector-based consensus algorithms is not a viable alternative to the development of a tailored consensus algorithm, and is hence also no substitute for the chase for strongest message adversaries in $\mathcal{SMP}$. We hence need a different way for approaching the latter, which will be presented in the following section.

5 Message Adversary Simulations and the Strongest Message Adversary for Consensus

The lemmas in the previous section showed that $\mathcal{AMP}_{n,n-1}[fd : \Sigma, \Omega]$ cannot be simulated atop of $\mathcal{SMP}_n[adv : MA]$ with some message adversary MA that allows to solve consensus. Even though failure detectors cannot hence be used directly to find a strongest message adversary, the concept of comparing models with different restrictions in terms of their computational power is nevertheless attractive. This idea was already used in [15] to structure communication predicates in the HO model, albeit the "general translations" introduced for this purpose suffered from the fact that one would need to solve repeated consensus.

[4] A note to the reviewers: These results have been added in a major revision of the forthcoming journal paper [9], in an attempt to answer some challenge of a reviewer. They have hence not been published before.

Our equivalent of a failure detector simulation is a *message adversary simulation* of MA M atop of M', using a suitable simulation algorithm A running in $\mathcal{SMP}_n[adv : M']$ that emulates $\mathcal{SMP}_n[adv : M]$. Note that A may also depend on the algorithm $\mathcal{A}$ that is to be run in $\mathcal{SMP}_n[adv : M]$ here. If such a simulation exists, for every $\mathcal{A}$, then M' and M have the same computational power, i.e., M' allows a solution for every problem where M allows a solution. We will now describe the details of our MA simulation, using the HO model as a basis.

Consider the HO model corresponding to $\mathcal{SMP}_n[adv : M']$, and let A be a still to-be-defined algorithm that maintains a variable $NewHO_p \subseteq \Pi$ at every process p. For some positive integer k, let the macro-round $\rho \geq 1$ for process p be the sequence of the k consecutive rounds $r_1 = k(\rho - 1) + 1, \ldots, r_k = k\rho$. Note that $k = k(p, \rho)$ may be different for different (receiver) processes p and macro rounds ρ here. We say that A *emulates* (macro-)rounds $\rho \in \{1, 2, \ldots\}$ of $\mathcal{SMP}_n[adv : M]$, if, in any run of the latter, the value of $NewHO_p^{(\rho)}$ computed at the end of macro-round ρ satisfies:

(E1) $q \in NewHO_p^{(\rho)}$ iff $\mathtt{s}_q^{r_1-1} \rightsquigarrow \mathtt{s}_p^{r_k}$, i.e., if there exist an integer l in $\{1, ..., k\}$, a chain of $l + 1$ processes $p_0, p_1, ..., p_l$ from $p_0 = q$ to $p_l = p$, and a subsequence of l increasing round numbers $r_1, ..., r_l$ 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 $NewHO_p^{(\rho)}$ for all $p \in \Pi, \rho > 0$ satisfies M.

Clearly, the purpose of (E1) and (E2) is to guarantee well-defined and correct emulations, respectively.

Implementing the above emulation, i.e., the emulation algorithm A, is trivial: Let $m_{p\to q}^r$ represent the message sent by p to q in round r in $\mathcal{SMP}_n[adv : M']$, and $m_{p\to q}^{(\rho)}$ the message sent in macro-round ρ in the simulated $\mathcal{SMP}_n[adv : M]$. A just piggy-backs $m_{p\to q}^{(\rho)}$ on message $m_{p\to q}^j$, for every $(\rho - 1)k + 1 \leq j \leq \rho k$, and delivers $m_{p\to q}^{(\rho)}$ in $z_q^{(\rho)}$ in macro-round ρ, along with putting q into $NewHO_p^{(\rho)}$, when some $m_{p\to q}^j$ has been successfully received. Unfortunately, however, this emulation is too restrictive for our purpose.

Our next step will hence be to define a more abstract *simulation* of $\mathcal{SMP}_n[adv : M]$, by relaxing (E1) in a way that still guarantees well-defined simulations. We recall that, by definition, $q \in HO(p, r)$ iff $m_{q\to p}^r \in z_p^r$, and that $m_{q\to p}^r = S_q^r(\mathtt{s}_q^{r-1}, p)$. The former is of course equivalent to $q \in HO(p, r)$ iff $m' \in z_p^r$ and $m' = m_{q\to p}^r$. Now consider the following relaxed variant of (E1), where we replace the requirement of p having *received* the message $m_{q\to p}^r$ by the requirement of q having attempted to *send* $m_{q\to p}^r$:

(E1') $q \in NewHO_p^{(\rho)}$, iff there exists at least one j, $(\rho-1)k+1 \leq j \leq \rho k$, for which p has acquired local knowledge of m' with $m' = S_q^j(\mathtt{s}_q^{j-1}, p)$.

We say that A *simulates* (macro-)rounds $\rho > 0$ of $\mathcal{SMP}_n[adv : M]$, if, in any run of the latter, the value of $NewHO_p^{(\rho)}$ computed at the end of macro-round ρ satisfies (E1') and (E2). At the first glance, (E1') appears to be equal to (E1), as it has the same outcome in the case where a chain of messages from q to p as specified in (E1) exists. However, the essential difference is played out in the case where such a chain does not exist: Sometimes, it may be possible for the simulation algorithm A at process p to locally simulate the execution of $\mathcal{A}$ at process q, and hence to locally compute m' without actual communication!

Using this type of message adversary simulations, in conjunction with the fact that every communication predicate can be viewed as a message adversary, we will prove in Lemma 5.2 below that consensus solvability and the ability to simulate the communication predicate SP_UNIF introduced in [15] are equivalent.

Definition 5.1. *Let SP_UNIF be the communication predicate where for all $p, q, r : HO(p, r) = HO(q, r)$.*

Lemma 5.2. *The following assertions are equivalent:*

(1) For any set of initial values V, there is an algorithm A that solves consensus in $\mathcal{SMP}_n[adv : M']$.

(2) M' allows to simulate $\mathcal{SMP}_n[adv : SP\_UNIF]$ in the execution of every algorithm $\mathcal{A}$.

Proof. The direction (2) $\rightarrow$ (1) follows from the fact that [15] provided a (trivial) algorithm $\mathcal{A}$ that solves multi-valued consensus. We can hence plug-in $\mathcal{A}$ in (2) to obtain a consensus algorithm in $\mathcal{SMP}_n[adv : M']$.

To show the direction (1) $\rightarrow$ (2), let A be an algorithm that solves multi-valued consensus in $\mathcal{SMP}_n[adv : M']$, and consider an arbitrary algorithm $\mathcal{A}$ to be executed in $\mathcal{SMP}_n[adv : SP\_UNIF]$. We design an algorithm B based on A and $\mathcal{A}$, which allows to simulate $\mathcal{SMP}_n[adv : SP\_UNIF]$ in the execution of $\mathcal{A}$.

To simulate the first macro-round $\rho = 1$, B first executes A on every process until consensus is solved. More specifically, p starts A with the local input value $x_p = state_p^{(0)}$, where $state_p^{(0)}$ denotes algorithm $\mathcal{A}$'s initial state. Let v be the common decision value, and $v.id = \ell$ for $v = state_\ell^{(0)}$. When A terminates at process p, B sets $NewHO_p^{(1)} := \{v.id\}$. By validity, v is indeed the initial state $state_\ell^{(0)}$ of some process $\ell \in \Pi$, and by agreement, $NewHO_p^{(1)} = NewHO_q^{(1)}$ for every $p, q \in \Pi$.

Now, assuming inductively that every process p knows $state_\ell^{(\rho-1)}$ and $state_p^{(\rho-1)}$(as well as $\mathcal{A}$), B at p can also locally compute the message $m_{\ell\rightarrow\ell}^{(\rho)} = S_\ell^{(\rho)}(state_\ell^{(\rho-1)}, \ell)$ and $m_{\ell\rightarrow p}^{(\rho)} = S_\ell^{(\rho)}(state_\ell^{(\rho-1)}, p)$ sent by ℓ in macro round ρ. Moreover, B sets the message vector $z_\ell^{(\rho)}$ of the messages "received" by the simulated algorithm $\mathcal{A}$ for process ℓ to $z_\ell^{(\rho)} = \{m_{\ell\rightarrow\ell}^{(\rho)}\}$ and $z_p^{(\rho)} = \{m_{\ell\rightarrow p}^{(\rho)}\}$, from where it can locally compute $state_\ell^{(\rho)} = T_\ell^{(\rho)}(state_\ell^{(\rho-1)}, z_\ell^{(\rho)})$ and $state_p^{(\rho)} = T_p^{(\rho)}(state_p^{(\rho-1)}, z_p^{(\rho)})$. Finally, p sets $NewHO_p^{(\rho)} = \{\ell\}$ accordingly.

By construction, (E1') clearly holds. Moreover, since agreement secures $NewHO_p^{(1)} = NewHO_q^{(1)}$, which in turn leads to $NewHO_p^{(\rho)} = NewHO_q^{(\rho)}$ for every $\rho \geq 1$ due to the identical local computations at p and q, B indeed simulates $\mathcal{SMP}_n[adv : SP\_UNIF]$, which confirms also (E2). □

With these preparations, we will now define and discuss our notion of a *strongest message adversary*:

Definition 5.3 (Strongest message adversary)**.** *A message adversary M is a strongest message adversary for some problem $\mathcal{P}$, if for every M' for which $\mathcal{P}$ is solvable in $\mathcal{SMP}_n[adv : M']$, there exists an algorithm A that allows to simulate $\mathcal{SMP}_n[adv : M]$ in the execution of every algorithm $\mathcal{A}$ that solves $\mathcal{P}$.*

A property that follows directly from Definition 5.3 is:

Corollary 5.4. *If a strongest message adversary for multi-valued consensus allows to solve some problem $\mathcal{P}$, it holds that every message adversary that allows to solve multi-valued consensus also allows to solve $\mathcal{P}$.*

By Lemma 5.2, SP_UNIF is a strongest message adversary for multi-valued consensus. Even more, as the simulation algorithm A used in the proof of Lemma 5.2 actually simulates the message adversary $STAR \subset SP\_UNIF$, where $HO(p, r) = HO(q, s)$ for every $r, s \geq 0$ and every $p, q \in \Pi$, it reveals that $STAR$ is also a strongest message adversary for multi-valued consensus.

Since every other message adversary that contains $STAR$ is also a strongest message adversary by definition, we finally obtain the following Corollary 5.5:

Corollary 5.5 (Class of strongest message adversaries for consensus)**.** *Let $STAR$ be the message adversary that consist of all sequences of all possible perpetual stars. Every message adversary that includes $STAR$ is a strongest message adversary for multi-valued consensus.*

Examples for such message adversaries are (SOURCE, QUORUM), VSSC(∞) introduced in Section 3, and SP_UNIF.

Interestingly, the findings above can be easily be adopted for k-set agreement as well: The same simulation algorithm B as used in the proof of Lemma 5.2 can be used to simulate k perpetual stars atop of a message adversary M' that allows to solve k-set agreement: As any k-set agreement algorithm A guarantees at most k different decision values, B indeed allows to simulate at most k perpetual stars with the k decisions as the centers. Hence:

Corollary 5.6 (Class of strongest message adversaries for k-set agreement)**.** *Every message adversary that contains all sequences of all possible perpetual k-stars is a strongest message adversary for k-set agreement.*

Examples for strongest message adversaries for k-set agreement are $\text{VSSC}_k(\infty)$ and the message adversary $VSSC_{D,H}(n,\infty) + MAJINF(k)$ introduced in [9].

6 Consequences of our Results

The results of Section 4, in particular, Lemma 4.7, reveal the following facts:

(i) Since VSSC(∞) does not allow to implement Σ, we cannot hope to run Σ, Ω-based consensus algorithms on top if it.

(ii) The message adversary (SOURCE, QUORUM) considered in [28], VSSC(∞) and SP_UINF are all incomparable in terms of graph sequence inclusion, even though they all belong to the class of strongest message adversaries.

(ii) There are message adversaries like the one introduced in [31], which (unlike VSSC(∞)) do not even guarantee a single strongly correct process in some runs. Implementing Σ subject to Definition 4.4 atop of such message adversaries is trivially impossible, as its specification becomes void.

On the other hand, the results of Section 5, in particular, Lemma 5.2, reveals that it is possible to simulate the message adversary SP_UNIF atop of any message adversary (hence also VSSC(∞)) that allows to solve multi-valued consensus. However, it is trivial to simulate Σ, Ω in $\mathcal{AMP}$ in $\mathcal{SMP}_n[adv : SP\_UNIF]$: Initially, process p outputs p as the leader and Π as the quorum. At the end of round 1, both the leader and the quorum is set to $NewHO(p,r)$. Therefore, we seem to have arrived at a contradiction of Lemma 4.7!

This seemingly paradoxical result is traceable to the fact that the set $NewHO(p,r)$ provided by the simulation of SP_UNIF need *not* contain a strongly correct process! Indeed, recall that the infinite repetition of $\mathcal{G}$ can also be achieved by letting every process p in the system *locally* simulate the behavior of some ℓ's algorithm. This is possible, since p knows both ℓ's deterministic algorithm and its initial state, from the star graph G in round 1.

Hence, it finally turns out that the impossibility of implementing Σ established in Lemma 4.7 depends crucially on the assumption to consider strongly correct processes as correct in the simulated $\mathcal{AMP}$. In principle, it might be possible to implement Σ (and also Ω) atop of any message adversary that allows to solve consensus if a weaker alternative of Definition 4.1 of correct processes

in $\mathcal{AMP}$ was used: For ℓ, it would essentially be sufficient if it managed to disseminate its initial state to all processes in the system once. Quite obviously, though, such a definition of a correct process would severely affect the semantics of failure detectors and hence the wealth of known results.

In addition, the principal ability to simulate Σ, Ω atop of the simulated system $\mathcal{SMP}_n[adv : SP\_UNIF]$ is not very useful in practice, as it hinges on the availability of a consensus algorithm for the bottom-level message adversary M'. Consequently, this possibility does not open up a viable alternative to the development of consensus algorithms tailored to specific message adversaries like the ones introduced in [31, 32].

Overall, it turns out that strongest message adversaries according to Definition 5.3 do not have much discriminating power, as essentially all message adversaries known to us that allow to solve consensus are strongest according to Corollary 5.5. Finding a better definition of a strongest message adversary is a topic of future research. Note, however, that naive ideas like one that (i) admits a solution algorithm and (ii) is maximal w.r.t. its set of admissible graph sequences it may generate do not easily work out: Given that the latter set is usually uncountable, as admissible graph sequences are infinite, it is not clear whether (ii) is well-defined in general.

7 Conclusions

We defined message adversary simulations as a means for defining a notion of a strongest message adversary for consensus in synchronous directed dynamic networks. It turned out that every message adversary that allows to solve consensus and admits all sequences consisting of perpetual star graphs is a strongest one. We elaborate on some seemingly paradoxical consequences of our results and their relation to asynchronous systems with failure detectors.

References

[1] Y. Afek and E. Gafni. Asynchrony from synchrony. In D. Frey, M. Raynal, S. Sarkar, R. Shyamasundar, and P. Sinha, editors, *Distributed Computing and Networking*, volume 7730 of *Lecture Notes in Computer Science*, pages 225–239. Springer Berlin Heidelberg, 2013.

[2] M. K. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. Stable leader election. In *DISC '01: Proceedings of the 15th Inter. Conference on Dis. Computing*, pages 108–122. Springer-Verlag, 2001.

[3] M. K. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. On implementing Omega with weak reliability and synchrony assumptions. In *Proceeding of the 22nd Annual ACM Symposium on Principles of Distributed Computing (PODC'03)*, pages 306–314, New York, NY, USA, 2003. ACM Press.

[4] M. K. Aguilera, C. Delporte-Gallet, H. Fauconnier, and S. Toueg. Communication-efficient leader election and consensus with limited link synchrony. In *PODC'04*, pages 328–337, St. John's, Newfoundland, Canada, 2004. ACM Press.

[5] E. Anceaume, A. Fernández, A. Mostéfaoui, G. Neiger, and M. Raynal. A necessary and sufficient condition for transforming limited accuracy failure detectors. *J. Comp. Sys. Sci.*, 68(1):123–133, 2004.

[6] M. Biely, M. Hutle, L. Draque Penso, and J. Widder. Relating stabilizing timing assumptions to stabilizing failure detectors regarding solvability and efficiency. In *9th International Symposium on Stabilization, Safety, and Security of Distributed Systems*, volume 4838 of *Lecture Notes in Computer Science*, pages 4–20, Paris, November 2007. Springer Verlag.

[7] M. Biely, P. Robinson, and U. Schmid. Agreement in directed dynamic networks. In *SIROCCO'12*, LNCS 7355, pages 73–84. Springer-Verlag, 2012.

[8] M. Biely, P. Robinson, U. Schmid, M. Schwarz, and K. Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. In *NETYS'15*, Springer LNCS 9466, pages 109–124, Agadir, Morocco, 2015. Springer International Publishing.

[9] M. Biely, P. Robinson, U. Schmid, M. Schwarz, and K. Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. *Theoretical Computer Science*, pages –, 2018.

[10] M. Biely, U. Schmid, and B. Weiss. Synchronous consensus under hybrid process and link failures. *Theoretical Computer Science*, 412(40):5602 – 5630, 2011. http://dx.doi.org/10.1016/j.tcs.2010.09.032.

[11] F. Bonnet and M. Raynal. On the road to the weakest failure detector for k-set agreement in message-passing systems. *Theoretical Computer Science*, 412(33):4273 – 4284, 2011.

[12] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *Journal of the ACM*, 43(2):225–267, March 1996.

[13] B. Charron-Bost, M. Függer, and T. Nowak. Approximate consensus in highly dynamic networks: The role of averaging algorithms. In M. M. Halldòrsson, K. Iwama, N. Kobayashi, and B. Speckmann, editors, *Automata, Languages, and Programming*, volume 9135 of *Lecture Notes in Computer Science*, pages 528–539. Springer Berlin Heidelberg, 2015.

[14] B. Charron-Bost, M. Hutle, and J. Widder. In search of lost time. *Information Processing Letters*, 110(21):928–933, October 2010.

[15] B. Charron-Bost and A. Schiper. The Heard-Of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, April 2009.

[16] É. Coulouma, E. Godard, and J. G. Peters. A characterization of oblivious message adversaries for which consensus is solvable. *Theor. Comput. Sci.*, 584:80–90, 2015.

[17] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, V. Hadzilacos, P. Kouznetsov, and S. Toueg. The weakest failure detectors to solve certain fundamental problems in distributed computing. In *PODC'04*, pages 338–346. ACM Press, 2004.

[18] C. Dwork, N. Lynch, and L. Stockmeyer. Consensus in the presence of partial synchrony. *Journal of the ACM*, 35(2):288–323, April 1988.

[19] A. Fernández and M. Raynal. From an asynchronous intermittent rotating star to an eventual leader. *IEEE Trans. Parallel Distrib. Syst.*, 21(9):1290–1303, 2010.

[20] E. Gafni. Round-by-round fault detectors (extended abstract): unifying synchrony and asynchrony. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, pages 143–152, Puerto Vallarta, Mexico, 1998. ACM Press.

[21] M. Hutle, D. Malkhi, U. Schmid, and L. Zhou. Chasing the weakest system model for implementing omega and consensus. *IEEE Transactions on Dependable and Secure Computing*, 6(4):269–281, 2009.

[22] P. Jayanti and S. Toueg. Every problem has a weakest failure detector. In *PODC '08*, pages 75–84, New York, NY, USA, 2008. ACM.

[23] I. Keidar and A. Shraer. Timeliness, failure detectors, and consensus performance. In *PODC'06*, pages 169–178, New York, NY, USA, 2006. ACM Press.

[24] F. Kuhn and R. Oshman. Dynamic networks: Models and algorithms. *SIGACT News*, 42(1):82–96, 2011.

[25] F. Kuhn, R. Oshman, and Y. Moses. Coordinated consensus in dynamic networks. PODC '11. ACM, 2011.

[26] D. Malkhi, F. Oprea, and L. Zhou. Ω meets paxos: Leader election and stability without eventual timely links. In *DISC'05*, volume 3724 of *LNCS*, pages 199–213, Cracow, Poland, 2005. Springer Verlag.

[27] A. Mostéfaoui and M. Raynal. Solving consensus using Chandra-Toueg's unreliable failure detectors: A general quorum-based approach. In P. Jayanti, editor, *DISC'99*, volume 1693 of *Lecture Notes in Computer Science*, pages 49–63, Bratislava, Slovak Republic, September 1999. Springer-Verlag GmbH.

[28] M. Raynal and J. Stainer. Synchrony weakened by message adversaries vs asynchrony restricted by failure detectors. In *PODC'13*, pages 166–175, 2013.

[29] N. Santoro and P. Widmayer. Time is not a healer. In *STACS'89*, LNCS 349, pages 304–313, Paderborn, Germany, February 1989. Springer-Verlag.

[30] U. Schmid, B. Weiss, and I. Keidar. Impossibility results and lower bounds for consensus under link failures. *SIAM Journal on Computing*, 38(5):1912–1951, 2009.

[31] M. Schwarz, K. Winkler, and U. Schmid. Fast consensus under eventually stabilizing message adversaries. ICDCN '16, pages 7:1–7:10, New York, NY, USA, 2016. ACM.

[32] K. Winkler, M. Schwarz, and U. Schmid. Consensus in directed dynamic networks with short-lived stability. *CoRR*, abs/1602.05852, 2016.

A Sufficiency of $\mathrm{VSSC}(\infty)$ and $\mathrm{VSSC}_k(\infty)$

In order to prove that $\mathrm{VSSC}(\infty)$ resp. $\mathrm{VSSC}_k(\infty)$ allow to solve consensus resp. k-set agreement, we need to introduce some basic notation from [9].

Definition A.1 (Source Component)**.** *A* source component $S \neq \emptyset$ *of a graph* $\mathcal{G}$ *is the set of vertices of a* strongly connected component *in* $\mathcal{G}$ *that has no incoming edges from other components, formally* $\forall p \in S, \forall q \in \mathcal{G} \colon q \;\rightarrow\; p \in \mathcal{G} \Rightarrow q \in S$.

Note that every weakly connected directed simple graph $\mathcal{G}$ has at least one source component. If a graph $\mathcal{G}$ contains only one source component S, it is called a root component.

Vertex-stable source/root components are source components that remain the same for multiple rounds in a given graph sequence, albeit their actual interconnection topology may vary.

Definition A.2 (Vertex-Stable Source Component)**.** *Given a graph sequence* $(\mathcal{G}^r)_{r>0}$*, we say that the consecutive sub-sequence of communication graphs* $\mathcal{G}^r$ *for* $r \in I = [a, b]$*,* $b \geq a$*, contains an* I-vertex-stable source component S*, if, for* $r \in I$*, every* $\mathcal{G}^r$ *contains* S *as a source component.*

We abbreviate I-vertex-stable source component as I-VSSC, and write $|I|$-VSSC if only the length of I matters. Note carefully that we assume $|I| = b - a + 1$ here, since $I = [a, b]$ ranges from the *beginning* of round a to the *end* of round b.

One can show that a certain amount of information propagation is guaranteed in any strongly connected component C that is vertex-stable, i.e., whose vertex set remains the same, for a given number of rounds:

Lemma A.3. *Let* $C \subseteq \Pi$ *with* $|C| > 1$*, let* $a \in \mathbb{N}$ *and let* C *form a SCC of* $\mathcal{G}^r$ *for all* $r \in [a + 1, a + |C| - 1]$*. Then,* $\forall p, q \in C$*, it holds that* $\mathtt{s}_p^a \rightsquigarrow \mathtt{s}_q^{a+|C|-1}$.

Corollary A.4 follows immediately from Lemma A.3 and the fact that, by definition, VSSCs are strongly connected components.

Corollary A.4. *For every* I*-vertex-stable source component* S *with* $|S| > 1$ *and* $I = [a, b]$*, it holds that* $\forall p, q \in S$*,* $\forall x, y \in I \colon y \geq x + |S| - 2 \Rightarrow \mathtt{s}_p^{x-1} \rightsquigarrow \mathtt{s}_q^y$.

In order to also model message adversaries that guarantee faster information propagation, Definition A.5 introduces a system parameter $D \leq n - 1$, called the *dynamic source diameter*.

Definition A.5 (D-bounded I-VSSC)**.** *A* I*-VSSC* S *is* D*-bounded with dynamic source diameter* D*, if* $\forall p, q \in S$*,* $\forall r, r' \in I \colon r' \geq r + D - 1 \Rightarrow \mathtt{s}_p^{r-1} \rightsquigarrow \mathtt{s}_q^{r'}$.

Analogous considerations apply for the *dynamic network depth* $E \leq n - 1$ in communication graphs $\mathcal{G}^r$ with a single source component.

Definition A.6 (E-influencing I-VSSC)**.** *A* I*-VSSC* S *is* E*-influencing with dynamic network depth* E*, if* $\forall p \in S$*,* $\forall q \in \Pi$*,* $\forall r, r' \in I \colon r' \geq r + E - 1 \Rightarrow \mathtt{s}_p^{r-1} \rightsquigarrow \mathtt{s}_q^{r'}$.

We can now specify the message adversary $\mathrm{VSSC}_{D,E}(d)$ introduced in [7, 9], which allows to solve consensus for a sufficiently large d (in particular, for $d = \infty$):

Definition A.7 (Consensus message adversary $\mathrm{VSSC}_{D,E}(d)$)**.** *For* $d > 0$*, the message adversary* $VSSC_{D,E}(d)$ *is the set of all sequences of communication graphs* $(\mathcal{G}^r)_{r>0}$*, where*

(i) for every round r*,* $\mathcal{G}^r$ *contains exactly one source component,*

(ii) all vertex-stable source components occurring in any $(\mathcal{G}^r)_{r>0}$ are D-bounded and E-influencing

(iii) for each $(\mathcal{G}^r)_{r>0}$, there exists some $r_{ST} > 0$ and an interval of rounds $J = [r_{ST}, r_{ST} + d - 1]$ with a D-bounded and E-influencing J-vertex-stable source component.

We conclude this section by showing that $\text{VSSC}(\infty) \subset \text{VSSC}_{D,E}(d)$ for $D = E = n - 1$. This follows immediately from setting, in addition to $D = E = n - 1$, $d = \infty$ and the following facts:

(1) (i) demands in both adversaries that every graph is rooted.

(2) (ii) is trivially fulfilled by any rooted graph sequence if $D = E = n - 1$, as shown by corollary A.4.

(3) if $d = \infty$ (iii) demands in both cases that the single source component eventually consists of the same vertices forever.

Thus, $\text{VSSC}(\infty) = \text{VSSC}_{n-1,n-1}(\infty)$ holds. Solvability for $\text{VSSC}_k(\infty)$ follows from the fact that the solution algorithm for $\text{VSSC}(\infty)$ can be run in every partition: as consensus is solved in every partition, k-set agreement is guarantee for the whole system.

B Detailed impossibility proofs

Lemma B.1. *$\mathcal{SMP}_n[adv : \text{VSSC}(\infty)]$ does not allow to implement $\mathcal{AMP}_{n,n-1}[fd : \Sigma]$.*

Proof. We will prove our lemma for $n = 2$ for simplicity, as it is straightforward to generalize the proof for arbitrary n. Suppose that, for all rounds r and any processes p, some algorithm $\mathcal{A}$ computes $out(p, r)$ s.t. for any admissible failure pattern F, $out \in \Sigma(F)$. Consider the graph sequence $\sigma = (p \to q)_{r \geq 1}$. Clearly, the failure pattern associated with σ is $F_\sigma(r) = \{q\}$. Hence, in the run ε starting from some initial configuration C^0 with sequence σ, there is some round r' s.t. $out(p, r) = \{p\}$ for any $r > r'$ by Definition 4.4. Let $\sigma' = (p \to q)_{r=1}^{r'}(p \leftarrow q)_{r>r'}$. By similar arguments as above, in the run ε' that starts from C^0 with sequence σ', there is a round r'' such that $out(q, r) = \{q\}$ for any $r > r''$. Finally, for $\sigma'' = (p \to q)_{r=1}^{r'}(p \leftarrow q)_{r=r'+1}^{r''}(p \leftrightarrow q)_{r>r''}$, let ε'' denote the run starting from C^0 with graph sequence σ''. Until round r', $\varepsilon'' \sim_p \varepsilon$, hence, as shown above, $out(p, r') = \{p\}$ in ε''. Similarly, until round r'', $\varepsilon'' \sim_q \varepsilon'$ and hence $out(q, r'') = \{q\}$ in ε''. Clearly, $\sigma, \sigma', \sigma'' \in \text{VSSC}(\infty)$ and $F_{\sigma''}(r) = \{\}$, that is, no process is faulty in σ''. However, in ε'', $out(p, r') \cap out(q, r'') = \emptyset$, a contradiction to Definition 4.4. □

We continue with the definitions of generalized failure detectors for the k-set agreement setting in crash-prone asynchronous message passing systems.

Definition B.2. *The range of the failure detector Ω_k is all k-subsets of 2^Π. For each failure pattern F, for every history $H \in \Omega_k(F)$, there $\exists LD = \{q_1, \ldots, q_k\} \in 2^\Pi$ and $t \in \mathcal{T}$ such that $LD \cap \mathcal{C} \neq \emptyset$ and for all $t' \geq t, p \in \mathcal{C} : H(p, t') = LD$.*

Definition B.3. *The failure detector Σ_k has range 2^Π. For each failure pattern F, for every $H \in \Sigma_k(F)$, two properties must hold: (1) for every $t, t' \in \mathcal{T}$ and $S \in \Pi$ with $|S| = k + 1$, $\exists p, q \in S : H(p, t) \cap H(q, t') \neq \emptyset$, (2) there is a time $t \in \mathcal{T}$ s.t. for every process p, for every $t' \geq t$: $H(p, t') \subseteq \mathcal{C}$.*

k-set agreement in our lock-step round model with link failures allows non-temporary partitioning, which in turn makes it impossible to use the definition of crashed and correct processes from

the previous section: In a partitioned system, every process p has at least one process q such that $\forall r' > r : \mathtt{s}_p^r \rightsquigarrow \mathtt{s}_q^{r'}$, but no p usually reaches all $q \in \Pi$ here. Definition 4.1 hence implies that there is no correct process in this setting. Hence, we employ the following generalized definition:

Definition B.4. *Given a infinite graph sequence σ, let a* minimal source set *S in σ be a set of processes with the property that $\forall q \in \Pi, \forall r > 0$ there exists $p \in S, r' > r$ such that $\mathtt{s}_p^r \rightsquigarrow \mathtt{s}_q^{r'}$. The set of* weakly correct *processes $\mathcal{WC}(\sigma)$ of a sequence σ is the union of all minimal source sets S in σ.*

This definition is a quite natural extension of correct processes in a model, which allows perpetual partitioning of the system. Based on this definition of weakly correct processes, it is possible to generalize some of our consensus-related results (obtained for Σ and Ω). First, we show that Σ_k cannot be implemented, since $\mathrm{VSSC}_k(\infty)$ allows the system to partition into k isolated components.

Lemma B.5. *$\mathcal{AMP}_{n,n-1}[fd : \Sigma_k]$ cannot be implemented under $\mathcal{SMP}_n[adv : \mathrm{VSSC}_k(\infty)]$.*

Proof. For $k = 1$, we can rely on Lemma Theorem B.1, as every $\sigma \in \mathrm{VSSC}(\infty)$ is also admissible in $\mathrm{VSSC}_k(\infty)$. Hence, $\Sigma_1 = \Sigma$ cannot be implemented in $\mathrm{VSSC}_k(\infty)$.

The impossibility can be expanded to $k > 1$ by choosing some σ that (i) perpetually partitions the system into k components $\widetilde{P} = \{P_1, \ldots, P_k\}$ that each have a single source component and consist of the same processes throughout the run, and (ii) demands eventually a vertex stable source component in every partition forever. Pick an arbitrary partition $P \in \widetilde{P}$. If $|P| > 1$, such a sequence does not allow to implement Σ in P (e.g., the message adversary could emulate the graph sequence used in Lemma Theorem B.1 in P). We hence know that $\exists p, p' \in P$ and $\exists r, r'$ such that $out(p, r) \cap out(p', r') = \emptyset$. Furthermore, and irrespective of $|P|$, as for every $p \in P$, it is indistinguishable whether any $q \in \widetilde{P} \setminus P$ is faulty in σ or not, p has to assume that every process $q \in \widetilde{P} \setminus P$ is faulty. Hence, for every $p \in P$, we must eventually have $out(p, r_i) \subseteq P$ for some sufficiently large r_i.

We now construct a set S of $k+1$ processes that violates Definition Theorem B.3: fix some $P \in \widetilde{P}$ with $|P| > 1$ and add the two processes $p, p' \in P$, as described above, to S. For every partition $P_j \in \widetilde{P} \setminus P$, add one process p_i from P_j to S. Since there exist r, r' such that $out(p, r) \cap out(p', r') = \emptyset$, and $\forall P_j \in \widetilde{P} \setminus P, \forall p \in P_j, \exists r_i \colon out(p_i, r_i) \subseteq P_i$ and, by the construction of S, we have that $\forall p, q \in S$, $\exists r_i, r_j$ such that $out(p, r_i) \cap out(q, r_j) = \emptyset$. This set S clearly violates Definition Theorem B.3, as required. □

Lemma B.6. *For $k > 1$, $\mathcal{AMP}_{n,n-1}[fd : \Omega_k]$ cannot be implemented under $\mathcal{SMP}_n[adv : \mathrm{VSSC}_k(\infty)]$.*

Proof. We show the claim for $k = 2$ and $n = 3$ as it is straight-forward to derive the general case from this. We show that supposing some algorithm could implement Ω_k under the adversary leads to a contradiction. The following graph sequences (a)–(e) are all admissible sequences under $\mathrm{VSSC}_k(\infty)$ (we assume that nodes not depicted are isolated):

(a) $(p_3 \leftarrow p_1 \rightarrow p_2)_{r>0}$

(b) $(p_3 \leftarrow p_2 \rightarrow p_1)_{r>0}$

(c) $(p_2 \leftarrow p_3 \rightarrow p_1)_{r>0}$

(d) $(p_1 \rightarrow p_2)_{r>0}$

(e) $(p_1 \rightarrow p_3)_{r>0}$

Let $\varepsilon_a, \ldots, \varepsilon_e$ be the runs resulting from the above sequences applied to the same initial configuration. By Definitions B.2and B.4, LD has to include p_1 in ε_a, p_2 in ε_b, and p_3 in ε_c. By Definition B.2, in ε_d, because $\varepsilon_a \sim_{p_1} \varepsilon_d$ and $\varepsilon_c \sim_{p_3} \varepsilon_d$ in all rounds, for some $t > 0$, for all $t' > t$, $out(p_1, t') = \{p_1, p_3\}$. A similar argument shows that in ε_e, for some $t > 0$, for all $t' > t$, $out(p_1, t') = \{p_1, p_2\}$, because $\varepsilon_a \sim_{p_1} \varepsilon_e$ and $\varepsilon_b \sim_{p_2} \varepsilon_e$. The indistinguishability $\varepsilon_d \sim_{p_1} \varepsilon_e$ provides the required contradiction, as for some $t > 0$, for all $t' > t$, $out(p_1, t')$ should be the same in ε_d and ε_e. □