Theoretical Computer Science 726 (2018) 41–77

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

# Gracefully degrading consensus and $k$-set agreement in directed dynamic networks ☆

Martin Biely [a], Peter Robinson [b], Ulrich Schmid [c], Manfred Schwarz [c], Kyrill Winkler [c]

[a] EPFL IC IIF LSR, Station 14, CH-1015 Lausanne, Switzerland
[b] Department of Computing & Software, McMaster University, Canada
[c] Embedded Computing Systems Group, TU Wien, Vienna, Austria

ARTICLE INFO

*Article history:*
Received 24 August 2015
Received in revised form 19 February 2018
Accepted 21 February 2018
Available online 6 March 2018
Communicated by D. Peleg

*Keywords:*
Directed dynamic networks
Consensus
$k$-Set agreement
Message adversaries
Impossibility results
Lower bounds
Failure detectors

ABSTRACT

We study distributed agreement in synchronous directed dynamic networks, where an omniscient message adversary controls the presence/absence of communication links. We prove that consensus is impossible under a message adversary that guarantees weak connectivity only, and introduce eventually vertex-stable source components (VSSCs) as a means for circumventing this impossibility: A VSSC$(k, d)$ message adversary guarantees that, eventually, there is an interval of $d$ consecutive rounds where every communication graph contains at most $k$ strongly connected components consisting of the same processes (with possibly varying interconnect topology), which have no incoming links from outside processes. We present a consensus algorithm that works correctly under a VSSC$(1, 4E + 2)$ message adversary, where $E$ is the dynamic network depth. Our algorithm maintains local estimates of the communication graphs, and applies techniques for detecting network stability and univalent system configurations. Several related impossibility results and lower bounds, in particular, that neither a VSSC$(1, E - 1)$ message adversary nor a VSSC$(2, \infty)$ one allow to solve consensus, reveal that there is not much hope to deal with (much) stronger message adversaries here.

However, we show that gracefully degrading consensus, which degrades to general $k$-set agreement in case of unfavorable network conditions, allows to cope with stronger message adversaries: We provide a $k$-universal $k$-set agreement algorithm, where the number of system-wide decision values $k$ is not encoded in the algorithm, but rather determined by the actual power of the message adversary in a run: Our algorithm guarantees at most $k$ decision values under a VSSC$(n, d)$ + MAJINF$(k)$ message adversary, which combines VSSC$(n, d)$ (with some small value of $d$, ensuring termination) with some information flow guarantee MAJINF$(k)$ between certain VSSCs (ensuring $k$-agreement). Since related impossibility results reveal that a VSSC$(k, d)$ message adversary is too strong for solving $k$-set agreement and that some information flow between VSSCs is mandatory for this purpose as well, our results provide a significant step towards the exact solvability/impossibility border of general $k$-set agreement in directed dynamic networks.

Finally, we relate (the eventually-forever-variants of) our message adversaries to failure detectors. It turns out that even though VSSC$(1, \infty)$ allows to solve consensus and to implement the $\Omega$ failure detector, it does not allow to implement $\Sigma$. This contrasts the fact that, in asynchronous message-passing systems with a majority of process crashes, $(\Sigma, \Omega)$ is a weakest failure detector for solving consensus. Similarly, although the message

☆ This work has been supported the Austrian Science Fund (FWF) projects P20529 (PSRTS), P28182 (ADynNet), S11405 (RiSE) and P26436 (SIC).
*E-mail addresses:* martin.biely@biely.eu (M. Biely), peter.robinson@mcmaster.ca (P. Robinson), s@ecs.tuwien.ac.at (U. Schmid), mschwarz@ecs.tuwien.ac.at (M. Schwarz), kwinkler@ecs.tuwien.ac.at (K. Winkler).

https://doi.org/10.1016/j.tcs.2018.02.019

adversary VSSC$(n, d)$ + MAJINF$(k)$ allows to solve $k$-set agreement, it does not allow to implement the failure detector $\Sigma_k$, which is known to be necessary for $k$-set agreement in asynchronous message-passing systems with a majority of process crashes. Consequently, it is not possible to adapt failure-detector-based algorithms to work in conjunction with our message adversaries.



## 1. Introduction

Dynamic networks such as wireless sensor networks, mobile ad-hoc networks and vehicle area networks, are becoming ubiquitous nowadays. The primary properties of such networks are sets of participants (called processes in the sequel) that are a priori unknown and potentially changing, time-varying connectivity between processes, and the absence of a central control. Dynamic networks is an important and very active area of research [1].

Accurately modeling dynamic networks is challenging, for several reasons: First, process mobility, process crashes/recoveries, deliberate joins/leaves, and peculiarities in the low-level system design like duty-cycling (used to save energy in wireless sensor networks) make static communication topologies, as typically used in classic network models, inadequate for dynamic networks. Certain instances of dynamic networks, in particular, peer-to-peer networks [2] and inter-vehicle area networks [3], even suffer from significant churn, i.e., a large number of processes that can appear/disappear over time, possibly in the presence of faulty processes [4], and hence consist of a potentially unbounded total number of participants over time. More classic applications like *mobile ad-hoc networks* (MANETS) [5], wireless sensor networks [6,7] and disaster relief applications [8] typically consist of a *bounded* (but typically unknown) total number of processes.

Second, communication in many dynamic networks, in particular, in wireless networks like MANETS, is inherently broadcast: When a process transmits, then every other process within its transmission range will observe this transmission — either by legitimately receiving the message or as some form of interference. This creates quite irregular communication behavior, such as capture effects and near-far problems [9], where certain (nearby) transmitters may "lock" a receiver and thus prohibit the reception of messages from other senders. Consequently, the "health" of a wireless link between two processes may vary heavily over time [10]. For low-bandwidth wireless transceivers, an acceptable link quality usually even requires communication scheduling [11] (e.g., time-slotted communication) for reducing the mutual interference. Overall, this results in a frequently changing spatial distribution of pairs of nodes that can communicate at a given point in time.

As a consequence, many dynamic networks, in particular, wireless ones [12], are not adequately modeled by means of bidirectional links. Fading and interference phenomenons [13,14], including capture effects and near-far problems, are *local* effects that affect only the receiver of a wireless link: Given that the sender, which is also the receiver of the reverse link, resides at a different location, the two receivers are likely to experience very different levels of fading and interference [15]. This effect is even more pronounced in the case of time-slotted communication, where forward and backward links are used at different times. Consequently, the existence of asymmetric communication links cannot be ruled out in practice: According to [16], 80% of the links in a typical wireless network are sometimes asymmetric.

Despite these facts, most of the dynamic network research we are aware of assumes bidirectional links [17,18]. The obvious advantage of this abstraction is simplicity of the algorithm design, as strong communication guarantees obviously make this task easier. Moreover, it allows the re-use of existing techniques for wireline networks, which naturally support bidirectional communication. However, there are also major disadvantages of this convenient abstraction: First, for dynamic networks that operate in environments with unfavorable communication conditions, like in disaster relief applications or, more generally, in settings with various interferers and obstacles that severely inhibit communication, bidirectional links may simply not be achievable. Implementing distributed services in such settings thus require algorithms that do not need bidirectional links right from the outset. Second, the entire system needs to be engineered in such a way that bidirectional single-hop communication can be provided within bounded time. This typically requires relatively dense networks and/or processes that are equipped with powerful communication interfaces, which incur significant cost when compared to sparser networks or/and cheaper or more energy-saving communication devices. And last but not least, if directed single-hop communication was already sufficient to reach some desired goal (say, reaching some destination process) via multi-hop messages, waiting for guaranteed single-hop bidirectional communication would incur a potentially significant, unnecessary delay. Obviously, in such settings, algorithmic solutions that do not need bidirectional single-hop communication could be significantly faster.

In this paper, we thus restrict our attention to dynamic networks consisting of an *unknown but bounded* total number of processes, which are interconnected by *directed* communication links. The system is assumed to be synchronous,[1] hence time is measured in discrete *rounds* that allow the processes to exchange at most one message. Time-varying communication

[1] As synchronized clocks are typically required for basic communication in wireless systems anyway, e.g., for transmission scheduling and sender/receiver synchronization, this is not an unrealistic assumption: Global synchrony can be implemented directly at low system levels, e.g., via IEEE 1588 network time synchronization or GPS receivers, or at higher levels via time synchronization protocols like FTSP [19] or even synchronizers [20].

is modeled as a sequence of *communication graphs*, which contain a directed edge between two processes if the message sent in the corresponding round is successfully received. A bidirectional link is modeled by a pair of directed links that are considered independent of each other here.

A natural approach to build robust services despite the dynamic nature of such systems is to use some sort of distributed agreement on certain system parameters like action schedules and operating modes, as well as on application-level issues: Such a solution allows to use arbitrary algorithms for generating local proposals, which are supplied as inputs to a consensus algorithm that finally selects one of them consistently at all processes. As opposed to master-slave-based solutions, this approach avoids the single point of failure formed by the process acting as the master.

The ability to reach *system-wide* consensus is hence the most convenient abstraction one could provide here. The first major contribution of our paper is hence a suite of impossibility results and a consensus algorithm for directed dynamic networks that, to the best of our knowledge, works under the weakest communication guarantees sufficient for consensus known so far.

Obviously, however, one cannot reasonably assume that every dynamic network always provides sufficiently strong communication guarantees for solving consensus. Fortunately, weaker forms of distributed agreement are sufficient for certain applications. In case of determining communication schedules [11], for example, which are used for staggering message transmission of nearby nodes in time to decrease mutual interference, it usually suffices if those processes that have to communicate regularly with each other (e.g., for implementing a distributed service within a partition) agree on their schedule. A more high-level example would be agreement on rescue team membership [21] in disaster relief applications.

For such applications, suitably designed $k$-set agreement algorithms [22], where processes must agree on at most $k$ different values system-wide, are a viable alternative to consensus ($k = 1$). This is particularly true if such a $k$-set agreement (i) respects partitions, in the sense that processes in the same (single) partition decide on the same value, and (ii) is *gracefully degrading*, in the sense that the actual number $k$ of different decision values depends on the *actual* network topology in the execution: If the network is well-behaved, the resulting $k$ is small (ideally, $k = 1$), whereas $k$ may increase under unfavorable conditions. Whereas any gracefully degrading algorithm must be $k$-universal, i.e., unaware of any a priori information on $k$, it should ideally also be $k$-*optimal*, i.e., produce the smallest number $k$ of different decisions possible.

The second major contribution of our paper are several impossibility results for $k$-set agreement in directed dynamic networks, as well as the, to the best of our knowledge, first instance of a worst-case $k$-optimal $k$-set agreement, i.e., a consensus algorithm that indeed degrades *gracefully* to general $k$-set agreement.

**Detailed contributions and paper organization.**

In Section 3, we introduce our detailed system model, which builds upon and extends the *message adversary* notation used in [23]. It consists of an (unknown) number $n$ of processes, where communication is modeled by a sequence of directed communication graphs, one for each round: If some edge $(p, q)$ is present in the communication graph $\mathcal{G}^r$ of round $r$, then process $q$ has received the message sent to it by $p$ in round $r$. The message adversary determines the set of links actually present in every $\mathcal{G}^r$, according to certain constraints that may be viewed as network assumptions.

With respect to consensus, we provide the following contributions:

(1) In Section 4, we show that communication graphs that are weakly connected in every round are not sufficient for solving consensus, and introduce an additional assumption that allows to overcome this impossibility. For this, we rely on the graph-theoretic notion of a *source component* (strongly connected component that has no incoming edges from vertices outside) and its dynamic counterpart, the *vertex-stable source component* (VSSC), which describes a set of vertices, constituting a source component for multiple consecutive rounds. We note that every directed graph has at least one source component and that the detailed connection topology of a VSSC may change arbitrarily from round to round, as long as its vertices still form a source component in the communication graph. Our message adversary VSSC($d$) requires that the communication graph in every round is weakly connected and has a single (possibly changing) source component. Since this assumption is still too weak for solving consensus, VSSC($d$) also requires that, eventually, there will be $d$ consecutive rounds where some source component is vertex-stable. In Section 5, we provide a consensus algorithm that works in this model, and prove its correctness. Our algorithm requires a window of stability of $d = 4E + 2$ rounds, where $E \leqslant n - 1$ is the *dynamic network depth* of the network (= the number of rounds required to reach all processes in the network from every process in the vertex-stable source component via multi-hop communication).

(2) In Section 4, we also show that every deterministic consensus or leader election algorithm needs to know (a bound on) $E$ under VSSC($d$), i.e., that there is no universal algorithm. In addition, we prove that consensus is impossible both under VSSC($E - 1$) and VSSC($2, \infty$) (VSSC($x, y$) is essentially the same as VSSC($y$), except that it allows up to $x$ source components per round). Therefore, $E$ is a lower bound for the window of stability of VSSCs if vertex stability of source components is the only guarantee of the message adversary. Interestingly, the resulting dynamic networks fall between the weakest and second-weakest category in the classification of [24], and neither allow to solve classic problems such as reliable broadcast, atomic broadcast, or causal-order broadcast nor counting, $k$-verification, $k$-token dissemination, all-to-all token dissemination, and $k$-committee election.

With respect to $k$-set agreement and gracefully degrading consensus, we provide the following contributions:

(3) In Section 6, we provide a fairly weak natural message adversary VSSC$(k, d)$ that is still too strong for solving $k$-set agreement: It reveals that the restriction to at most $k$ simultaneous VSSCs in every round is *not* sufficient for solving $k$-set agreement if just a single VSSC is vertex-stable for less than $n - k$ rounds: A generic reduction of $k$-set agreement to consensus introduced in [25], in conjunction with certain bivalence arguments, is used to construct a non-terminating run in this case. Moreover, *eventual* stability of *all* VSSCs is also not enough for solving $k$-set agreement, not even when it is guaranteed that (substantially) less than $k$ VSSCs exist simultaneously. The latter is a consequence of some adversarial partitioning over time, which could happen in our dynamic networks.

(4) In Section 7, we show that the message adversary VSSC$(n, d)$ + MAJINF$(k)$, which combines VSSC$(n, d)$ (ensuring termination) with some information flow guarantee MAJINF$(k)$ between certain VSSCs (ensuring $k$-agreement), is sufficient for solving $k$-set agreement. Basically, MAJINF$(k)$ guarantees that if we choose $k + 1$ VSSCs, a *majority influence* chain between at least two of the chosen VSSCs exists. Despite being fairly strong, the resulting message adversary VSSC$(n, d)$ + MAJINF$(k)$ allows to implement a $k$-universal $k$-set agreement algorithm, which naturally respects partitions and is *worst-case k-optimal*, in the sense that no algorithm can solve $k - 1$-set agreement under VSSC$(n, d)$ + MAJINF$(k)$. To the best of our knowledge, it is the first gracefully degrading consensus algorithm proposed so far.

Finally, in the spirit of [23], we include a relation of our message adversaries to failure detectors. Whereas such a comparison obviously only makes sense for the eventually-forever-variants VSSC$(\infty)$ and VSSC$(n, \infty)$ + MAJINF$(k)$ of our message adversaries, it provides some very interesting insights:

(5) In Section 8, we show that even though VSSC$(1, \infty)$ allows to solve consensus and to implement the $\Omega$ failure detector, it does not allow to implement $\Sigma$. This contrasts the fact that, in asynchronous message-passing systems with a majority of process crashes, $(\Sigma, \Omega)$ is a weakest failure detector for solving consensus. Similarly, although the message adversary VSSC$(n, \infty)$ + MAJINF$(k)$ allows to solve $k$-set agreement, it does not allow to implement the failure detector $\Sigma_k$. Again, this is in contrast to the fact that $\Sigma_k$ is known to be necessary for $k$-set agreement in asynchronous message-passing systems with a majority of process crashes. One of the consequences of these findings is that it is not possible to adapt failure-detector-based algorithms to work in conjunction with our message adversaries.

## 2. Related work

Dynamic networks have been studied intensively in research (see the overview by Kuhn and Oshman [1] and the references therein). Besides work on peer-to-peer networks like [2], where the dynamicity of nodes (churn) is the primary concern, different approaches for modeling dynamic connectivity have been proposed, both in the networking context and in the context of classic distributed computing. Casteigts et al. [24] introduced a comprehensive classification of time-varying graph models.

**Models.** There is a rich body of literature on dynamic graph models going back to [26], which also mentions for the first time modeling a dynamic graph as a sequence of static graphs. A more recent paper using this approach is [17], where distributed computations are organized in lock-step synchronous rounds. Communication is described by a sequence of per-round communication graphs, which must adhere to certain network assumptions (like $T$-interval connectivity, which says that there is a common subgraph in any interval of $T$ rounds). Afek and Gafni [27] introduced message adversaries for specifying network assumptions in this context, and used them for relating problems solvable in wait-free read-write shared memory systems to those solvable in message-passing systems. Raynal and Stainer [23] also used message adversaries for exploring the relationship between round-based models and failure detectors.

Besides time-varying graphs, several alternative approaches that consider missing messages as failures have also been proposed in the past: Moving omission failures [28], round-by-round fault detectors [29], the heard-of model [30] and the perception-based failure model [31].

**Agreement problems.** Agreement problems in dynamic networks with undirected communication graphs have been studied in [18,32,33]; agreement in directed graphs has been considered in [34,35,27,23,36–39].

In particular, the work by Kuhn et al. [18] focuses on the $\Delta$-coordinated consensus problem, which extends consensus by requiring all processes to decide within $\Delta$ rounds of the first decision. Since they consider only undirected graphs that are connected in every round, without node failures, solving consensus is always possible. In terms of the classes of [24], the model of [40] is in one of the strongest classes (Class 10) in which every process is always reachable by every other process. On the other hand, [34,36] do consider directed graphs, but restrict the dynamicity by not allowing stabilizing behavior. Consequently, they also belong to quite strong classes of network assumptions in [24]. In sharp contrast, the message adversary tolerated by our algorithms does not guarantee bidirectional (multi-hop) communication between all processes, hence falls between the weakest and second weakest class of models defined in [24].

The solvability/impossibility border of consensus under message adversaries that support eventual stabilization has been explored in [35,37–39]. As it turned out, consensus can be solved for graph sequences where the *set* of graphs occurring in the sequence would render consensus impossible under an oblivious message adversary [28,41]. Whereas [35,37] are subsumed by the present paper, the algorithms presented in [38,39] allow to solve consensus for even shorter periods of stability.

The leader election problem in dynamic networks has been studied in [33,42], where the adversary controls the mobility of nodes in a wireless ad-hoc network. This induces dynamic changes of the (undirected) network graph in every round and requires any leader election algorithm to take $\Omega(Dn)$ rounds in the worst case, where $D$ is a bound on information propagation.

Regarding $k$-set agreement in dynamic networks, we are not aware of any previous work except [43], where bidirectional links are assumed, and our previous paper [44], where we assumed the existence of an underlying *static* skeleton graph (a non-empty common intersection of the communication graphs of all rounds) with at most $k$ *static* source components. Note that this essentially implies a directed dynamic network with a static core. By contrast, in this paper, we allow the directed communication graphs to be fully dynamic. In [45], we provided $k$-set agreement algorithms for partially synchronous systems with weak synchrony requirements.

**Degrading consensus problems.** We are also not aware of related work exploring gracefully degrading consensus or $k$-universal $k$-set agreement. However, there have been several attempts to weaken the semantics of consensus, in order to cope with partitionable systems and excessive faults. Vaidya and Pradhan introduced the notion of *degradable* agreement [46], where processes are allowed to also decide on a (fixed) default value in case of excessive faults. The *almost everywhere agreement* problem introduced by [47] allows a small linear fraction of processes to remain undecided. Aguilera et. al. [48] considered quiescent consensus in partitionable systems, which requires processes outside the majority partition not to terminate. None of these approaches is comparable to gracefully degrading $k$-set agreement, however: On the one hand, we allow more different decisions, on the other hand, all correct processes are required to decide and every decision must be the initial value of some process.

Ingram et. al. [49] presented an asynchronous leader election algorithm for dynamic systems, where every component is guaranteed to elect a leader of its own. Whereas this behavior clearly matches our definition of graceful degradation, contrary to decisions, leader assignments are revocable and the algorithm of [49] is guaranteed to successfully elect a leader only once the topology eventually stabilizes.

## 3. Model

We consider a synchronous distributed system made up of a fixed set of distributed processes $\Pi$ with $|\Pi| = n \geqslant 2$, which have fixed unique ids and communicate via unreliable message passing. Processes will be denoted by $p_i$, $p_j$ etc.

Similar to the $\mathcal{LOCAL}$ model [50], we assume that processes are deterministic state machines that organize their computations as an infinite sequence of communication-closed [51] lock-step rounds. For every $p_i \in \Pi$, $\mathbb{s}_i$ denotes its local state, taken from a potentially infinite state space. It also comprises an input variable $x_i$, which holds some fixed initial value $v_i$ at the beginning of an execution, and an output variable $y_i$, which is initially undefined ($\perp$) and can be changed to some value $\neq \perp$ exactly once. $\mathbb{s}_i^r$, $r \geqslant 1$, denotes the state at the end of round $r$, $\mathbb{s}_i^0$ denotes the initial state. In each round $r > 0$, each process performs three steps in the following order: First $p_i$ broadcast a message, then receives a subset of the messages sent in this round, and finally updates the state from $\mathbb{s}_i^{r-1}$ to $\mathbb{s}_i^r$, based on the messages received and $\mathbb{s}_i^{r-1}$. Note that processes do not know, without receiving explicit feedback in later rounds, which processes received their round $r$ broadcast.

The evolving nature of the network topology is modeled as an infinite sequence of simple directed graphs $\mathcal{G}^1, \mathcal{G}^2, \ldots$, which is determined by an omniscient *message adversary* [23,27] that may view the processes' internal states at any time. Given such a graph sequence and a set of initial states $\{\mathbb{s}_i^0 | p_i \in \Pi\}$, the corresponding *run* is the execution of system where $\mathcal{G}^r$ is used as the round $r$ communication graph. For our deterministic algorithms, a run is completely determined by the initial states of the processes and the sequence of communication graphs.

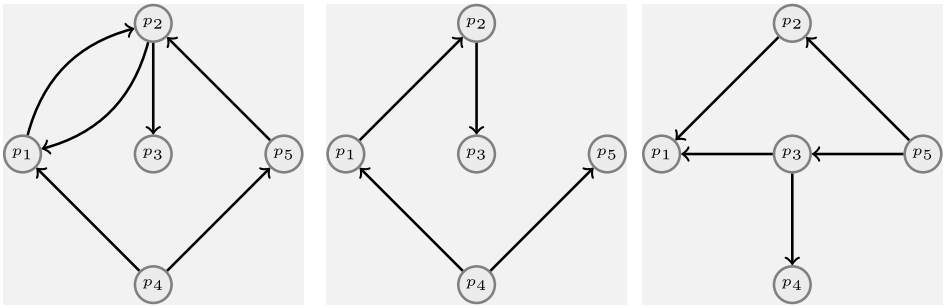
**Definition 1** *(Communication graph).* A *communication graph* $\mathcal{G} = \langle V, E \rangle$ is a simple directed graph on node set $V = \Pi$. An edge $(p_i \to p_j)$ is in $E$ if and only if $p_j$ successfully receives $p_i$'s message. For a given run, we denote the round $r$ communication graph by $\mathcal{G}^r = \langle V, E^r \rangle$. The set $\mathcal{N}_j^r$ denotes $p_j$'s *in-neighbors* in $\mathcal{G}^r$ (excluding $p_j$).

Note that we will sloppily write $(p_i \to p_j) \in \mathcal{G}^r$ to denote $(p_i \to p_j) \in E^r$, as well as $p_i \in \mathcal{G}^r$ to denote $p_i \in V = \Pi$. We emphasize again that $p_i$ does not have any *a priori* knowledge of its neighbors, i.e., $p_i$ does not know who receives its round $r$ broadcast, and does not know who it will receive from in round $r$ before its round $r$ computation.

Fig. 1 shows a sequence of communication graphs for a network of 5 processes, for rounds 1 to 3.

Since every $\mathcal{G}^r$ can range arbitrarily from $n$ isolated nodes to a fully connected graph, there is no hope to solve any non-trivial agreement problem without restricting the power of the adversary to drop messages[2] to some extent. Inspired by [23], we encapsulate a particular restriction, e.g., that every communication graph must be weakly connected, by means of a particular *message adversary*. Note that Definition 2 generalizes the notation introduced in [27], which just specified the *set* of communication graphs the adversary may choose from in every round, to sets of *sequences* of communication graphs.

[2] Even though the adversary can only affect communication in our model, it is also possible to model classic send and/or receive omission process failures [52] (and thereby also crash failures): A process that is send/receive omission faulty in round $r$ has no outgoing/incoming edges to/from some other processes in $\mathcal{G}^r$.




(a) $\mathcal{G}^1$ (b) $\mathcal{G}^2$ (c) $\mathcal{G}^3$

**Fig. 1.** Graph sequence with a single source component per round.

**Definition 2** *(Message adversary).* A message adversary *Adv* (for our system $\Pi$ of $n$ processes) is a set of sequences of communication graphs $(\mathcal{G}^r)_{r>0}$. A particular sequence of communication graphs $(\mathcal{G}^r)_{r>0}$ is *feasible* for *Adv*, if $(\mathcal{G}^r)_{r>0} \in Adv$.

Informally, we say that some message adversary *Adv* guarantees some property (like "all graphs are weakly connected"), called a *network assumption*, if every $(\mathcal{G}^r)_{r>0} \in Adv$ satisfies this property.

Complementing the traditional approach of partially ordering system models or unreliable failure detectors [53] via their problem solving power (task implementability), the restricted nature of our message adversaries allows us to employ a much simpler and direct way of relating those: For a fixed system $\Pi$ of $n$ processes, we say that $A$ is stronger than $B$ if and only if $A \supseteq B$, i.e., if $A$ can generate at least the communication graph sequences that can be generated by $B$. As a consequence, an algorithm that works correctly under message adversary $A$ will also work under $B \subseteq A$.

### 3.1. Consensus and k-set agreement

To formally introduce agreement problems, we consider some finite value domain $\mathcal{V}$ with $\perp \neq V$, and say that $p_i$ has *decided* in round $r$ (or state $s_i^r$ is decided) if $y_i^r = v \neq \perp$ in round $r$. If $y_i^{r-1} = \perp$ and $y_i^r = v \neq \perp$, we say that $p_i$ *decides* in round $r$ on $v$. Otherwise, it is (still) undecided. Note that, in the context of the particular algorithms introduced in later sections, we will sometimes also assign additional attributes to states.

**Definition 3** *(Consensus).* Algorithm $\mathcal{A}$ solves *consensus*, if the following properties hold in every run of $\mathcal{A}$:

(Agreement) If process $p_i$ decides on $v$ and $p_j$ decides on $v'$, then $v = v'$.
(Validity) If process $p_i$ decides on $v$, then $v$ is some $p_j$'s initial value $x_j$.
(Termination) Every process must eventually decide.

For the *k-set agreement problem* [22], we assume that both $|\mathcal{V}| > k$ and $n > k$ to rule out trivial solutions:

**Definition 4** *(k-set agreement).* Algorithm $\mathcal{A}$ solves *k-set agreement*, if the following properties hold in every run of $\mathcal{A}$:

($k$-Agreement) At most $k$ different decision values are obtained system-wide in any run.
(Validity) If process $p_i$ decides on $v$, then $v$ is some $p_j$'s initial value $x_j$.
(Termination) Every process must eventually decide.

Clearly, consensus is the special case of 1-set agreement; set agreement is a short-hand for $n-1$-set agreement.

We call a consensus or $k$-set agreement algorithm *universal*, if it does not have any a priori knowledge of the network (and hence of $n$). A $k$-set agreement algorithm is called *k-universal*, if it is universal and does not even require a priori knowledge of $k$.

### 3.2. Influence in dynamic networks

We will now establish what it means for a process $p_i$ to *influence* some process $p_j$, which is central in our paper. Note carefully that such an influence is always paired with time: In the spirit of [54,18], for a given sequence $(\mathcal{G}^r)_{r>0}$ of communication graphs, we say that process $p_i$ at the end of round $r$ influences $p_j$ in round $s$, denoted as $s_i^r \rightsquigarrow s_j^s$, if the state of process $p_i$ at the end of round $r$ could have affected the state of process $p_j$ at the end of round $s$. Clearly, in our system model, this requires process $p_i$ to send a message in round $r+1$ or later that (directly or indirectly, via some message chain) reaches $p_j$ at the latest in round $s$, so that it could affect its state $s_j^s$ reached at the end of round $s$.

Formally, this is defined via the influence relation given in Definition 5.

**Definition 5** *(Influence relation).* For a given run with sequence of communication graphs $(\mathcal{G}^r)_{r>0}$, the *influence relation* is the smallest relation that satisfies the following conditions for processes $p_i, p_j, p_k \in \Pi$ and rounds $r, r', r'' > 0$:

**LOCALITY:** $s_i^r \rightsquigarrow s_i^{r+1}$
**NEIGHBORHOOD:** $(p_i \to p_j) \in \mathcal{G}^{r+1} \Rightarrow s_i^r \rightsquigarrow s_j^{r+1}$
**TRANSITIVITY:** $s_i^r \rightsquigarrow s_j^{r'}$ and $s_j^{r'} \rightsquigarrow s_k^{r''} \Rightarrow s_i^r \rightsquigarrow s_k^{r''}$

### 3.3. Vertex-stable source components

We will now define the cornerstones of the message adversaries used in the remaining paper. Message adversaries such as VSSC($d$) (Definition 12) and VSSC($k, d$) (Definition 15) will be defined via the properties of the sequences of feasible communication graphs. Informally, most of those will rest on the pivotal concept of *source components*, which are strongly connected components in $\mathcal{G}^r$ without *incoming* edges from processes outside the component. The graphs generated by our message adversaries will be required to eventually guarantee source components that are vertex-stable, i.e., consist of the same *set* of nodes (with possibly varying interconnect) during a sufficiently large number of consecutive rounds. It will turn out that vertex-stability guarantees that eventually all members receive information from each other.

**Definition 6** *(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 $\forall p_i \in S, \forall p_j \in \mathcal{G}\colon (p_j \to p_i) \in \mathcal{G} \Rightarrow p_j \in S$.

By contracting strongly connected components (SCCs), it is easy to see that every weakly connected directed simple graph $\mathcal{G}$ has at least one source component, see Lemma 4. Hence, if $\mathcal{G}$ has $k$ source components, it has at most $k$ weakly connected components.

We now introduce vertex-stable source components as source components that remain the same for multiple rounds in a given graph sequence, albeit their actual interconnection topology may vary.

**Definition 7** *(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 $r \in I = [a, b]$, $b \geqslant a$, 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 $b$; hence, $I = [r, r]$ is not empty but rather represents round $r$.

The most important property of a $I$-VSSC is that information is guaranteed to spread among its vertices if the interval $I$ is large enough, as expressed in Corollary 1 below. To prove this, we need a few basic observations and lemmas. Our first observation is a direct consequence of the definition of a strongly connected component.

**Observation 1.** Let $C$ denote the set of processes of a strongly connected component of some graph $\mathcal{G}$, and $C'$ be any proper subset of $C$. Then, there exists a process $p_i \in C'$ s.t. $(p_i \to p_j) \in \mathcal{G}$ for some $p_j \in C \setminus C'$.

Based on influence and strongly connected components, we 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. The following Lemma 1 shows that if the number of rounds of the interval of vertex stability $|[a, b]|$ matches the size of the component minus 1, then for all $p_i \in C$, $s_i^{a-1}$ reaches every process of $C$ in round $b$ at latest.

**Lemma 1.** *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, $\forall p_i, p_j \in C$, it holds that $s_i^a \rightsquigarrow s_j^{a+|C|-1}$.*

**Proof.** For an arbitrary process $p_i$ in $C$, let $P_i^y \subseteq C$ be the set of processes $p_j$ of $C$ for which $s_i^a \rightsquigarrow s_j^y$ holds. Using induction on $y \geqslant a + 1$, we show that $|P_i^y| \geqslant \min\{y - a + 1, |C|\}$; as $y - a + 1 \geqslant |C|$ for $y \geqslant a + |C| - 1$, this proves our lemma.

For the induction start $y = a + 1$, as $C$ with $|C| > 1$ is the vertex-set of a strongly connected component in round $a + 1$, Observation 1 implies that $p_i$ has at least one neighbor such that $s_i^a \rightsquigarrow s_j^{a+1}$. By **LOCALITY**, we also have $s_i^a \rightsquigarrow s_i^{a+1}$, hence $|P_i^{a+1}| \geqslant 2 = \min\{2, |C|\}$ as required. For the induction step, assume $|P_i^y| \geqslant \min\{y - a + 1, |C|\}$, and consider two cases: (i) If $|P_i^y| < |C|$, then the induction hypothesis implies $y - a + 1 < |C|$, i.e., $y + 1 \in I$. By Observation 1, there is some process in $P_i^y$ that has at least one neighbor $p_j' \notin P_i^y$ in round $y + 1$, which, by **NEIGHBOURHOOD** and **TRANSITIVITY**, results in $|P_i^{y+1}| \geqslant \min\{y + 1 - a + 1, |C|\}$ as required. (ii) If already $|P_i^y| = |C|$, then by **LOCALITY** $|P_i^{y+1}| \geqslant |P_i^y|$, so $|P_i^{y+1}| = |C| \geqslant \min\{y + 1 - a + 1, |C|\}$ holds trivially. □

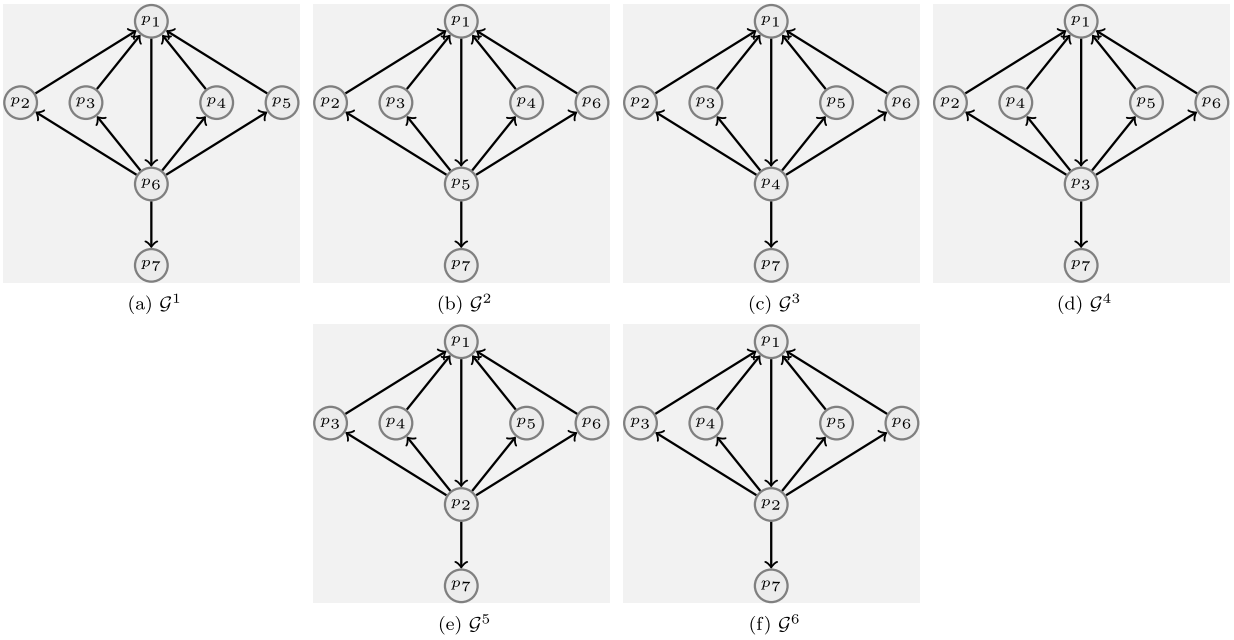



**Fig. 2.** Example graph sequence with constant diameter of 3 but dynamic source diameter and dynamic network depth in the order of $n$.

Corollary 1 follows immediately from Lemma 1 and the fact that, by definition, VSSCs are strongly connected components.

**Corollary 1.** *For every $I$-vertex-stable source component $S$ with $|S| > 1$ and $I = [a, b]$, it holds that $\forall p_i, p_j \in S$, $\forall x, y \in I$: $y \geqslant x + |S| - 2 \Rightarrow s_i^{x-1} \rightsquigarrow s_j^y$.*

In order to also model message adversaries that guarantee faster information propagation, Definition 8 introduces a system parameter $D$, called the *dynamic source diameter*. Informally, it guarantees that the message sent by a process $p_i \in S$, where $S$ is a $I$-VSSC with $|I| = D$, i.e., $I = [a, a + D - 1]$, in round $a$ can reach, directly or through message forwarding, every other process in $S$ by the end of round $a + D - 1$. Corollary 1 revealed that every sufficiently long $I$-VSSC $S$ guarantees $D \leqslant |S| - 1$; all sufficiently long VSSCs hence necessarily give $D \leqslant n - 1$. Choosing some $D < n - 1$ can be used to force the message adversary to speed-up information propagation accordingly. For example, we show in Section 3.4 that certain expander graph topologies ensure $D = O(\log n)$.

Analogous considerations apply for the *dynamic network depth* $E$ in communication graphs $\mathcal{G}^r$ with a single source component: As all graphs are weakly connected in this case (see Lemma 4), analogous versions of Lemma 1 and Corollary 1 are easily established.

**Definition 8** *($D$-bounded $I$-VSSC).* A $I$-VSSC $S$ is $D$-bounded with dynamic source diameter $D$, if $\forall p_i, p_j \in S$, $\forall r, r' \in I$: $r' \geqslant r + D - 1 \Rightarrow s_i^{r-1} \rightsquigarrow s_j^{r'}$.

**Definition 9** *($E$-influencing $I$-VSSC).* A $I$-VSSC $S$ is $E$-influencing with dynamic network depth $E$, if $\forall p_i \in S$, $\forall p_j \in \Pi$, $\forall r, r' \in I$: $r' \geqslant r + E - 1 \Rightarrow s_i^{r-1} \rightsquigarrow s_j^{r'}$.

We note that, by definition, for $|I| < D$ and $|I| < E$, an $I$-VSSC is trivially $D$-bounded and $E$-influencing. While it might be tempting to assume a connection between the graph diameter and the dynamic source diameter, resp. the dynamic network depth, in general, these notions are independent of each other. To illustrate this, Fig. 2 depicts an example where the graph diameter is constant even though, due to $p_1$, the dynamic source diameter and the dynamic network depth are in the order of the number of vertices. It is straightforward to generalize this example to $n$ vertices.

To formalize information propagation from source components to the rest of the network in the general case with more than a single source component per communication graph $\mathcal{G}^r$, one has to account for the fact that a process $p_j$ outside any source component could be reachable from *multiple* source components. Intuitively speaking, this allows modeling dynamic networks that do not "cleanly" partition. Similarly to Lemma 1, the following Lemma 2 shows that there is a guaranteed information propagation from at least one process of the set of VSSCs to every process in the system, provided all occurring source components are $I$-VSSCs with $|I| \geqslant n - 1$.

**Lemma 2.** *Let $n \geqslant 2$ and $R = \{S_1, S_2 \ldots S_\ell\}$ be a set of $\ell \geqslant 1$ $I$-VSSCs with $I = [a+1, a+n-1]$ such that, for any $r \in I$, every source component of $\mathcal{G}^r$ is in $R$. Then, for all $p_j \in \Pi$, it holds that $\exists S \in R$ such that $s_i^a \rightsquigarrow s_j^{a+n-1}$ for some $p_i \in S$.*

**Proof.** Let $R' = \bigcup_{S \in R} S$ denote the set of processes of all VSSCs of $R$. First, we show an analogue of Observation 1 for processes outside any source component of $R$: In every $\mathcal{G}^r$, $r \in I$, at least one process of $\Pi \setminus R'$ has an incoming edge from a process contained in some $S$ of $R$. Suppose that this is not the case. Then, contracting the strongly connected components of $\mathcal{G}^r$ yields at least one node, contracted entirely from nodes of $\Pi \setminus R'$, with no incoming edges. Hence, some source component of $\mathcal{G}^r$ consists entirely of nodes from $\Pi \setminus R'$ and thus cannot be in $R$. This contradicts the assumptions made on $R$.

Now, let $P_R(r)$ be the set of processes $p_j \in \Pi$ for which there exists some $S \in R$ such that $s_i^a \rightsquigarrow s_j^r$ holds for some $p_i \in S$. Using induction on $r \geqslant a+1$, we show that $|P_R(r)| \geqslant \min\{r-a+1, n\}$; as $r - a + 1 \geqslant n$ for $r \geqslant a+n-1$, this proves the lemma.

For the induction start $r = a+1$, **LOCALITY** implies that $P_R(a+1)$ contains all processes in $R'$, in addition to at least one process of $\Pi \setminus R'$, secured by our equivalent of Observation 1. Hence, $|P_R(a+1)| \geqslant 2 = \min\{2, n\}$ as required. For the induction step, assume $|P_R(r)| \geqslant \min\{r-a+1, n\}$, and consider two cases: (i) If $|P_R(r)| < n$, then the induction hypothesis implies $r - a + 1 < n$, i.e., $r + 1 \in I$. Since $R' \subseteq P_R(a+1) \subseteq P_R(r)$, there is at least one process $p'_j \notin P_R(r)$ that must be contained in $\Pi \setminus R'$; thus, **NEIGHBORHOOD** and **TRANSITIVITY** in conjunction with our equivalent of Observation 1 secure $|P_R(r+1)| \geqslant \min\{r+1-a+1, n\}$. (ii) If already $|P_R(r)| = n$, then $|P_R(r+1)| \geqslant |P_R(r)|$ by **LOCALITY**, so $|P_R(r+1)| = n \geqslant \min\{r+1-a+1, n\}$ holds trivially. □

Again, we introduce a parameter $H$ that allows a more fine-grained modeling of the information propagation in a dynamic network than just assuming the worst case $n-1$ secured by Lemma 2. For this purpose, Definition 10 generalizes Definition 9 from a single $I$-VSSC to a set $R$ of $I$-VSSCs. If $|I| \geqslant H$ it guarantees that every process in the network receives a message from some member of at least one $I$-VSSC of $R$ within $H$ rounds. Note carefully, though, that this does not necessarily imply that there exists an $E$-influencing $I$-VSSC. In the special case where $R$ is a singleton set, however, the sole member of $R$ is obviously a $E$-influencing VSSC.

**Definition 10** *($H$-influencing set of $I$-VSSCs).* A set $R = \{S_1, S_2 \ldots S_\ell\}$ of $\ell \geqslant 1$ $I$-VSSCs with $I = [a, b]$ is $H$-influencing with dynamic network depth $H$ if $\forall p_j \in \Pi$ $\exists S \in R$ s.t. $\forall r \in I$: if $r \geqslant a + H - 1$ then $s_i^{a-1} \rightsquigarrow s_j^r$ for some $p_i \in S$.

### *3.4. An example for $E$-influencing $I$-VSSCs with $E < n-1$: Expander topologies*

We conclude this section with an example of a network topology that guarantees that all $I$-VSSCs are $E$-influencing for some $E$ that is much smaller than $n-1$, which justifies why we introduce this parameter (as well as $D$) explicitly in our model.[3]

An undirected graph $\mathcal{G}$ is an *$\alpha$-vertex expander* if, for all sets $R \subset V(\mathcal{G})$ of size $\leqslant |V(\mathcal{G})|/2$, it holds that $\frac{|\mathcal{N}(R)|}{|R|} \geqslant \alpha$, where $\mathcal{N}(R)$ is the set of neighbors of $R$ in $\mathcal{G}$, i.e., those nodes in $V(\mathcal{G}) \setminus R$ that have a neighbor in $R$. (Explicit expander constructions can be found in [56].) As we need an expander property for *directed* communication graphs, we consider, for a vertex/process set $R$ and a round $r$, both the set $\mathcal{N}_+^r(R)$ of nodes outside of $R$ that are reachable from $R$ and the set of nodes $\mathcal{N}_-^r(R)$ that can reach $R$ in $r$. Definition 11 ensures an expansion property both for subsets $R$ chosen from source components (property (a)) and other processes (properties (b), (c)).

**Definition 11** *(Directed expander topology).* There is a fixed constant $\alpha$ and a fixed set $S$ such that the following conditions hold for all sets $R \subseteq V(\mathcal{G}^r)$:

*(a)* If $|R| \leqslant |S|/2$ and $R \subseteq S$, then $\frac{|\mathcal{N}_+^r(R) \cap S|}{|R|} \geqslant \alpha$ and $\frac{|\mathcal{N}_-^r(R) \cap S|}{|R|} \geqslant \alpha$.
*(b)* If $|R| \leqslant n/2$ and $S \subseteq R$, then $\frac{|\mathcal{N}_+^r(R)|}{|R|} \geqslant \alpha$.
*(c)* If $|R| \leqslant n/2$ and $S \cap R = \emptyset$, then $\frac{|\mathcal{N}_-^r(R)|}{|R|} \geqslant \alpha$.

The following Lemma 3 shows that (1) Definition 11 does not contradict the existence of a single source component and that (2) these expander topologies guarantee that $I$-VSSCs are both $D$-bounded with $D = O(\log n)$ and $E$-influencing with $E = O(\log n)$.

**Lemma 3.** *There are sequences of graphs $(\mathcal{G}^r)_{r>0}$ with a single source component in every $\mathcal{G}^r$ where Definition 11 holds and where, for any such run, every $I$-VSSC is $D$-bounded and $E$-influencing with $D = O(\log n)$ and $E = O(\log n)$.*

[3] An expander topology can be maintained in a dynamic network by using the protocol in [55].

**Proof.** We will first argue that *directed* graphs with a single source component exist that satisfy Definition 11. Consider the simple *undirected* graph $\bar{\mathcal{U}}$ that is the union of an $\alpha$-vertex expander on $S^I$ with member set $S$, and an $\alpha$-vertex expander on $V(\mathcal{G}^r)$. We turn $\bar{\mathcal{U}}$ into a directed graph by replacing every edge $(p_i, p_j) \in E(\bar{\mathcal{U}})$ with oriented directed edges $p_i \to p_j$ and $p_j \to p_i$. This guarantees Properties (a)–(c). In order to guarantee the existence of exactly one source component, we drop all directed edges pointing to $S^I$ from the remaining graph, i.e., we remove all edges $p_i \to p_j$ where $p_i \notin S$ and $p_j \in S$, which leaves Properties (a)–(c) intact and makes the $S$ from Definition 11 the single source component of the graph. We stress that the actual topologies chosen by the adversary might be quite different from this construction, which merely serves to show the existence of such graphs.

We also recall that our message adversaries like the one given in Definition 12 will rely on $I$-vertex-stable source components, which only require that the set of vertices remains unchanged, whereas the interconnect topology can change arbitrarily. Adding Definition 11 does of course not change this fact.

We will first show that the "per round" expander topology stipulated by Definition 11 is strong enough to guarantee that every sufficiently long VSSC is $D$-bounded with $D = O(\log n)$.

Let $S$ be some $I$-VSSC with $I = [a, b]$ and $|I| = \Omega(\log n)$. For $i \geqslant 1$, let $\mathcal{P}_i \subseteq S$ be the set of processes $p_j$ in $S$ such that $s_i^{a-1} \rightsquigarrow s_j^{a+i-1}$, and $\mathcal{P}_0 = \{p_i\}$. The result $D = O(\log n)$ follows immediately from Lemma 1 if $|S| \in O(\log n)$, so assume that $|S| \in \Omega(\log n)$ and consider some process $p_i \in S$. For round $a$, Property (a) yields $|\mathcal{P}_1| \geqslant |\mathcal{P}_0|(1+\alpha)$. In fact, for all $i$ where $|\mathcal{P}_i| \leqslant |S|/2$, we can apply Property (a) to get $|\mathcal{P}_{i+1}| \geqslant |\mathcal{P}_i|(1+\alpha)$, hence $|\mathcal{P}_i| \geqslant \min\{(1+\alpha)^i, |S|/2\}$. Let $\ell$ be the smallest value such that $(1+\alpha)^\ell > |S|/2$, which guarantees that $|\mathcal{P}_\ell| > |S|/2$. That is, $\ell = \left\lceil \frac{\log(|S|/2)}{\log(1+\alpha)} \right\rceil \in O(\log n)$. Now consider any $p_j \in S$ and define $\mathcal{Q}_{i-1} \subset S$ as the set of nodes that causally influence the set $\mathcal{Q}_i$ in round $a+i$, for $\mathcal{Q}_{2\ell+1} = \{p_j\}$. Again, by Property (a), we get $|\mathcal{Q}_{i-1}| \geqslant |\mathcal{Q}_i|(1+\alpha)$, so $|\mathcal{Q}_{2k-i}| \geqslant \max\{(1+\alpha)^i, |S|/2\}$. From the definition of $\ell$ above, we thus have $|\mathcal{Q}_\ell| > |S|/2$. Since $\mathcal{P}_\ell \cap \mathcal{Q}_\ell \neq \emptyset$, it follows that every $p_i \in S$ influences every $p_j \in S$ within $2\ell \in O(\log n)$ rounds. While the above proof has been applied to the starting round $x = a$ only, it is evident that it carries over literally also for any $x < s - 2\ell$, which shows that $S$ is indeed a $D$-bounded $I$-VSSC.

What remains to be shown is that $S$ is also a $E$-influencing VSSC with $E = O(\log n)$. We use Properties (b) and (c) similarly as in the above proof: For any round $x \in [r, s - 2k']$, we know by (b) that any process $p_i \in S$ has influenced at least $n/2$ nodes by round $x + k'$ where $k' = \lceil \log_{1+\alpha}(n/2) \rceil \in O(\log n)$ by arguing as for the $\mathcal{P}_i$ sets above. Now (c) allows us to reason along the same lines as for the sets $\mathcal{Q}_{i-1}$ above. That is, any $p_j$ in round $x + 2k'$ will be influenced by at least $n/2$ nodes. Therefore, any $p_i$ will influence every $p_j \in \Pi$ by round $x + 2k'$, which completes the proof. □

This confirms that sequences of communication graphs with $D < n - 1$ and $E < n - 1$ indeed exists and are compatible with message adversaries such as VSSC($d$) stated in Definition 12 below.

## 4. Consensus impossibilities and lower bounds

In this section, we will introduce a message adversary $\text{VSSC}_{D,E}(d)$ that allows to solve consensus for $d \geqslant 2D + 2E + 2$ in our model, and justify its particular properties by showing that relaxations lead to impossibilities. First and foremost, it requires that every $\mathcal{G}^r$ is rooted, i.e., contains only a single source component. Moreover, albeit the processes do not need to know $n$, they need a priori knowledge of the dynamic source diameter $D$ and the dynamic network depth $E$ from Definitions 8 and 9. And finally, our message adversary must guarantee that, eventually, a $d$-VSSC occurs. Interestingly, whereas $\text{VSSC}_{D,E}(d)$ allows to solve consensus for $d \geqslant 2D + 2E + 2$, it is too strong for solving other standard problems in dynamic networks such as reliable broadcasting.

Since consensus is trivially impossible for an unrestricted message adversary, which may just inhibit any communication in the system, it is natural to consider the question whether weakly connected communication graphs $\mathcal{G}^r$ in every round $r$ allow to solve consensus. However, it is not difficult to see that this does not work, even when all $\mathcal{G}^r = \mathcal{G}$ are the same, i.e., in a static topology: Consider the case where $\mathcal{G}$ contains two source components $S_1$ and $S_2$; such a graph obviously exists, cf. Lemma 4 below. If all processes in $S_1$ start with initial value 0 and all processes in $S_2$ start with initial value 1, they must decide on their own initial value (by validity and termination) and hence violate agreement. After all, no process in, say, $S_1$ ever has an incoming link from any process not in $S_1$.

Therefore, we restrict our attention to message adversaries that guarantee a *single* source component in $\mathcal{G}^r$ for any round $r$. Fig. 1 showed a sequence of graphs where this is the case. Some simple properties of such graphs are asserted by Lemma 4.

**Lemma 4.** *Any graph $\mathcal{G}$ contains at least one and at most n source components (isolated processes), which are all disjoint. If $\mathcal{G}$ contains a single source component $S$, then $\mathcal{G}$ is weakly connected, and there is a directed (out-going) path from every $p_i \in S$ to every $p_j \in \mathcal{G}$.*

**Proof.** We first show that every weakly connected directed simple graph $\mathcal{G}$ has at least one source component. To see this, contract every SCC to a single vertex and remove all resulting self-loops. The resulting graph $\mathcal{G}'$ is a directed acyclic graph (DAG) (and of course still weakly connected), and hence $\mathcal{G}'$ has at least one vertex $S$ (corresponding to some SCC in $\mathcal{G}$) that has no incoming edges. By construction, any such vertex $S$ corresponds to a source component in the original graph $\mathcal{G}$. Since $\mathcal{G}$ has at least 1 and at most $n$ weakly connected components, the first statement of our lemma follows.

To prove the second statement, we use the observation that there is a directed path from $u$ to $v$ in $\mathcal{G}$ if and only if there is a directed path from the vertex $C_u$ (containing $u$) to the vertex $C_v$ (containing $v$) in the contracted graph $\mathcal{G}'$. If there is only one source component in $\mathcal{G}$, the above observations imply that there is exactly one vertex $S$ in the contracted graph $\mathcal{G}'$ that has no incoming edges. Since $\mathcal{G}'$ is connected, $S$ has a directed path to every other vertex in $\mathcal{G}'$, which implies that every process $p_i \in S$ has a directed path to every vertex $p_j$, as required. □

Obviously, assuming a single source component makes consensus solvable if the source component is static (shown in detail in [44]). In this paper, we allow the source component to change throughout the run, i.e., the (single) source component $S$ of $\mathcal{G}^r$ might consist of a different set of processes in every round $r$. However, it will turn out that a sufficiently long interval of vertex-stability is indispensable for solving consensus in this setting. In the sequel, we will consider the message adversary $\mathrm{VSSC}_{D,E}(d)$ stated in Definition 12, which enforces the dynamic source diameter $D$ and the dynamic network depth $E \geqslant D$ and is parameterized by some stability window duration $d > 0$.

**Definition 12** *(Consensus message adversary $\mathrm{VSSC}_{D,E}(d)$).* For $d > 0$, the message adversary $\mathrm{VSSC}_{D,E}(d)$ 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.

Note that all the impossibility results and lower bounds in this section hold also when item (ii) is dropped or replaced by something weaker (like merely $D$-bounded VSSCs, as is done in Definition 15). Actually, it is only needed by the consensus algorithm in Section 5, and has been added already here solely for the purpose of avoiding two different definitions of essentially the same message adversary.

We first establish some general properties of the graph sequences generated by $\mathrm{VSSC}_{D,E}(d)$.

**Lemma 5** *(Properties of $\mathrm{VSSC}_{D,E}(d)$). In every sequence $(\mathcal{G}^r)_{r>0}$ of communication graphs feasible for $\mathrm{VSSC}_{D,E}(d)$,*

*(i) there is at least one process $p_i$ such that $\forall p_j \in \Pi$: $\mathrm{s}_i^0 \rightsquigarrow \mathrm{s}_j^{n(n-2)+1}$ holds, where $\mathrm{s}_i^0$ represents $p_i$'s initial state.*
*(ii) Conversely, for $n > 2$, the adversary can choose some sequence $(\mathcal{G}^r)_{r>0}$ where no process $p_i$ is causally influenced by all other processes $p_j$, i.e., $\nexists p_i \in \Pi$ s.t. $\exists y$ and $\forall p_j \in \Pi$: $\mathrm{s}_j^0 \rightsquigarrow \mathrm{s}_i^y$.*

**Proof.** Definition 12 guarantees that there is (at most) one source component in every $\mathcal{G}^r$, $r > 0$. Since we have infinitely many graphs in $(\mathcal{G}^r)_{r>0}$ but only finitely many processes, there is at least one process $p_i$ in the source component of $\mathcal{G}^r$ for infinitely many $r$. Let $r_1, r_2, \ldots$ be this sequence of rounds. Moreover, let $\mathcal{P}_0 = \{p_i\}$, and define for each $i > 0$ the set $\mathcal{P}_i = \mathcal{P}_{i-1} \cup \{p_j : \exists p'_j \in \mathcal{P}_{i-1} : p'_j \in \mathcal{N}_j^{r_i}\}$.

Using induction, we will show that $|\mathcal{P}_k| \geqslant \min\{n, k+1\}$ for $k \geqslant 0$. Consequently, by the end of round $r_{n-1}$ at latest, $p_i$ will have causally influenced all processes in $\Pi$. Induction base $k = 0$: $|\mathcal{P}_0| \geqslant \min\{n, 1\} = 1$ follows immediately from $\mathcal{P}_0 = \{p_i\}$. Induction step $k \to k+1$, $k \geqslant 0$: First assume that already $|\mathcal{P}_k| = n \geqslant \min\{n, k+1\}$; since $|\mathcal{P}_{k+1}| \geqslant |\mathcal{P}_k| = n \geqslant \min\{n, k+1\}$, we are done. Otherwise, consider round $r_{k+1}$ and $|\mathcal{P}_k| < n$: Since $p_i$ is in the source component of $G^{r_{k+1}}$, there is a path from $p_i$ to any process $p_j$, in particular, to any process $p_j$ in $\Pi \setminus \mathcal{P}_k \neq \emptyset$. Let $(v \to w)$ be an edge on such a path, such that $v \in \mathcal{P}_k$ and $w \in \Pi \setminus \mathcal{P}_k$. Clearly, the existence of this edge implies that $v \in \mathcal{N}_w^{r_{k+1}}$ and thus $w \in \mathcal{P}_{k+1}$. Since this implies $|\mathcal{P}_{k+1}| \geqslant |\mathcal{P}_k| + 1 \geqslant k + 1 + 1 = k + 2 = \min\{n, k+2\}$ by the induction hypothesis, we are done.

Finally, at most $n(n-2)+1$ rounds are needed until all processes $p_j$ have been influenced by $p_i$, i.e., $r_{n-1} \leqslant n(n-2)+1$: A pigeonhole argument reveals that at least one process $p_i$ must have been in the source component for $n-1$ times after so many rounds. After all, if every $p_i$ appeared at most $n-2$ times, we could fill up at most $n(n-2)$ rounds. By the above result, this is enough to secure that some $p_i$ influenced every $p_j$.

The converse statement (ii) follows directly from considering a static star, for example, i.e., a communication graph where there is one central process $p_c$, and for all $r$, $\mathcal{G}^r = \langle \Pi, \{(p_c \to p_j) | p_j \in \Pi \setminus \{p_c\}\}\rangle$. Clearly, $p_c$ cannot be causally influenced by any other process, and for $p_j, p_k \neq p_j \in \Pi \setminus \{p_c\}$ and $\forall x, y$ $\mathrm{s}_j^x \rightsquigarrow \mathrm{s}_k^y$ does not hold. On the other hand, this topology satisfies Definition 12, which includes the requirement of at most one source component per round. □

In the light of Lemma 5, it is interesting to relate the message adversary in Definition 12 to the classification of [24]: It is apparent that $\mathrm{VSSC}_{D,E}(d)$ belongs to a class that it is stronger than the weakest class that requests one node that eventually reaches all others, but weaker than the second-weakest class that requests one node that is reached by all. By contrast, models like [18,40] that assume bidirectionally connected graphs $\mathcal{G}^r$ in every round belong to the strongest classes (Class 10) in [24].

In Theorem 1, we will examine the solvability of several broadcast problems [40] under the message adversary $\text{VSSC}_{D,E}(d)$. It will turn out that none of these are implementable under our assumptions—basically, because there is no guarantee of (eventual) bidirectional communication. This is clearly in contrast to the usual strong bond between some of these problems and consensus in traditional settings.

**Theorem 1.** *The message adversary $\text{VSSC}_{D,E}(d)$ given in Definition 12, for any d, belongs to a class that is between the weakest and second-weakest in [24]. Neither* reliable broadcast, atomic broadcast, *nor* causal-order broadcast *can be implemented. Moreover, there is no algorithm that solves* counting, *k*-verification, *k*-token dissemination, all-to-all token dissemination, *and k*-committee election.

**Proof.** We first consider reliable broadcast, which requires that when a correct process broadcasts $m$, every correct process eventually delivers $m$. Suppose that the adversary chooses the communication graphs $\forall r: \mathcal{G}^r = \langle \{p_i, p_j, p_\ell\}, \{(p_i \to p_j), (p_j \to p_\ell)\}\rangle$, which matches Definition 12. Clearly, $p_j$ is a correct process in our model. Since $p_i$ never receives a message from $p_j$, $p_i$ can trivially never deliver a message that $p_j$ broadcasts.

For the token dissemination problems stated in [40], consider the same communication graphs and assume that there is a token that only $p_\ell$ has. Since no other process ever receives a message from $p_\ell$, token dissemination is impossible.

For counting, $k$-verification, and $k$-committee election, we return to the static star round graph $\mathcal{G}^r = \langle \Pi, \{(p_c \to p_j) | p_j \in \Pi \setminus \{p_c\}\}\rangle$ with central node $p_c$ considered in the proof of Lemma 5. As the local history of any process is obviously independent of $n$ here, it is impossible to solve any of these problems. □

### 4.1. Necessity of a priori knowledge of the dynamic network depth

We will now show that every correct solution for consensus, as well as for the related leader-election problem, requires some a priori knowledge of the dynamic network depth of the communication graphs generated by the adversary. Recall that a universal algorithm does not have any priori knowledge of the network, i.e., does not even know upper bounds for the dynamic network depth $E$ (and hence for $n$ and $D$).

**Theorem 2** *(Impossibility of universal consensus). There is no universal algorithm that can solve consensus under any message adversary $\text{VSSC}_{D,E}(d)$ as given in Definition 12, i.e., works correctly under $\text{VSSC}_{D,E}(d)$ for any choice of d.*

**Proof.** Assume for the sake of a contradiction that there is such a universal algorithm $\mathcal{A}$, w.l.o.g. for a set of input values $\mathcal{V}$ that contains 0 and 1. Consider a run $\alpha_v$ of $\mathcal{A}$ on a communication graph $\mathcal{G}$ that forms a (very large) static directed line rooted at process $p_i$ and ending in process $p_j$. Process $p_i$ has initial value $v \in \{0, 1\}$, while all other processes have initial value 0. Clearly, the universal algorithm $\mathcal{A}$ must allow $p_i$ to decide on $v$ by the end of round $\kappa$, where $\kappa$ is a constant (independent of $E$, $D$ and $n$; we assume that $n$ is large enough to guarantee $n - 1 > \kappa$). Next, consider a run $\beta_v$ of $\mathcal{A}$ that has the same initial states as $\alpha_v$, and communication graphs $(\mathcal{G}^r)_{r>0}$ that, during rounds $[1, \kappa]$, are also the same as in $\alpha_v$ (defining what happens after round $\kappa$ will be deferred). In any case, since $\alpha_v$ and $\beta_v$ are indistinguishable for $p_i$ until its decision round $\kappa$, it must also decide $v$ in $\beta_v$ at the end of round $\kappa$.

However, since $n > \kappa + 1$, $p_j$ has not been causally influenced by $p_i$ by the end of round $\kappa$. Hence, it has the same state $s_i^{\kappa+1}$ both in $\beta_v$ and in $\beta_{1-v}$. As a consequence, it cannot have decided by round $\kappa$: If $p_j$ decided $v$, it would violate agreement with $p_i$ in $\beta_{1-v}$. Now assume that runs $\beta_v$, $\beta_{1-v}$ are actually such that the stable window occurs later than round $\kappa$, i.e., $r_{ST} = \kappa + 1$, and that the adversary just reverses the direction of the line then: For all $\mathcal{G}^r$, $r \geqslant \kappa + 1$, $p_j$ is the source component and $p_i$ is the last process of the resulting topology. Observe that the resulting $\beta_v$ still satisfies Definition 12, since $p_j$ itself forms the only source component. Now, $p_j$ must eventually decide on some value $v'$ in some later round $\kappa'$, but since $p_j$ has been in the same state at the end of round $\kappa$ in both $\beta_v$ and $\beta_{1-v}$, it is also in the same state in round $\kappa'$ in both runs. Hence, its decision contradicts the decision of $p_i$ in $\beta_{1-v'}$. □

We now use a more involved indistinguishability argument to show that a slightly weaker problem than consensus, namely, leader election is also impossible to solve universally under the message adversary $\text{VSSC}_{D,E}(d)$. The classic leader election problem (cf. [57]) assumes that, eventually, exactly one process irrevocably elects itself as leader (by entering a special ELECTED state) and every other process elects itself as non-leader (by entering the NON-ELECTED state). Non-leaders are not required to know the process id of the leader.

Whereas it is easy to achieve leader election in our model when consensus is solvable, by just reaching consensus on the process ids in the system, the opposite is not true: Since the leader elected by some algorithm need not be in the source component that exists when consensus terminates, one cannot use the leader to disseminate a common value to all processes in order to solve consensus atop of leader election.

**Theorem 3** *(Impossibility of universal leader election). There is no universal algorithm that can solve leader election under any message adversary $\text{VSSC}_{D,E}(d)$ as given in Definition 12, i.e., works correctly under $\text{VSSC}_{D,E}(d)$ for any choice of d.*

**Proof.** We assume that there is a universal algorithm $\mathcal{A}$ that solves the problem. Consider the execution $\alpha_i(m)$ of $\mathcal{A}$ in a static unidirectional chain of $m$ processes, headed by process $p_i$: Since $p_i$ has only a single out-going edge and does not know $n$, it cannot know whether it has neighbors at all. Since it might even be alone in the single-vertex graph consisting of $p_i$ only, it must elect itself as leader in any $\alpha_i(m)$, $m \geqslant 1$, after some $T_i$ rounds ($T_i$ may depend on $i$, however, as we do not restrict $\mathcal{A}$ to be time-bounded).

Let $i$ and $j$ be two arbitrary different process ids, and let $T_i$ resp. $T_j$ be the termination times in the executions $\alpha_i(m)$ resp. $\alpha_j(m')$, for any $m$, $m'$; let $T = \max\{T_i, T_j\}$.

We now build a system consisting of $n = 2T + 3$ processes. To do so we assume a chain $\mathcal{G}_i$ of $T + 1$ processes headed by $p_i$ and ending in process $p_c$, a second chain $\mathcal{G}_j$ of $T + 1$ processes headed by $p_j$ and ending in process $p_k$, and the process $p_\ell$.

Now consider an execution $\beta$, which proceeds as follows: For the first $T$ rounds, the communication graph is the unidirectional ring created by connecting the above chains with edges $(p_k \to p_i)$, $(p_c \to p_\ell)$ and $(p_\ell \to p_j)$; its source component clearly is the entire ring. Starting from round $T + 1$ on, process $p_\ell$ forms the single vertex source component, which feeds, through edges $(p_\ell \to p_j)$ and $(p_\ell \to p_c)$ the two chains $\mathcal{G}_j$ and $\bar{\mathcal{G}}_i$, with $\bar{\mathcal{G}}_i$ being $\mathcal{G}_i$ with all edges reversed. Note that, from round $T + 1$ on, there is no edge connecting processes in $\mathcal{G}_i$ with those in $\mathcal{G}_j$ or vice versa.

Let $p_m$ be the process that is elected leader in $\beta$. We distinguish 2 cases:

1. If $p_m \in \mathcal{G}_j \cup \{p_\ell\}$, then consider the execution $\beta_i$ that is exactly like $\beta$, except that there is no edge $(p_k \to p_i)$ during the first $T$ rounds: $p_i$ is the single source component here. Clearly, for $p_i$, the execution $\beta_i$ is indistinguishable from $\alpha_i(2T+3)$ during the first $T_i \leqslant T$ rounds, so it must elect itself leader. However, since no process in $\mathcal{G}_j \cup \{p_\ell\}$ (including $p_c = p_m$) is causally influenced by $p_i$ during the first $T$ rounds, all processes in $\mathcal{G}_j \cup \{p_\ell\}$ have the same state after round $T$ (and all later rounds) in $\beta_i$ as in $\beta$. Consequently, $p_m$ also elects itself leader in $\beta_i$ as it does in $\beta$, which is a contradiction.
2. On the other hand, if $p_m \in \mathcal{G}_i$, we consider the execution $\beta_j$, which is exactly like $\beta$, except that there is no edge $(p_\ell \to p_j)$ during the first $T$ rounds: $p_j$ is the single source component here. Clearly, for $p_j$, the execution $\beta_j$ is indistinguishable from $\alpha_j(T+1)$ (made up of the chain $\mathcal{G}_j$) during the first $T_j \leqslant T$ rounds, so it must elect itself leader. However, since no process $p_c$ in $\mathcal{G}_i \cup \{p_\ell\}$ (including $p_c = p_m$) is causally influenced by $p_j$ during the first $T$ rounds, $p_c$ has the same state after round $T$ (and all later rounds) in $\beta_j$ as in $\beta$. Consequently, $p_m$ also elects itself leader $\beta_j$ as it does in $\beta$, which is again a contradiction.

This completes the proof of Theorem 3. □

### 4.2. Impossibility of consensus with too short stability intervals

The goal of this section is to show that some $I$-VSSC $S$ must be vertex-stable sufficiently long for solving consensus in our model. In essence, what is needed for this purpose is that every member of $S$ is able to reach the entire network. Recalling Definition 9, this requires $|I| \geqslant E$ and hence $d \geqslant E$ in Definition 12.

To show that $\mathrm{VSSC}_{D,E}(E)$ is indeed necessary in our setting, we will now consider a stronger message adversary $\mathrm{VSSC'}_{D,E}(E-1)$ given in Definition 14 below: It is stronger than $\mathrm{VSSC}_{D,E}(E)$ as its stability interval is shorter, but still slightly weaker than $\mathrm{VSSC}_{D,E}(E-1)$, in that it also guarantees one process to be reached from the processes in $S$ within $E$ rounds, despite the too short stability interval $I$. Note carefully that, since there is only one such process, it would be reached if $|I|$ was actually $E$. This property is formally captured by *almost* $E-1$-*influencing* VSSCs introduced in Definition 13, which is slightly weaker than Definition 9 in that $I$-VSSC's with $|I| = E - 1$ are no longer arbitrary.

**Definition 13** *(Almost $E-1$-influencing $I$-VSSC).* An $I$-VSSC $S$ with $I = [a, b]$ is almost $E-1$-influencing, with dynamic network depth $E > 0$, if either $|I| < E - 1$ or else $\forall x \in [a, b - E + 2]$ there exists a unique process $p_j \in \Pi$ such that $\forall p_i \in S\colon \mathrm{s}_i^{x-1} \rightsquigarrow \mathrm{s}_j^{x+E-1}$, while for all $p_k \in \Pi \setminus \{p_j\}$ we have $\forall p_i \in S\colon \mathrm{s}_i^{x-1} \rightsquigarrow \mathrm{s}_k^{x+E-2}$.

**Definition 14.** For $d > 0$ and $n > 2$, the message adversary $\mathrm{VSSC'}_{D,E}(d)$ 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 almost $E-1$-influencing $J$-vertex-stable source component.

Note carefully that Definition 14 allows the message adversary to choose *any* communication graph sequence that is consistent with the conditions stated therein. In particular, $\mathrm{VSSC'}_{D,E}(E-1)$ could also choose a sequence of communication graphs that guarantee dynamic network depth $E - 1$.

We have the following Lemma 6 that relates our message adversaries:

**Lemma 6.** *It holds that* $\text{VSSC}_{D,E}(E-1) \supseteq \text{VSSC'}_{D,E}(E-1)$ *and* $\text{VSSC}_{D,E}(E-1) \supseteq \text{VSSC}_{D,E}(E)$*, albeit* $\text{VSSC'}_{D,E}(E-1)$ *and* $\text{VSSC}_{D,E}(E)$ *are incomparable. Thus, in particular, every sequence of communication graphs generated by the message adversary* $\text{VSSC'}_{D,E}(E-1)$ *is also feasible for* $\text{VSSC}_{D,E}(E-1)$*.*

**Proof.** A comparison of Definition 14 and Definition 12 reveals that they differ only in item (iii). Since almost $E-1$-influencing is slightly weaker than $\text{VSSC}_{D,E}(E-1)$, as the adversary only needs to guarantee $s_i^{x-1} \rightsquigarrow s_j^{x+E-2}$ for every $p_i \in S$ (including the process $p_j$ exempted from this requirement in Definition 13) in the latter, $\text{VSSC}_{D,E}(E-1) \supseteq \text{VSSC'}_{D,E}(E-1)$ follows: Note carefully that all our message adversaries assume $D$-bounded and $E$-influencing VSSCs. The second statement $\text{VSSC}_{D,E}(E-1) \supseteq \text{VSSC}_{D,E}(E)$ follows immediately from Definition 12, as the adversary $\text{VSSC}_{D,E}(E-1)$ may of course also generate a VSSC that is vertex-stable for $E$ rounds. Finally, the incomparability of $\text{VSSC'}_{D,E}(E-1)$ and $\text{VSSC}_{D,E}(E)$ follows from the fact that (i) $\text{VSSC'}_{D,E}(E-1)$ cannot be forced to generate graph sequences that are vertex-stable for $E$ rounds, whereas (ii) $\text{VSSC}_{D,E}(E)$ cannot be forced to generate VSSCs that are almost $E-1$-influencing. □

We will now prove that the message adversary $\text{VSSC'}_{D,E}(E-1)$, and hence, by Lemma 6, also $\text{VSSC}_{D,E}(E-1)$, is too strong for solving consensus: Processes can *withhold* information from each other, which causes consensus to be impossible [34]. In order to simplify our proof, we assume that the adversary has to fix the start of $J = [r_{ST}, r_{ST}+E-2]$ and the set of source component members $S$ in the eventually generated $J$-VSSC $S$ before the beginning of the execution (but given the initial values). Note that this does not strengthen the adversary, and hence does not weaken our impossibility result: For deterministic algorithms, the whole execution depends only on the initial values and the sequence of the $\mathcal{G}^r$'s, so the adversary could simulate the execution and determine every $\mathcal{G}^{r+1}$ based on this.

**Lemma 7.** *Consider two runs of a consensus algorithm* $\mathcal{A}$ *under message adversary* $\text{VSSC'}_{D,E}(E-1)$*, for some a priori fixed* $J = [r_{ST}, r_{ST}+E-2]$ *and a corresponding* $J$*-VSSC* $S$*, which start from two univalent configurations* $C'$ *and* $C''$ *that differ only in the state of one process* $p_i$ *at the beginning of round* $r$*. Then,* $C'$ *and* $C''$ *cannot differ in valency.*

**Proof.** The proof proceeds by assuming the contrary, i.e., that $C'$ and $C''$ have different valency. We will then apply the same sequence of round graphs to extend the execution prefixes that led to $C'$ and $C''$ to get two different runs $e'$ and $e''$. It suffices to show that there is at least one process $p_j$ that cannot distinguish $e'$ from $e''$: This implies that $p_j$ will eventually decide on the same value in both executions, which contradicts the assumed different valency of $C'$ and $C''$.

Our choice of the round graphs depends on the following exhaustive cases:

(i) For $p_i \notin S$, we let the adversary choose any source component consisting of the processes in $S$, for all $\mathcal{G}^s$ with $s \geqslant r$. Obviously, every process (i.e., we can choose any) $p_j \in S$ has the same state throughout $e'$ and $e''$.

(ii) For $p_i \in S$ and $r \in J$, we choose any source component consisting of the processes in $S$ for all $\mathcal{G}^s$ with $r \leqslant s \leqslant r_{ST}+E-2$. For $s > r_{ST}+E-2$, we chose the source component $\{p_j\}$, where $p_j$ is the process that does not hear from any process in $S$ (and hence from $p_i$) within $J$ according to Definition 13. Hence, $p_j$ has the same state in $e'$ and $e''$, both during $J$ and afterwards, where it is the single source component.

(iii) For $p_i \in S$ and $r \notin J$, we choose graphs $\mathcal{G}^s$ where the source component is $\{p_j\}$ and $p_i$ has only in-edges for $r \leqslant s < r_{ST}$; $p_j$ (satisfying $p_j \notin S$ and hence $p_j \neq p_i$) is again the "distant" process allowed by Definition 13. From $s = r_{ST}$ on, we choose the same graphs $\mathcal{G}^s$ as in case (ii). It is again obvious that $p_j$ has the same state throughout $e'$ and $e''$, since $p_i$ cannot communicate to any process before $J$ and does not reach $p_j$ within $J$.

In any case, for process $p_j$, the sequence of states in the extensions starting from $C'$ and $C''$ is hence the same. Therefore, the two runs are indistinguishable for $p_j$, which cannot hence decide differently. This provides the required contradiction to the different valencies of $C'$ and $C''$. □

The next Lemma 8 establishes *connectedness* of the successor graphs of a configuration [34] and is a general property on graphs independent from the model used in this section. It is based upon constructing a sequence of graphs that differ only in one edge. Note that our construction is complicated by the fact that it must maintain $D$-boundedness of all intermediate graphs. We use $d_{\mathcal{G}}(v, w)$ denote the distance (number of edges on a shortest path) from $v$ to $w$ in graph $\mathcal{G}$. Subsequently, Lemma 9 shows that in the case where $n > 2$, we can even find connected successor graphs while avoiding a specific source component $S$.

**Lemma 8** *(Connectedness). For any two graphs* $\mathcal{G}'$ *and* $\mathcal{G}''$*, we can find a finite sequence of graphs* $\mathcal{G}', \mathcal{G}_1, \ldots \mathcal{G}_i \ldots \mathcal{G}''$*, each with a single source component, where any two consecutive graphs differ only by at most one edge. We say that the configurations* $C'$ *resp.* $C''$ *reached by applying* $\mathcal{G}'$ *resp.* $\mathcal{G}''$ *to the same configuration* $C$ *are* connected *in this case. Moreover, the following can be asserted:*

*(i) If the source components of* $\mathcal{G}'$ *and* $\mathcal{G}''$ *consist of the same set of processes* $S' = S'' = S$*, the same is true for all* $\mathcal{G}_i$ *and either* $\mathcal{G}' \subseteq \mathcal{G}_i$ *or* $\mathcal{G}'' \subseteq \mathcal{G}_i$*.*

*(ii) If $S' \neq S''$ and the source component $S_i$ of $\mathcal{G}_i$ is the same as the source component of $\mathcal{G} \in \{\mathcal{G}', \mathcal{G}''\}$, then either for all $v \in S_i$, for all $w \in \mathcal{G}_i$, $d_{\mathcal{G}_i}(v, w) \leqslant d_{\mathcal{G}}(v, w)$ or all but one node of $S_i$ have distance 1 to all other nodes, i.e., $|\{v \in S_i \mid \forall w \in \mathcal{G}_i, d_{\mathcal{G}_i}(v, w) = 1\}| \geqslant |S_i| - 1$.*

**Proof.** We describe how to construct the sequence by stating, for each step of our construction, which edge $e$ is modified in $\mathcal{G}_i$ in order to arrive at $\mathcal{G}_{i+1}$. Let $G' = \langle V, E' \rangle$ and let $G'' = \langle V, E'' \rangle$.

To show (i), we provide a construction that assumes $S' = S'' = S$. In the first phase of the construction, add some edge $e$ from $E'' \setminus E'$. When no such edge remains we have constructed the graph $\mathcal{G}_j = \mathcal{G}' \cup \mathcal{G}''$. We then commence the second phase by removing some $e$ from $E' \setminus E''$ until no such edge remains. For each $\mathcal{G}_i$ in the sequence constructed in this way, we have that either $\mathcal{G}' \subseteq \mathcal{G}_i$ or $\mathcal{G}'' \subseteq \mathcal{G}_i$, which implies that each graph has a single source component since $\mathcal{G}'$ and $\mathcal{G}''$ both have a single source component themselves. To see that all $\mathcal{G}_i$ have the same source component, suppose some $\mathcal{G}_i$ has a source component different from $S$. Hence an edge $(v, w)$ was added for $v \notin S$, $w \in S$ or all edges $(v, w)$ with $w \in S \setminus \{v\}$ were removed for some $v \in S$. Both contradict the assumption that $S' = S'' = S$, however.

To show (ii), we assume $S' \neq S''$ and, w.l.o.g., that there is some $u \in S' \setminus S''$ (if there is no such node $u$, we can still use the same construction, albeit with reversed direction). First, we add some edge $e$, chosen arbitrarily one-by-one, until we arrive at the complete graph. Since $u$ is always in the source component of each $\mathcal{G}_i$ generated this way (there is always a spanning tree rooted at $u$ contained in $\mathcal{G}_i$), each $\mathcal{G}_i$ has a single source component and no $\mathcal{G}_i$ has source component $S''$. Furthermore, $\mathcal{G}' \subseteq \mathcal{G}_i$ and hence (ii) holds. It remains to be shown that from the complete graph we can successively delete edges to arrive at $\mathcal{G}''$ while respecting (ii). For simplicity, we show that, equivalently, we can add edges to $\mathcal{G}''$ and arrive at the complete graph without ever violating (ii). We start by adding to $\mathcal{G}''$ the edge $e = (v \to w)$ for $v \in S''$, $w \in V$ until no such edge remains. Clearly adding an edge cannot increase any distances and hence (ii) is preserved. We continue by removing $e = (u \to v)$ for $v \in V \setminus \{u\}$. We observe that in the resulting graphs $\mathcal{G}_i$ the distance from the source component to any other node is 1, hence (ii) holds. Moreover, no new source component could have been generated this way, since $u$ already has some incoming edges by our construction. We note that in the following final steps only edges are added, hence no additional source component can be generated here as well. Fix some $v \in V \setminus \{u\}$ and add $e = (v \to w)$ for any $w \in V$. Still, for all nodes of $S_i \setminus \{v\}$, the distance to any other node is equal to 1, ascertaining (ii). Eventually, this yields some $\mathcal{G}_i$ where there is an edge $(v \to w)$ from every $v \in V \setminus \{u\}$ to each $w \in V$. Finally, we add $e = (u \to v)$ for $v \in V$ to arrive at the complete graph. Again, the distance from any node in $S_i \setminus \{u\}$ to any other node is equal to 1 and hence (ii) holds. □

**Lemma 9.** *Pick two arbitrary graphs $\mathcal{G}'$, $\mathcal{G}''$ with exactly one source component $S'$, $S''$, respectively. If $n > 2$ and given some non-empty $S$ with $S \neq S'$ and $S \neq S''$, there is a sequence of graphs $\mathcal{G}', \ldots, \mathcal{G}_i, \ldots, \mathcal{G}''$ such that any two consecutive graphs of the sequence differ in at most one edge and each $\mathcal{G}_i$ of the sequence has a unique source component that is different from $S$.*

**Proof.** We show that, for any graph $\overline{\mathcal{G}}'$ with source component $\overline{S}'$, there is such a sequence $\overline{\mathcal{G}}', \ldots, \overline{\mathcal{G}}_i, \ldots, \overline{\mathcal{G}}''$ for any graph $\overline{\mathcal{G}}''$ with source component $\overline{S}''$ if $\overline{S}''$ differs from $\overline{S}'$ in at most one node, i.e., $|\overline{S}' \setminus \overline{S}'' \cup \overline{S}'' \setminus \overline{S}'| \leqslant 1$. Repeated application of this fact implies the lemma, because, for $n > 2$, it is easy to find a sequence $S', \ldots, S_i, \ldots S''$ of subsets of $\Pi$ s.t. each two consecutive sets of the sequence differ from each other in at most one node and each set of the sequence is $\neq S$.

We sketch how to construct the desired graphs $\overline{\mathcal{G}}_i$ of the sequence in three phases.

Phase 1: Remove all edges (one by one) between nodes of $\overline{S}'$ until only a cycle (or, in general, circuit) remains, and then remove all edges between nodes outside of $\overline{S}'$ until only chains going out from $\overline{S}'$ remain. Let $\overline{\mathcal{G}}_j$ be the graph resulting from this operation and $\overline{S}_j$ be its source component.

Phase 2: If we need to add a node $p$ to $\overline{S}_j$, for some $q \in \overline{S}_j$, first add $(q \to p)$. For any $q' \neq q$, $(q' \to p) \in \overline{\mathcal{G}}_j$, where $p \neq q'$, remove $(q' \to p)$. Finally, add $(p \to q)$. If we need to remove a node $p$ from $\overline{S}_j$, for some $(q \to p)$, $(p \to q') \in \overline{\mathcal{G}}_j$, with $q, q' \in \overline{S}_j$, subsequently add $(q \to q')$ and $(q' \to q)$, then remove $(p \to q)$ and $(p \to q')$.

Phase 3: Since we now already have some graph $\overline{\mathcal{G}}_k$ with source component $\overline{S}''$, it is easy to add/remove edges one by one to arrive at the topology of $\overline{\mathcal{G}}''$. First, we add edges until the nodes of $\overline{S}''$ are completely connected among each other, the nodes not in $\overline{S}''$ are completely connected among each other, and there is an edge from every node of $\overline{S}''$ to each node not in $\overline{S}''$. Second, we remove the edges not present in $\overline{\mathcal{G}}''$. □

The proof of the following impossibility result follows roughly along the lines of the proof of [34, Lemma 3]. It shows, by means of induction on the round number, that a consensus algorithm $\mathcal{A}$ cannot reach a univalent configuration after any finite number of rounds.

**Theorem 4** *(Impossibility of consensus under $VSSC_{D,E}(E-1)$). There is no algorithm that solves consensus under the message adversary $VSSC'_{D,E}(E-1)$, and hence none under $VSSC_{D,E}(E-1)$ for $E > 1$ and $n > 2$.*

**Proof.** We follow roughly along the lines of the proof of [34, Lemma 3] and show per induction on the round number, that no algorithm $\mathcal{A}$ can reach a univalent configuration by round $r$, for any $r > 0$. Since no process can have decided in a bivalent configuration, this violates the termination property of consensus.

For the base case, we consider binary consensus only and argue similar to [58] but make use of our stronger validity property: Let $C_x^0$ be the initial configuration, where the processes with the smallest ids start with 1 and all others with 0. Clearly, in $C_0^0$ all processes start with 0 and in $C_n^0$ all start with 1, so the two configurations are 0- and 1-valent, respectively. To see that for some $x$ $C_x^0$ must be bivalent, consider that this is not the case, then there must be a $C_x^0$ that is 0-valent while $C_{x+1}^0$ is 1-valent. But, these configurations differ only in $p_i$, and so by Lemma 7 they cannot be univalent with different valency.

For the induction step we assume that there is a bivalent configuration $C$ at the beginning of round $r-1$, and show that there is at least one such configuration at the beginning of round $r$. We proceed by contradiction and assume all configurations at the beginning of round $r$ are univalent. Since $C$ is bivalent and all configurations at the beginning of $r$ are univalent, there must be two configurations $C'$ and $C''$ at the beginning of round $r$ which have different valency. Clearly, $C'$ and $C''$ are reached from $C$ by two different round $r-1$ graphs $\mathcal{G}' = \langle \Pi, E' \rangle$ and $\mathcal{G}'' = \langle \Pi, E'' \rangle$. As we explain in more detail below, we can apply Lemmas 8 and 9 to show that there is a sequence of applicable graphs such that $C'$ and $C''$ are connected. Each pair of subsequent graphs in this sequence differs only in one link $(v \to w)$, such that the resulting configurations differ only in the state of $w$. Moreover, if the source component in $\mathcal{G}'$ and $\mathcal{G}''$ is the same, all graphs in the sequence also have the same source component. Since the valency of $C'$ and $C''$ was assumed to be different, there must be two configurations $\overline{C}'$ and $\overline{C}''$ in the corresponding sequence of configurations that have different valency and differ only in the state of one process, say $p_i$. Applying Lemma 7 to $\overline{C}'$ and $\overline{C}''$ again produces a contradiction, and so not all successors of $C$ can be univalent.

It remains to be shown that Lemmas 8 and 9 indeed yield a sequence of applicable graphs, i.e., that extending the sequence of the $r-1$ graphs so far accordingly yields a prefix of some sequence of VSSC'$_{D,E}(E-1)$. By the assumptions of the theorem we may assume that $E > 1$ and $n > 2$, which allows us to apply all the claims of Lemma 9. Since all graphs in the sequence described by Lemmas 8 and 9 have exactly one source component, item (i) of Definition 14 is clearly satisfied.

Let $S^{r-1}$ denote the source component of $\mathcal{G}^{r-1}$, let $S'$, $S''$ denote the source component of $\mathcal{G}'$, $\mathcal{G}''$, respectively. If $S^{r-1} \neq S'$ and $S^{r-1} \neq S''$, Lemma 9 allows us to construct a sequence s.t. the source component of no $\mathcal{G}_i$ of the sequence is $S^{r-1}$. Therefore, $E$-influence of all VSSCs in the sequence is preserved and (ii) of Definition 14 holds in this case. If $S' = S''$, since both $\mathcal{G}'$ and $\mathcal{G}''$ preserved the $E$-influence, so does every $\mathcal{G}_i$ in the sequence described in item (i) of Lemma 8, because every $\mathcal{G}_i$ contains either $\mathcal{G}'$ or $\mathcal{G}''$. If $S' \neq S''$ and, say $S' = S^{r-1}$, then each $\mathcal{G}_i$ with source component $S_i = S'$ either preserves the distances from $S'$ to all other nodes or the distance from all but at most one node $p_j$ of $S'$ to all other nodes is 1, i.e., $\forall p_k \in S_i \setminus \{p_j\}, \forall p_\ell \in \Pi : d_{\mathcal{G}_i}(p_k, p_\ell) = 1$. In the former case, $E$-influence is preserved because $\mathcal{G}'$ preserved $E$-influence. In the latter case, $p_j$ reached at least one process $p_k \in S_i$ in round $r-1$ since it was part of $S^{r-1}$ by assumption. As $\forall p_k \in S_i \setminus \{p_j\}, \forall p_\ell \in \Pi : d_{\mathcal{G}_i}(p_k, p_\ell) = 1$, in addition to $s_k^{r-1} \rightsquigarrow s_\ell^r$, we have that $s_j^{r-1} \rightsquigarrow s_\ell^r$. Thus, $E$-influence is preserved for any $E > 1$ also in this case and item (ii) of Definition 14 is satisfied.

If $r \in J$, i.e., round $r$ is part of the stability phase, it follows that $\mathcal{G}'$ and $\mathcal{G}''$ have the same source component and so do all graphs in the sequence provided by item (i) of Lemma 8. Hence (iii) of Definition 14 also holds.

We have hence established that VSSC'$_{D,E}(E-1)$ is too strong for consensus, which implies the same for VSSC$_{D,E}(E-1)$ according to Lemma 6. □

## 5. A consensus algorithm for VSSC$_{D,E}(2D+2E+2)$

In this section, we show that it is possible to solve consensus under the message adversary VSSC$_{D,E}(2D+2E+2)$.

The underlying idea of our consensus algorithm is to use flooding to propagate the largest input value to everyone. However, as Definition 12 does not guarantee bidirectional communication between every pair of processes according to (ii) of Lemma 5, flooding is not sufficient: The largest input value could be hidden at a single process $p_i$ that never has outgoing edges. If such a process $p_i$ would never accept smaller values, it is impossible to reach agreement (without potentially violating validity). Thus, we have to find a way to force $p_i$ to accept also a smaller value.

A well-known technique to do so is *locking* a candidate value. Obviously, we do not want any process to lock its value, but rather some process(es) that will be able to impose their locked value, i.e., can successfully flood the system. In addition, we may allow processes that have successfully locked a value to decide only when they are sure that every other process has accepted their value as well. According to Definition 10, both can be guaranteed when these processes have been in a vertex stable source component long enough which is guaranteed by VSSC$_{D,E}(2D+2E+2)$.

The first major ingredient of our consensus algorithm is a network approximation algorithm (described in Section 5.1), which allows processes to detect their source component membership in (past) rounds. The core of our consensus algorithm (presented in Section 5.2) then exploits this knowledge for reaching agreement on locked values and imposes the resulting value on all processes in the network. As we will see, the main complication comes from the fact that a process can detect whether it has been part of the source component of round $r$ only with some latency.

### 5.1. The local network approximation algorithm

According to our system model, no process $p_i$ has any initial knowledge of the network. In order to learn about VSSCs, for example, it hence needs to *locally* acquire such knowledge. Process $p_i$ achieves this by means of Algorithm 1, which

**Algorithm 1** Local Network Approximation (Process $p_i$).

Provides externally callable function `InStableSource`($I$).

**Variables and Initialization:**
1: $A_i := \langle V_i, E_i \rangle$ initially $(\{p_i\}, \emptyset)$ // weighted digraph without multi-edges and loops
**Emit round $r$ messages:**
2: send $\langle A_i \rangle$ to all current neighbors
**Round $r$: computation:**
3: **for** $p_j \in \mathcal{N}_i^r$ and $p_j$ sent message $\langle A_j \rangle$ in $r$ **do**
4: **if** $\exists$ edge $e = (p_j \xrightarrow{T} p_i) \in E_i$ **then**
5: replace $e$ with $(p_j \xrightarrow{T'} p_i)$ in $E_i$ where $T' \leftarrow T \cup \{r\}$
6: **else**
7: add $e := (p_j \xrightarrow{\{r\}} p_i)$ to $E_i$
8: $V_i \leftarrow V_i \cup V_j$
9: **for** every pair of nodes $(p_k, p_\ell) \in V_i \times V_i$, $p_k \neq p_\ell$ **do**
10: **if** $T' = \bigcup \left\{ S \mid \exists p_j \in \mathcal{N}_i^r : (p_k \xrightarrow{S} p_\ell) \in E_j \right\} \neq \emptyset$ **then**
11: replace $(p_k \xrightarrow{T} p_\ell)$ in $E_i$ with $(p_k \xrightarrow{T \cup T'} p_\ell)$; add $(p_k \xrightarrow{T'} p_\ell)$ if no such edge exists

12: **function** `InStableSource`($I$)
13: Let $A_i|t$ be induced graph of $\left\{ (p_k \xrightarrow{T} p_\ell) \in E_i \mid t \in T \right\}$
14: Let $C_i|t$ be $A_i|t$ if it is strongly connected, or the empty graph otherwise.
15: **if** $\forall t_1, t_2 \in I : C_i := V(C_i|t_1) = V(C_i|t_2) \neq \emptyset$ **then**
16: return $C_i$
17: **else**
18: return $\emptyset$

maintains a *network estimate* $A_i$ in a local variable.[4] $A_i$ is a graph that holds the local estimates of every communication graph $\mathcal{G}^r$ that occurred so far, simply by labeling an edge $(p_i \to p_j)$ with the set of round numbers of every $\mathcal{G}^r$ once $p_i$ received evidence that $(p_i \to p_j)$ was present in round $r$.

Initially, $A_i$ consists of process $p_i$ only. In every round, every process $p_i$ broadcasts its current $A_i$ and fuses it with the network estimates received from its neighbors. In more detail, $p_i$ updates $A_i$ whenever $p_j \in \mathcal{N}_i^r$, by adding $(p_j \xrightarrow{\{r\}} p_i)$ if $p_j$ is $p_i$'s neighbor for the first time, or by updating the label of the edge $(p_j \xrightarrow{U} p_i)$ to $(p_j \xrightarrow{U \cup \{r\}} p_i)$ (line 5 and line 7). Moreover, $p_i$ also receives $A_j$ from $p_j$ and uses this information to update its own knowledge: The loop in line 9 ensures that $p_i$ has an edge $(v \xrightarrow{T \cup T'} w)$ for each $(v \xrightarrow{T'} w)$ in $A_j$, where $T$ is the set of rounds previously known to $p_i$.

Given $A_i$, we use $A_i|t$ with[5] $0 < t \leqslant r$ to denote the current estimate of $\mathcal{G}^t$ contained in $A_i$. Formally, $A_i|t$ is the graph induced by the set of edges

$$E_i|t = \left\{ e = (p_k \to p_\ell) \mid \exists T, t \in T : (p_k \xrightarrow{T} p_\ell) \in A_i \right\}.$$

As the information about $p_j$'s neighbors in $\mathcal{G}^t$ might take many rounds to reach some process $p_i$ (if it ever arrives at $p_i$), $A_i|t$ may never be fully up-to-date, and as only reported edges are added to the estimate (but not all reports need to reach $p_i$), $A_i|t$ will be an under-approximation of $\mathcal{G}^t$. For example, a process $p_i$ that does not have any incoming links from other processes, throughout the entire run of the algorithm, cannot learn anything about the remaining network, i.e., $A_i$ will permanently be the singleton graph.

Algorithm 1 finally provides an externally callable function `InStableSource`($I$), which will be used by the core consensus algorithm to find out whether the calling process $p_i$ was member in an $I$-VSSC $S$ and to query the set of all members of $S$. We will prove in Lemma 11 below that $p_i$ is a member of a $I$-VSSC if $A_i|t$ is strongly connected and consists of the same non-empty set $S$ of processes for all $t \in I$. Informally, this is due to the fact that the members of an $I$-VSSC will not be able to acquire knowledge of the topology outside $S$ within $I$, as they do not have incoming links from outside.

We start our analysis of Algorithm 1 with Lemma 10, which shows that $A_i|t$ underapproximates $\mathcal{G}^t$ in a way that consistently includes neighborhoods. Its proof uses the trivial invariant asserting $A_i|t = \langle \{p_i\}, \emptyset \rangle$ at the end of every round $r < t$.

**Lemma 10.** *If $A_i|t$ contains $(p_k \to p_\ell)$ at the end of some round $r$, then (i) $(p_k \to p_\ell) \in \mathcal{G}^t$, i.e., $A_i|t \subseteq \mathcal{G}^t$, and (ii) $A_i|t$ also contains $(p_m \to p_\ell)$ for every $p_m \in \mathcal{N}_\ell^t \subseteq \mathcal{G}^t$.*

[4] We denote the value of a variable $v$ of process $p_i$ at the end of its round $r$ computation as $v_i^r \in s_i^r$; we usually suppress the superscript when it refers to the current round.

[5] To simplify the presentation, we have refrained from purging outdated information from the network approximation graph. Actually, our consensus algorithm only queries `InStableSource` for intervals that span at most the last $2E + 1$ rounds, i.e., any older information could safely be removed from the approximation graph, resulting in a message complexity that is polynomial in $n$.

**Proof.** We first consider the case where $r < t$: At the end of round $r$, $A_i|t$ is empty, i.e., there are no edges in $A_i|t$. As the precondition of the Lemma's statement is false, the statement is true.

For the case where $r \geqslant t$, we proceed by induction on $r$:

Induction base $r = t$: If $A_i|t$ contains $(p_k \to p_\ell)$ at the end of round $r = t$, it follows from $A_j|t = \langle\{p_j\}, \emptyset\rangle$ at the end of every round $r < t$, for every $p_j \in \Pi$, that $p_\ell = p_i$, since $p_i$ is the only processor that can have added this edge to its graph approximation. Clearly, it did so only when $p_k \in \mathcal{N}_i^t$, i.e., $(p_k \to p_\ell) \in \mathcal{G}^t$, and included also $(p_m \to p_\ell)$ for every $p_m \in \mathcal{N}_i^t$ on that occasion. This confirms (i) and (ii).

Induction step $r \to r+1$, $r \geqslant t$: Assume, as our induction hypothesis, that (i) and (ii) hold for any $A_j|t$ at the end of round $r$, in particular, for every $p_j \in \mathcal{N}_i^{r+1}$. If indeed $(p_k \to p_\ell)$ in $A_i|t$ at the end of round $r+1$, it must be contained in the union of round $r$ approximations

$$U = (A_i|t) \cup \left( \bigcup_{p_j \in \mathcal{N}_i^{r+1}} A_j|t \right)$$

and hence in some $A_k|t$ with $k \in \{i, j\}$ at the end of round $r$. Note that the edges (labeled $r+1$) added in round $r+1$ to $A_i$ are irrelevant for $A_i|t$ here, since $t < r+1$.

Consequently, by the induction hypothesis, $(p_k \to p_\ell) \in \mathcal{G}^t$, thereby confirming (i). As for (ii), the induction hypothesis also implies that $(p_m \to p_\ell)$ is also in this $A_k|t$. Hence, every such edge must be in $U$ and hence in $A_i|t$ at the end of round $r+1$ as asserted. □

The following Lemma 11 shows that locally detecting $A_i|t$ to be strongly connected (in line 14 of Algorithm 1) implies that $p_i$ is in the source component of round $t$. This result rests on the fact that $A_i|t$ underapproximates $\mathcal{G}^t$ (Lemma 10.(i)), but does so in a way that never omits an in-edge at any process $p_j \in A_i|t$ (Lemma 10.(ii)).

**Lemma 11.** *If the graph $C_i|t$ (line 14) with $t < r$ is non-empty in round $r$, then $p_i$ is member of $S$, the source component of $\mathcal{G}^t$.*

**Proof.** For a contradiction, assume that $C_i|t$ is non-empty (hence $A_i|t$ is an SCC by line 14), but $p_i \notin S$. Since $p_i$ is always included in any $A_i$ by construction and $A_i|t$ underapproximates $\mathcal{G}^t$ by Lemma 10.(i), this implies that $A_i|t$ cannot be the source component of $\mathcal{G}^t$. Rather, $A_i|t$ must contain some process $p_k$ that has an in-edge $(p_j \to p_k)$ in $\mathcal{G}^t$ that is not present in $A_i|t$. As $p_k$ and hence some edge $(p_j \xrightarrow{t} p_k)$ is contained in $A_i|t$, because it is an SCC, Lemma 10.(ii) reveals that this is impossible. □

From the definition of the function `InStableSource`($I$) in Algorithm 1 and Lemma 11, we get the following Corollary 2.

**Corollary 2.** *If the function* `InStableSource`($I$) *evaluates to $S \neq \emptyset$ at process $p_i$ in round $r$, then $\forall x \in I$ where $x < r$, it holds that $p_i$ is a member of $S$ and $S$ is the source component of $\mathcal{G}^x$.*

The following Lemma 12 proves that, in a sufficiently long $I = [a, b]$ with a $I$-vertex-stable source component $S$, every member $p_i$ of $S$ detects an SCC for round $a$ (i.e., $C_i|a \neq \emptyset$) with a latency of at most $D$ rounds (i.e., at the end of round $a + D$). Informally speaking, together with Lemma 11, it asserts that if there is an $I$-vertex-stable source component $S$ for a sufficiently long interval $I$, then a process $p_i$ observes $C_i|a \neq \emptyset$ from the end of round $a + D$ on if and only if $p_i \in S$.

**Lemma 12.** *Consider an interval of rounds $I = [a, b]$, such that there is a $D$-bounded $I$-vertex-stable source component $S$ and assume $|I| = b - a + 1 > D$. Then, from the end of round $a + D$ onwards, we have $C_i|a = S$, for every process in $p_i \in S$.*

**Proof.** Consider any $p_j \in S$. At the beginning of round $a+1$, $p_j$ has an edge $(p_k \xrightarrow{T} p_j)$ in its approximation graph $A_j$ with $a \in T$ if and only if $p_k \in \mathcal{N}_j^a$. Since processes always merge all graph information from other processes into their own graph approximation, it follows from the definition of a $D$-bounded $I$-vertex-stable source component (Definition 8) in conjunction with the fact that $a + 1 \leqslant b - D + 1$ that every $p_i \in S$ has these in-edges of $p_j$ in its graph approximation by the end of round $a + 1 + D - 1$. Since $S$ is a vertex-stable source-component, it is strongly connected without in-edges from processes outside $S$. Hence $C_i|a = S$ from the end of round $a + D$ on, as asserted. □

This immediately gives us the following Corollary 3, which ensures that in a sufficiently long $I$-VSSC $S$, with $I = [a, b]$ and member set $S$, every $p_i \in S$ detects its membership in the $J$-VSSC $S$, $J = [a, b - D] \subseteq I$, with a latency of at most $D$ rounds.

**Algorithm 2** Solving Consensus; code for process $p_i$.

```
1: Simultaneously run Algorithm 1.

   Variables and Initialization:
2: x_i ∈ ℕ, initially own input value
3: locked_i, decided_i ∈ {false, true} initially false
4: lockRound_i ∈ ℤ initially 0

   Emit round r messages:
5: if decided_i then
6:    send ⟨DECIDE, x_i⟩ to all neighbors
7: else
8:    send ⟨lockRound_i, x_i⟩ to all neighbors

   Round r computation:
9: if not decided_i then
10:   if received ⟨DECIDE, x_j⟩ from any neighbor p_j then
11:      x_i ← x_j
12:      decide on x_i and set decided_i ← true

13:   else // p_i only received ⟨lock_j, x_j⟩ messages (if any):
14:      (lockRound_i, x_i) ← max{(lock_j, x_j) | p_j ∈ N_i^r ∪ {p_i}}  // lexical order in max
15:      if InStableSource([r − D − 1, r − D]) ≠ ∅ then
16:         if (not locked_i) then
17:            locked_i ← true
18:            lockRound_i ← r
19:         else
20:            if InStableSource([lockRound_i, lockRound_i + E]) ≠ ∅ then
21:               decide on x_i and set decided_i ← true
22:      else // InStableSource([r − D − 1, r − D]) returned ∅
23:         locked_i ← false
```

**Corollary 3.** *Consider an interval of rounds $I = [a, b]$, with $|I| = b - a + 1 > D$, such that there is a $D$-bounded vertex-stable source component $S$. Then, from the end of round $b$ on, a call to* `InStableSource`$([a, b - D])$ *returns $S$ at every process in $S$.*

Together, Corollaries 2 and 3 reveal that `InStableSource`(.) *precisely* characterizes the caller's actual membership in the $[a, b - D]$-VSSC $S$ in the communication graphs from the end of round $b$ on.

### 5.2. Core consensus algorithm for $VSSC_{D,E}(2D + 2E + 2)$

As explained in Section 5, the core consensus algorithm stated in Algorithm 2 builds upon the network approximation algorithm given as Algorithm 1: Relying on Corollary 2, every process uses `InStableSource` provided by Algorithm 1 to detect whether it has been in the vertex-stable source component of some past round(s). Since Corollary 3 reveals that `InStableSource` has a latency of up to $D \leqslant E$ rounds for reliably detecting that a process is in the vertex-stable source component of some (interval of) rounds, our algorithm (conservatively) looks back $D$ rounds in the past when locking a value.

In more detail, Algorithm 2 proceeds as follows: Initially, no process has locked a value, that is, $locked_i = \text{FALSE}$ and $lockRound_i = 0$. Processes try to detect whether they are privileged by evaluating the condition in line 15. When this condition is true in some round $\ell$, they lock the current value (by setting $locked_i = \text{TRUE}$ and *lockRound* to the current round), unless $locked_i$ is already TRUE. Note that our locking mechanism does not actually protect the value against being overwritten by a larger value being also locked in $\ell$; it locks out only those values that have older locks $l < \ell$.

When the process $p_m$ that had the largest value in the source component of round $\ell$ detects that it has been in a vertex-stable source component in all rounds $\ell$ to $\ell + E$ (line 20), it can decide on its current value. As all other processes in that source component must have had $p_m$'s value imposed on them, they can decide as well. After deciding, a process stops participating in the flooding of locked values, but rather (line 6) floods the network with $\langle \text{DECIDE}, x \rangle$. At the point when the stability window guaranteed by Definition 12 with $d = 2D + 2E + 2$ is large enough to allow every process to receive this message, all processes will eventually decide.

Before we turn our attention to the correctness proof of Algorithm 2, we need to define how the network approximation algorithm and the core consensus algorithm are combined to form a joint algorithm in our computation model. Let $m\_appr_i^{r-1}$ be the information to be broadcast by the network approximation algorithm and $m\_c_i^{r-1}$ the information to be broadcast by the consensus algorithm in round $r$. Process $p_i$ actually performs the following steps in round $r$:

(i) At the beginning of round $r$, broadcast a message containing $m\_appr_i^{r-1}$ and $m\_c_i^{r-1}$, which are both based on $\mathtt{s}_i^{r-1}$.
(ii) Receive all messages based on $\mathcal{G}^r$.
(iii) At the end of round $r$,

1. execute the computing step of the network approximation algorithm, using $m\_appr_j^{r-1}$ from all messages received in (ii).
2. execute the computing step of the consensus algorithm, using $m\_c_j^{r-1}$ from all messages received in (ii).

Note carefully that this joint execution scheme implies that when `InStableSource()` is called in step (iii.2) of the consensus algorithm, the network approximation algorithm is already in the state $s_{p_i}^r$ reached at the end of round $r$, so $A_i$ has already been updated with the information received in round $r$. Consequently, according to Corollaries 2 and 3, a call to `InStableSource`($I$) with $I = [a, b - D]$ by $p_i$ in the computing step at the end of round $b$ (or a later round) returns $S \neq \emptyset$ *precisely* when a $I$-VSSC $S$ containing $p_i$ existed.

Our correctness proof starts with the validity property of consensus according to Definition 3.

**Lemma 13** *(Validity). Every decision value is the input value of some process.*

**Proof.** Processes decide either in line 12 or in line 21. When a process decides via the former case, it has received a $\langle \text{DECIDE}, x_j \rangle$ message, which is sent by $p_j$ if and only if $p_j$ has decided on $x_j$ in an earlier round. In order to prove validity, it is thus sufficient to show that processes can only decide on some process' input value when they decide in line 21, where they decide on their current estimate $x_i$. Let the round of this decision be $r$. The estimate $x_i$ is either $p_i$'s initial value, or was updated in some round $r' \leqslant r$ in line 14 from a value received by way of one of its neighbors' $\langle lockRound, x \rangle$ message. In order to send such a message, $p_j$ must have had $x_j = x$ at the beginning of round $r'$, which in turn means that $x_j$ was either $p_j$'s initial value, or $p_j$ has updated $x_j$ after receiving a message in some round $r'' < r$. By repeating this argument, we will eventually reach a process that sent its initial value, since no process can have updated its decision estimate prior to the first round. □

The following Lemma 14 states a number of properties maintained by our algorithm when the first process $p_i$ has decided. Essentially, they say that there has been a vertex-stable source component in the interval $I = [\ell - D - 1, \ell + E]$ centered around the lock round $\ell$ (but not earlier), and asserts that all processes in that source component chose the same lock round $\ell$.

**Lemma 14.** *Suppose that process $p_i$ decides in round $r$, no decisions occurred before $r$, and $\ell = lockRound_i^{r-1}$, then*

(i) *$p_i$ is in the $I$-vertex-stable source component $S$ with $I = [\ell - D - 1, \ell + E]$,*
(ii) *$\ell + E \leqslant r \leqslant \ell + E + D$,*
(iii) *$S \neq S'$, where $S'$ is the source component of $G^{\ell - D - 2}$, and*
(iv) *all processes in $S$ executed line 18 in round $\ell$, and no process in $\Pi \setminus S$ can have executed line 18 in a round $\geqslant \ell$.*

**Proof.** Item (i) follows since line 15 has been continuously TRUE since round $\ell$ and from Lemma 11. As for item (ii), $\ell + E \leqslant r$ follows from the requirement of line 20, while $r \leqslant \ell + E + D$ follows from (i) and the fact that by Lemma 12 the requirement of line 20 cannot be, for the first time, fulfilled strictly after round $\ell + E + D$. From Lemma 12, it also follows that if $S = S'$, then the condition in line 15 would return true already in round $\ell - 1$, thus locking would occur already in round $\ell - 1$. Since $p_i$ did not lock in round $\ell - 1$, (iii) must hold. Finally, from (i), (iii), and Lemma 12, it follows that every other process in $S$ also has `InStableSource`$([\ell - D - 1, \ell - D])$ = TRUE in round $\ell$. Moreover, due to (iii), `InStableSource`$([\ell - 1 - D - 1, \ell - 1 - D])$ = FALSE in round $\ell - 1$, which causes all the processes in $S$ (as well as those in $\Pi \setminus S$) to set *locked* to 0. Since `InStableSource`$([\ell' - D - 1, \ell' - D])$ cannot become true for any $\ell' \geqslant \ell$ at a process $p_j \in \Pi \setminus S$, as $C_j | r = \emptyset$ for any $r \in I$ by Corollary 2, (iv) also holds. □

The following Lemma 15 asserts that if a process decides, then it has successfully imposed its proposal value on all other processes.

**Lemma 15** *(Agreement). Suppose that process $p_i$ decides in line 21 in round $r$ and that no other process has executed line 21 before $r$. Then, for all $p_j$, it holds that $x_j^{r-1} = x_i^{r-1}$.*

**Proof.** Using items (i) and (iv) in Lemma 14, we can conclude that $p_i$ was in $S$, the vertex-stable source component of rounds $\ell = lockRound_i^{r-1}$ to $\ell + E$, and that all processes in it $S$ have locked in round $\ell$. Therefore, in the interval $[\ell, \ell + E]$, $\ell$ is the maximal value of *lockRound*. More specifically, all processes $p_j$ in $S$ have $lockRound_j = \ell$, whereas all processes $p_k$ in $\Pi \setminus S$ have $lockRound_k < \ell$ during these rounds by Lemma 14.(iv). Let $p_m \in S$ have the largest proposal value $x_m^\ell = x_{max}$ among all processes in $S$. Since $p_m$ is in $S$, there is a causal chain of length at most $E$ from $p_m$ to any $p_j \in \Pi$. Note carefully that guaranteeing this property requires item (ii) of Definition 12, as the first decision (in round $r$) need not occur in the eventually guaranteed $2D + 2E + 2$-VSSC but already in some earlier "spurious" VSSC.

Since no process executed line 21 before round $r$, no process will send DECIDE messages in $[\ell, \ell + E]$. Thus, all processes continue to execute the update rule of line 14, which implies that $x_{max}$ will propagate along the aforementioned causal path to $p_j$. □

**Theorem 5** *(Consensus under $VSSC_{D,E}(2D + 2E + 2)$). Let $r_{ST}$ be the beginning of the stability window guaranteed by the message adversary $VSSC_{D,E}(2D + 2E + 2)$ given in Definition 12. Then, Algorithm 2 in conjunction with Algorithm 1 solves consensus by the end of round $r_{ST} + 2D + 2E + 1$.*

**Proof.** Validity holds by Lemma 13. Considering Lemma 15, we immediately get agreement: Since the first process $p_i$ that decides must do so via line 21, there are no other proposal values left in the system.

Observe that, so far, we have not used the liveness part of Definition 12. In fact, Algorithm 2 is always safe in the sense that agreement and validity are not violated, even if there is no vertex-stable source component.

We now show the termination property. By Corollary 3, we know that every process in $p_i \in S$ evaluates the predicate `InStableSource`$([r_{ST}, r_{ST} + 1]) =$ TRUE in round $\ell = r_{ST} + D + 1$, thus locking in that round. Furthermore, Definition 12 and Corollary 3 imply that at the latest in round $d = \ell + E + D$ every process $p_i \in S$ will evaluate the condition of line 20 to TRUE and thus decide using line 21. Thus, every such process $p_i$ will send out a message $m = \langle \text{DECIDE}, x_i \rangle$. By Definition 10 and Definition 12, we know that every $p_j \in \Pi$ will receive a DECIDE message at the latest in round $d + E = \ell + D + 2E = r_{ST} + 2D + 2E + 1$ and decide by the end of this round. □

We conclude our considerations regarding consensus under our eventually stabilizing message adversary $VSSC_{D,E}(d)$ by pointing out that the upper bound $2D + 2E + 2$ and the lower bound $E - 1$ of the stability interval $d$ match up to a small constant factor. Part of our on-going research is devoted to closing this gap.

## 6. Impossibilities and lower bounds for *k*-set agreement

In this section, we will turn our attention from consensus to general $k$-set agreement and prove related impossibility results and lower bounds. We will accomplish this by showing that certain "natural" message adversaries do not allow to solve $k$-set agreement. For example, as excessive partitioning of the system into more than $k$ source components makes $k$-set agreement trivially impossible, one natural assumption is to restrict the maximum number of source components per round in our system to $k$.

Definition 15 below defines the generic message adversary $VSSC_{D,H}(k, d)$, which allows at most $k$ VSSCs per round and guarantees a common window of vertex stability of duration at least $d$. Note that it involves both the dynamic source diameter $D$ and the dynamic network depth $H$ according to Definition 8 and Definition 10 (that have to be enforced by the message adversary).

**Definition 15** *(Message adversary $VSSC_{D,H}(k, d)$).* For $k > 0$, $d > 0$, the message adversary $VSSC_{D,H}(k, d)$ is the set of all sequences of communication graphs $(\mathcal{G}^r)_{r>0}$, where

(i) for every round $r$, $\mathcal{G}^r$ contains at most $k$ source components,
(ii) all vertex-stable source components occurring in any $(\mathcal{G}^r)_{r>0}$ are $D$-bounded,
(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 $H$-influencing set of $1 \leqslant \ell \leqslant k$ $D$-bounded $J$-vertex-stable source components $S_1, \ldots, S_\ell$.

Like for Definition 12, item (ii) has only been added for the sake of the $k$-set agreement algorithm Algorithm 4; the impossibility results and lower bounds also hold when (ii) is dropped or replaced by something weaker. Note that the message adversary $VSSC_{D,H}(k, 1)$ guarantees at most $k$ VSSCs in every $\mathcal{G}^r$, $r > 0$.

For $k = 1$, we can relate $VSSC_{D,H}(k, d)$ and $VSSC_{D,E}(d)$ given in Definition 15. First, $VSSC_{D,H}(1, d)$ differs from $VSSC_{D,E}(d)$ in that item (iii) rests on $H$-influencing VSSCs (Definition 10) rather than on $E$-influencing ones (Definition 8). Therefore, every run that is feasible w.r.t. item (iii) of $VSSC_{D,E}(d)$ for $E := H$ is also feasible w.r.t. item (iii) of $VSSC_{D,H}(1, d)$. Second, item (ii) of $VSSC_{D,E}(d)$ is also more demanding than item (ii) of $VSSC_{D,H}(1, d)$ as it requires all VSSCs to be both $D$-bounded and $E$-influencing. Consequently, it follows that $VSSC_{D,H}(d) \subseteq VSSC_{D,H}(1, d)$.

We will now prove that it is impossible to solve $k$-set agreement for $1 \leqslant k < n - 1$ under the message adversary $VSSC_{D,H}(k, \min\{n - k, H\} - 1)$, even under the slightly weaker version of this message adversary stated in Theorem 7 below. We will use the generic impossibility theorem provided in [25, Thm. 1] for this purpose. In a nutshell, the latter exploits the fact that $k$-set agreement is impossible if $k$ sufficiently disconnected components may occur and consensus cannot be solved in some component.

We first introduce the required definitions: Two executions of an algorithm $\alpha, \beta$ are *indistinguishable* (until decision) for a set of processes $\mathcal{D}$, denoted $\alpha \overset{\mathcal{D}}{\sim} \beta$, if for any $p_i \in \mathcal{D}$ it holds that $p_i$ executes the same state transitions in $\alpha$ and in $\beta$ (until it decides). Now consider a model of a distributed system $\mathcal{M} = \langle \Pi \rangle$ that consists of the set of processes $\Pi$ and a *restricted model* $\mathcal{M}' = \langle \mathcal{D} \rangle$ that is computationally compatible to $\mathcal{M}$ (i.e., an algorithm designed for a process in $\mathcal{M}$ can

be executed on a process in $\mathcal{M}'$) and consists of the set of processes $\mathcal{D} \subseteq \Pi$. Let $\mathcal{A}$ be an algorithm that works in system $\mathcal{M} = \langle \Pi \rangle$, where $\mathcal{M}_{\mathcal{A}}$ denotes the set of runs of algorithm $\mathcal{A}$ on $\mathcal{M}$, and let $\mathcal{D} \subseteq \Pi$ be a nonempty set of processes. Given any restricted system $\mathcal{M}' = \langle \mathcal{D} \rangle$, the *restricted algorithm* $A_{|\mathcal{D}}$ for system $\mathcal{M}'$ is constructed by dropping all messages sent to processes outside $\mathcal{D}$ in the message sending function of $\mathcal{A}$. We also need the following similarity relation between runs in computationally compatible systems (cf. [25, Definition 3]): Let $\mathcal{R}$ and $\mathcal{R}'$ be sets of runs, and $\mathcal{D}$ be a non-empty set of processes. We say that *runs $\mathcal{R}'$ are compatible with runs $\mathcal{R}$ for processes in $\mathcal{D}$*, denoted by $\mathcal{R}' \preccurlyeq_{\mathcal{D}} \mathcal{R}$, if $\forall \alpha \in \mathcal{R}'\ \exists \beta \in \mathcal{R} \colon \alpha \overset{\mathcal{D}}{\sim} \beta$.

**Theorem 6** *(k-Set agreement impossibility [25, Thm. 1]). Let $\mathcal{M} = \langle \Pi \rangle$ be a system model and consider the runs $\mathcal{M}_A$ that are generated by some fixed algorithm $A$ in $\mathcal{M}$, where every process starts with a distinct input value. Fix some nonempty and pairwise disjoint sets of processes $\mathcal{D}_1, \ldots, \mathcal{D}_{k-1}$, and a set of distinct decision values $\{v_1, \ldots, v_{k-1}\}$. Moreover, let $\mathcal{D} = \bigcup_{1 \leqslant h < k} \mathcal{D}_h$ and $\overline{\mathcal{D}} = \Pi \setminus \mathcal{D}$. Consider the following two properties:*

(**dec-$\mathcal{D}$**) *For every set $\mathcal{D}_h$, value $v_h$ was proposed by some $p_i \in \mathcal{D}$, and there is some $p_j \in \mathcal{D}_h$ that decides $v_h$.*
(**dec-$\overline{\mathcal{D}}$**) *If $p_j \in \overline{\mathcal{D}}$ then $p_j$ receives no messages from any process in $\mathcal{D}$ until every process in $\overline{\mathcal{D}}$ has decided.*

*Let $\mathcal{R}_{(\overline{\mathcal{D}})} \subseteq \mathcal{M}_A$ and $\mathcal{R}_{(\mathcal{D},\overline{\mathcal{D}})} \subseteq \mathcal{M}_A$ be the sets of runs of $A$ where (dec-$\overline{\mathcal{D}}$) respectively both, (dec-$\mathcal{D}$) and (dec-$\overline{\mathcal{D}}$), hold.[6] Suppose that the following conditions are satisfied:*

**(A)** *$\mathcal{R}_{(\overline{\mathcal{D}})}$ is nonempty.*
**(B)** *$\mathcal{R}_{(\overline{\mathcal{D}})} \preccurlyeq_{\overline{\mathcal{D}}} \mathcal{R}_{(\mathcal{D},\overline{\mathcal{D}})}$.*

*In addition, consider a restricted model $\mathcal{M}' = \langle \overline{\mathcal{D}} \rangle$ such that the following properties hold:*

**(C)** *There is no algorithm that solves consensus in $\mathcal{M}'$.*
**(D)** *$\mathcal{M}'_{A_{|\overline{\mathcal{D}}}} \preccurlyeq_{\overline{\mathcal{D}}} \mathcal{M}_A$.*

*Then, $A$ does not solve $k$-set agreement in $\mathcal{M}$.*

The proof of Theorem 7 below utilizes Theorem 6 in conjunction with the impossibility of consensus under $\mathrm{VSSC}_{D,E}(E-1)$ established in Theorem 4.

**Theorem 7** *(Impossibility of $k$-set agreement under $\mathrm{VSSC}_{D,H}(k, \min\{n-k, H\}-1)$). There is no algorithm that solves $k$-set agreement with $n > k+1$ processes under the message adversary $\mathrm{VSSC}_{D,H}(k, \min\{n-k, H\}-1)$ stated in Definition 15, for any $1 \leqslant k < n-1$, even if there are $k-1$ source components $S_1, \ldots, S_{k-1}$ that are vertex-stable all the time, i.e., in $[1, \infty]$ (and only source component $S_k$ is vertex-stable for at most $\min\{n-k, H\}-1$ rounds).*

**Proof.** Suppose that there is a $k$-set algorithm $\mathcal{A}$ that works correctly under the assumptions of our theorem. For $k = 1$, recalling $\mathrm{VSSC}_{D,H}(d) \subseteq \mathrm{VSSC}_{D,H}(1, d)$, Theorem 7 is implied by Theorem 4 with $E := H \leqslant n-1$.

To prove our theorem for $k > 1$, we will show that the conditions of the generic Theorem 6 are satisfied, thereby providing a contradiction to the assumption that $\mathcal{A}$ exists. Let $\mathcal{D}_i = \{p_i\}$ for $0 < i \leqslant k-1$ and let $\mathcal{D} = \bigcup_{i=1}^{k-1} \mathcal{D}_i$. Consequently, $\overline{\mathcal{D}} = \{p_k, p_{k+1}, \ldots, p_n\}$ and $|\overline{\mathcal{D}}| \geqslant 2$.

**(A)** The set of runs $\mathcal{R}_{(\overline{\mathcal{D}})}$ of $\mathcal{A}$ where no process in $\overline{\mathcal{D}}$ receives any message from $\mathcal{D}$ before it decides is nonempty: We choose the communication graph in every round to be such that $\overline{\mathcal{D}}$ has no incoming links from $\mathcal{D}$ until every process in $\overline{\mathcal{D}}$ has decided. Since any such sequence of communication graphs satisfies the assumptions of our theorem, $\mathcal{R}_{(\overline{\mathcal{D}})} \neq \emptyset$.

**(B)** The set of runs $\mathcal{R}_{(\mathcal{D},\overline{\mathcal{D}})}$ of $\mathcal{A}$ where both (i) some process in every $D_i$ decides $v_i$ and (ii) no process in $\overline{\mathcal{D}}$ receives any message from $\mathcal{D}$ before it decides satisfies $\mathcal{R}_{(\overline{\mathcal{D}})} \preccurlyeq_{\overline{\mathcal{D}}} \mathcal{R}_{(\mathcal{D},\overline{\mathcal{D}})}$: Let $\mathcal{H}$ be the set of runs where processes $p_i$ have unique input values $x_i = i$, $0 < i < k$, the communication graph in every round is such that $p_1, \ldots, p_{k-1}$ are isolated, and $p_k, \ldots, p_n$ are weakly connected (with a single source component) until every process has decided. By the assumptions of our theorem, $\mathcal{H}$ is non-empty. Since (i) the processes in $\overline{\mathcal{D}}$ never receive a message from a process in $\mathcal{D}$ in both $\mathcal{R}_{(\overline{\mathcal{D}})}$ and $\mathcal{H}$, and (ii) the initial values of the processes in $\overline{\mathcal{D}}$ are not restricted in $\mathcal{H}$ in any way, it is easy to find, for any run $\rho \in \mathcal{R}_{(\overline{\mathcal{D}})}$, a run $\rho' \in \mathcal{H}$ such that $\rho \overset{\overline{\mathcal{D}}}{\sim} \rho'$. Because obviously $\mathcal{H} \subseteq \mathcal{R}_{(\mathcal{D},\overline{\mathcal{D}})}$, we have established $\mathcal{R}_{(\overline{\mathcal{D}})} \preccurlyeq_{\overline{\mathcal{D}}} \mathcal{R}_{(\mathcal{D},\overline{\mathcal{D}})}$.

**(C)** Consensus is impossible in $\mathcal{M}' = \langle \overline{\mathcal{D}} \rangle$: Let $\overline{\mathcal{D}}$ be the partition containing the $k$th source component $S_k$, which is perpetually changing in every round, except for some interval of rounds $I = [r_{ST}, r_{ST} + \ell - 1]$, where $\ell = \min\{n-k, H\} - 1$, for

[6] Note that $\mathcal{R}_{(\overline{\mathcal{D}})}$ is by definition compatible with the runs of the restricted algorithm $A_{|\overline{\mathcal{D}}}$.

some fixed $r_{ST}$. During this interval, let the topology of $\overline{\mathcal{D}}$ be such that there exists some $p_i \in R_k$ and some $p_j \in \overline{\mathcal{D}}$ such that $\mathtt{s}_i^{r_{ST}-1} \rightsquigarrow \mathtt{s}_j^{r_{ST}+\ell}$ but not $\mathtt{s}_i^{r_{ST}-1} \rightsquigarrow \mathtt{s}_j^{r_{ST}+\ell-1}$.

Since $|\overline{\mathcal{D}}| = n-k+1$, such a topology (e.g. a chain with head $p$ and tail $q$) can be created by the message adversary $\mathrm{VSSC}_{D,H}(\ell)$, which is even stronger than the corresponding message adversary underlying Theorem 4. Hence, consensus is impossible in $\overline{\mathcal{D}}$.

**(D)** $\mathcal{M}'_{\mathcal{A}_{|\overline{\mathcal{D}}}} \preccurlyeq_{\overline{\mathcal{D}}} \mathcal{M}_{\mathcal{A}}$: Fix any run $\rho' \in \mathcal{M}'_{\mathcal{A}_{|\overline{\mathcal{D}}}}$ and consider a run $\rho \in \mathcal{M}_{\mathcal{A}}$, where every process in $\overline{\mathcal{D}}$ has the same sequence of state transitions in $\rho$ as in $\rho'$. Such a run $\rho$ exists, since the processes in $\overline{\mathcal{D}}$ can be disconnected from $\mathcal{D}$ in every round in $\mathcal{M}_{\mathcal{A}}$, so $\rho \overset{\overline{\mathcal{D}}}{\sim} \rho'$. □

Since Theorem 7 tells us that no $k$-set agreement algorithm (for $1 \leqslant k < n-1$) can *terminate* with insufficient concurrent stability of the at most $k$ source components in the system, it is tempting to assume that $k$-set agreement becomes solvable if a round exists after which all communication graphs remain the same. However, we will prove in Theorem 8 below that this is not the case for any $1 < k \leqslant n-1$. We will again use the generic Theorem 6, this time in conjunction with the variant of the well-known impossibility of consensus with lossy links [28,34] provided in Lemma 16, to prove that ensuring at most $k$ different decision values is impossible here, as too many decision values may originate from the unstable period.

**Lemma 16.** *Let $\mathcal{M}' = \langle p_i, p_j \rangle$ be a two-processor subsystem of our system $\mathcal{M} = \langle \Pi \rangle$. If the sequence of communication graphs $\mathcal{G}^r$, $r > 0$, of $\mathcal{M}$ are restricted by the existence of a round $r' > 0$ such that (i) for $r < r'$, $(p_i \to p_j) \in \mathcal{G}^r$ and/or $(p_j \to p_i) \in \mathcal{G}^r$, and no other edges incident with $p_i$ or $p_j$ are in $\mathcal{G}^r$, and (ii) for $r \geqslant r'$, there are no edges incident with $p_i$ and $p_j$ at all in $\mathcal{G}^r$, then consensus is impossible in $\mathcal{M}'$.*

**Proof.** Up to $r'$, this is ensured by the impossibility of 2-processor consensus with a lossy but at least unidirectional link established in [34, Lemma 3]. After $r'$, this result continues to hold (and is even ensured by the classic lossy link impossibility [28]). Hence, consensus is indeed impossible in $\mathcal{M}'$. □

**Theorem 8.** *There is no algorithm that solves $k$-set agreement for $n \geqslant k+1$ processes under the message adversary $\mathrm{VSSC}_{D,H}(k, \infty)$, for every $1 < k < n$.*

**Proof.** Suppose again that there is a $k$-set algorithm $\mathcal{A}$ that works correctly under the assumptions of our theorem. We restrict our attention to runs of $\mathcal{M}_{\mathcal{A}}$ where, until $r_{ST}$, (i) the same set of $k-1$ source components $\{\mathcal{D}_1, \ldots, \mathcal{D}_{k-1}\}$ with $\mathcal{D} = \bigcup_{i=1}^{k-1} \mathcal{D}_i$ exists in every round, and (ii) two remaining processes $\overline{\mathcal{D}} = \Pi \setminus \mathcal{D} = \{p_1, p_2\}$ exist, which are (possibly only uni-directionally, i.e., via a lossy link) connected in every round, without additional edges to or from $\mathcal{D}$. After $r_{ST}$, the communication graph remains the same, except that the processes in $\overline{\mathcal{D}}$ are disconnected from each other and there is an edge from, say, $p_1$ to some process in $\mathcal{D}$ in every round. Note that these runs satisfy Definition 15 for $d = \infty$, as the number of source components never exceeds $k$.

Moreover, we let the adversary choose $r_{ST}$ sufficiently large such that the processes in $\mathcal{D}$ have decided. Since the processes in $\mathcal{D}_i$ ($i < 0 < k$) never receive a message from the remaining system before $r_{ST}$, in which case they must eventually unilaterally decide, we can safely assume this.

We can now again employ the generic impossibility Theorem 6 in this modified setting. The proofs of properties (A), (B) and (D) remain essentially the same as in Theorem 7. It hence only remains to prove:

(C) Consensus is impossible in $\mathcal{M}' = \left\langle \overline{\mathcal{D}} \right\rangle$: This follows immediately from Lemma 16 with $r' = r_{ST}$. □

The following Theorem 9 reveals that even (considerably) less than $k$ source components per round before stabilization and a single perpetually stable source component after stabilization are not sufficient for solving $k$-set agreement.

**Theorem 9.** *There is no algorithm that solves $k$-set agreement for $n \geqslant k+1$ processes under the message adversary $\mathrm{VSSC}_{D,H}(\lceil k/2 \rceil + 1, \infty)$, for every $1 < k < n$, even if $\mathcal{G}^r = \mathcal{G}$, $r \geqslant r_{ST}$, where $\mathcal{G}$ contains only a single source component.*

**Proof.** We show that, under the assumption that $\mathcal{A}$ exists, there is a sequence of communication graphs that is feasible for our message adversary that leads to a contradiction. We choose $x_i = i$ for all $p_i \in \Pi$ and let $\mathcal{D}_i = \{p_{1+2i}, p_{2+2i}\}$ for $0 \leqslant i < \lceil k/2 \rceil - 1$. If $k$ is even, let $\mathcal{D}_{k/2-1} = \{p_{k-1}, p_k\}$; if $k$ is odd, let $\mathcal{D}_{\lceil k/2-1 \rceil} = \{p_k\}$. In any case, let $\mathcal{D}_{\lceil k/2 \rceil} = \{p_{k+1}\}$. Finally, let $\overline{\mathcal{D}} = \{p_{k+2}, \ldots, p_n\}$. Note that $\overline{\mathcal{D}}$ may be empty, while all $\mathcal{D}_i$ are guaranteed to contain at least one process since $n > k$. For all rounds, the processes in $\overline{\mathcal{D}}$ have an incoming edge from a process in one of the $\mathcal{D}_i$.

We split the description of the adversarial strategy into $\lceil k/2 \rceil + 1$ phases in each of which we will force some $\mathcal{D}_i$ to take $|\mathcal{D}_i|$ decisions. To keep processes $p_{1+2i}, p_{2+2i} \in \mathcal{D}_i$ with $|\mathcal{D}_i| = 2$ from deciding on the same value before their respective phase $i$, the adversary restricts $\mathcal{G}^r$ such that (i) there are no links to $\mathcal{D}_i$ from any other $\mathcal{D}_j$ and (ii) either the

edge $(p_{1+2i} \to p_{2+2i})$ or $(p_{1+2i} \leftarrow p_{2+2i})$ or both are in $\mathcal{G}^r$, in a way that causes Lemma 16 to apply. Note carefully that any such $\mathcal{G}^r$ indeed has no more than $\lceil k/2 \rceil + 1$ source components.

In the initial phase, $\mathcal{D}_{\lceil k/2 \rceil}$ is forced to decide: Since $p_{k+1}$ has no incoming edges from another node in $\mathcal{G}^r$, this situation is indistinguishable from a run where $p_{k+1}$ became the single source component after $r_{ST}$. Thus, by the correctness of $\mathcal{A}$, $p_{k+1}$ must eventually decide on $x_{k+1} = k+1$. At this point, the initial phase ends, and we can safely allow the adversary to modify $\mathcal{G}^r$ in such a way that $p_{k+1}$ has an incoming edge from some other process.

We now proceed with $\lceil k/2 \rceil - 1$ phases: In the $i$th phase, $0 \leqslant i < \lceil k/2 \rceil - 1$, the adversary drops any link between the processes $p_{1+2i}, p_{2+2i} \in \mathcal{D}_i$ (and does not provide an incoming link from any other process, as before) in any $\mathcal{G}^r$. Since, for both processes this is again indistinguishable from the situation where they become the single source component after $r_{ST}$, both will eventually decide in some future round (if they have not already decided). Since the adversary may have chosen a link failure pattern in earlier phases that causes the impossibility (= forever bivalent run) of Lemma 16 to apply, as $\mathcal{M}'_{\mathcal{A}_{|\mathcal{D}_i}} \preccurlyeq_{\mathcal{D}_i} \mathcal{M}_{\mathcal{A}}$, it follows that $\mathcal{A}$ and hence $\mathcal{A}_{|\mathcal{D}_i}$ cannot have solved consensus in $\mathcal{D}_i$. Since $\mathcal{A}$ solves $k$-set agreement, $p_{1+2i}$ and $p_{2+2i}$ must hence decide on two *different* values. Moreover, since neither $p_{1+2i}$ nor $p_{2+2i}$ ever received a message from a process not in $\mathcal{D}_i$, their decision values must be different from the ones in all former phases.

Finally, after $p_{1+2i}$ and $p_{2+2i}$ have made their decisions, the adversary may again modify $\mathcal{G}^r$ such that they have an incoming edge from some other process, thereby reducing the number of source components by two and preserving the maximum number $\lceil k/2 \rceil + 1$ of source components, and continue with the next phase.

If $k$ is even, then the final phase $\lceil k/2 \rceil - 1$ forces two more decisions just as described above; otherwise, $p_k$ provides one additional decision value (which happens concurrently with the initial phase here). In either case, we have shown that all $p_i$ with $1 \leqslant i \leqslant k+1$ have decided on different values, which contradicts the assumption that a correct algorithm $\mathcal{A}$ exists. □

Note that Theorem 9 reveals an interesting gap between 2-set agreement and 1-set agreement, i.e., consensus: It shows that 2-set agreement is impossible with $\lceil k/2 \rceil + 1 = 2$ source components per round before and a single fixed source component after stabilization. By contrast, if we reduce the number of source components per round to a single one before stabilization (and still consider a single fixed source component thereafter), even 1-set agreement becomes solvable [35].

## 7. Algorithms for $k$-set agreement

In this section, we will provide a message adversary MAJINF($k$) (Definition 19) that is sufficiently weak for solving $k$-set agreement if combined with $\mathrm{VSSC}_{D,H}(n, 3D+H)$ (Definition 15). Although we can of course not claim that it is a strongest one in terms of problem solvability (we did not even define what this means), we have some intuitions that it is not too far from the solvability/impossibility border.

### 7.1. Set agreement

To illustrate some of the ideas that will be used in our message adversary for general $k$-set agreement, we start with the simple case of $n-1$-set agreement (also called *set agreement*) first. Note that Theorem 7 does not apply here. To circumvent the impossibility result of Theorem 9, it suffices to strengthen the assumption of at most $n-1$ source components in every round such that the generation of too many decision values during the unstable period is ruled out. A straightforward way to achieve this is to just forbid $n$ different decisions obtained in source components consisting of a single process. Achieving this is easy under the $\Sigma_{n-1}$-influence message adversary given in Definition 16, the name of which has been inspired by the $\Sigma_{n-1}$ failure detector [59].

**Definition 16** *($\Sigma_{n-1}$-influence message adversary).* The message adversary $\Sigma_{n-1}$-MAJ is the set of all sequences of communication graphs $(\mathcal{G}^r)_{r>0}$ that satisfy the following: if each process $p_i \in \Pi$ becomes a single-node source component during a non-empty set of Intervals $X_i$, then any selection $\{I_1, \ldots, I_n\}$ with $I_i \in X_i$ for $1 \leqslant i \leqslant n$, contains two distinct $I_i = [a, b]$ and $I_j = [a', b']$ such that $\mathrm{s}_i^b \rightsquigarrow \mathrm{s}_j^{a'}$.

It is easy to devise a set agreement algorithm that works correctly in a dynamic network under Definition 16, provided (a bound on) $n$ is known: In Algorithm 3, process $p_i$ maintains a proposal value $v_i$, initially $x_i$, and a decision value $y_i$, initially $\perp$, which are broadcast in every round. If $p_i$ receives no message from any other process in a round, it decides by setting $y_i = v_i$. If $p_i$ receives a message from some $p_j$ that has already decided ($y_j \neq \perp$), it sets $y_i = y_j$. Otherwise, it updates $v_i$ to the maximum of $v_i$ and all received values $v_j$. At the end of round $n$, a process that has not yet decided sets $y_i := v_i$, and all processes terminate.

**Theorem 10** *(Correctness Algorithm 3). Algorithm 3 solves $n-1$-set agreement in a dynamic network under message adversary $\Sigma_{n-1}$-MAJ given in Definition 16.*

**Proof.** Termination (after $n$ rounds) and also validity are obvious, so it only remains to show $n-1$-agreement. Assume, w.l.o.g., that the processes $p_1, p_2, \ldots$ are ordered according to their initial values $x_{p_1} \leqslant x_{p_2} \leqslant \ldots$, and let $R^k$ be the set of

**Algorithm 3** Set agreement algorithm for message adversary $\Sigma_{n-1}$-MAJ.

**Set agreement algorithm, code for process $p_i$:**
1: $v_i := x_i \in V$ // initial value
2: $y_i := \perp$
**Emit round $r$ messages:**
3: send $\langle v_i, y_i \rangle$ to all
**Receive round $r$ messages:**
4: receive $\langle v_j, y_j \rangle$ from all current neighbors
**Round $r$: computation:**
5: $v_i := \max\{v_i, v_j : p_j \in \mathcal{N}_i\}$
6: **if** $\exists j : (y_j \neq \perp) \wedge (y_i = \perp)$ **then**
7: $\quad y_i := y_j$
8: **if** $(\mathcal{N}_i = \emptyset) \wedge (y_i = \perp)$ **then**
9: $\quad y_i := v_i$
10: **if** $(r = n) \wedge (y_i = \perp)$ **then**
11: $\quad y_i := v_i$; terminate

different values (in $y_i$ or, if still $y_i = \perp$, in $v_i$) present in the system at the beginning of round $k \geqslant 1$; $R^1$ is the set of initial values. Obviously, $R^1 \supseteq R^2 \supseteq \ldots$, and since $n-1$-agreement is fulfilled if $|R^{n+1}| < n$, we only need to consider the case where all $x_i$ are different.

Consider process $p_1$: If $p_1$ gets a message from some other process $p_j$ in round 1, $x_1 \notin R^2$ as (i) $p_1$ does not decide on its own value and sets $v_1 \geqslant v_j \geqslant x_j > x_1$ and (ii) no process that receives a message containing $x_1$ from $p_1$ takes on this value. Hence, $n-1$-set agreement will be achieved in this case. Otherwise, $p_1$ does not get any message in round 1 and hence decides on $x_1$.

Proceeding inductively, assume that $p_\ell \in P^{i-1} = \{p_1, \ldots, p_{i-1}\}$ has decided on $x_\ell$ by round $k \leqslant \ell$, and received only messages from processes with smaller index in rounds $1, \ldots, k-1$ and no message in round $k$. Now consider process $p_i$: If $p_i$ gets a message from some process $p_j$ with $j > i$ in some round $k \leqslant i$, with minimal $k$, before it decides, then $x_i \notin R^{k+1}$ as (i) $p_i$ does not decide on its own value and sets $v_i \geqslant v_j \geqslant x_j > x_i$, (ii) $p_i$ did not send its value to any process in $P^{i-1}$ before their decisions, and (iii) no process with index larger than $i$ that receives a message containing $x_i$ from $p_i$ takes on this value. Hence, $n-1$-set agreement will be achieved in this case. Otherwise, if $p_i$ gets a message from some process $p_\ell \in P^{i-1}$ in round $i$, it will decide on $p_\ell$'s decision value $x_\ell$ and hence also cause $x_i \notin R^{i+1}$. In the only remaining case, $p_i$ does not get any message in round $i$ and hence decides on $x_i$, which completes the inductive construction of $P^i = \{p_1, \ldots, p_i\}$ for $i < n$.

Now consider $p_n$ in round $n$ in the above construction of $P^n$: Definition 16 prohibits the only case where $n-1$-agreement could possibly be violated, namely, when $p_n$ also decides on $x_n$: During the first $n$ rounds, we would have obtained $n$ single-node source components no two of which influence each other in this case. Thus, we cannot extend the inductive construction of $P^i$ to $i = n$, as the resulting execution would be infeasible. □

### 7.2. A message adversary for general k-set agreement

Whereas the set agreement solution introduced in the previous subsection is simple, it is apparent that Definition 16 is quite demanding. In particular, it requires explicit knowledge of (a bound on) $n$. We will now provide a message adversary MAJINF($k$) (Definition 19), which is sufficient for general $k$-set agreement if combined with $\mathrm{VSSC}_{D,H}(k, 3D+H)$ (Definition 15) and even with $\mathrm{VSSC}_{D,H}(n, 3D+H)$. We obtained this combination by adding some additional properties to the necessary network conditions implied by our impossibility Theorems 7 and 9.[7]

To avoid non-terminating (i.e., forever undecided) executions as predicted by Theorem 7, we require the *stable interval* constraint guaranteed by the message adversary $\mathrm{VSSC}_{D,H}(n, 3D+H)$ to hold. The parameter $D$, which can always be safely set to $D = n-1$ according to Lemma 1, allows to adapt the message adversary to the actual dynamic source diameter guaranteed in the VSSCs of a given dynamic network. Note that, since $D > 0$, rounds where no message is received are not forbidden here (in contrast to Definition 16).

In order to also circumvent executions violating the $k$-agreement property established by Theorem 9, we introduce the *majority influence* constraint guaranteed by the message adversary MAJINF($k$) given in Definition 19 below. Like Definition 16 for set agreement, it guarantees some (minimal) information flow between sufficiently long-lasting vertex-stable source components that exist at different times. As visualized in Fig. 3, it implies that the information available in any such $I$-VSSC, with $|I| > 2D$, originates in at most $k$ "initial" $J$-VSSCs, where $|J| > 2D$. Thereby, it enhances the very limited information propagation that could occur in our model solely under $\mathrm{VSSC}_{D,H}(k, 3D+H)$, which is too strong for solving $k$-agreement.

[7] An alternative way to derive sufficient network assumptions for, e.g., $n-2$-set agreement could be to generalize Definition 16: One could e.g. assume that at least two out of every set of $n-1$ different source components consisting of 1 or 2 processes are influenced by a common predecessor source component. Whereas this assumption does not require vertex stability of source components, it effectively ensures that information propagates not slower as in VSSCs. Owing to this fact, it also prohibits the existence of the node $p_j$ in Definition 14, resp. Definition 13, for which each $p_i$ in the source component fails to influence $p_j$ by round $x+E-2$, thereby causing the proof of Theorem 7 to fail. Working out the details may turn out difficult, though: After all, unlike single-process source components, larger source components suffer from the problem that its members cannot always determine whether the source component was a VSSC or not. Influence must hence be conservative, in the sense that it involves even *potential* 2-process source components.

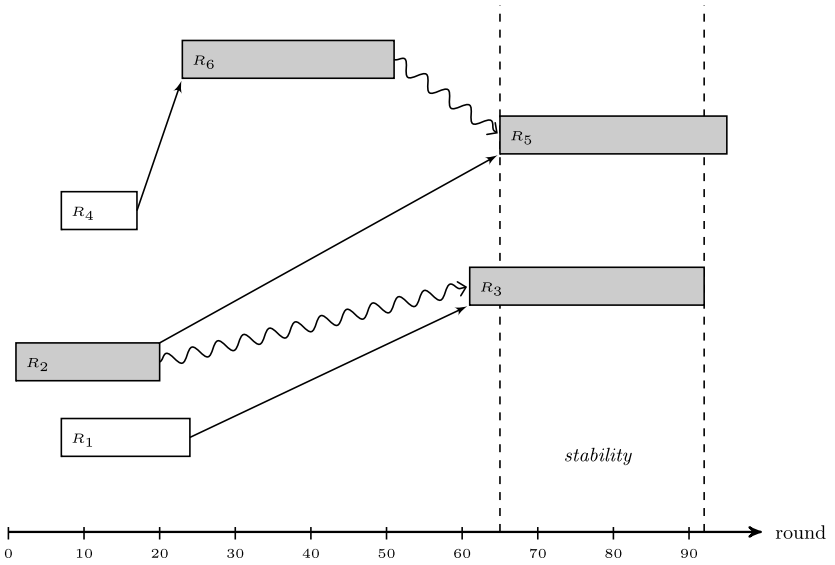



**Fig. 3.** VSSCs influencing each other in a run, for $k = 2$. Time progresses from left to right; all grey rectangles are stable for more than $2D$ rounds, white rectangles are stable between $D+1$ and $2D$ rounds. Snaked arrows represent majority influence, thin arrows represent (weak) influence. At most two grey rectangles may exist that are not majority-influenced by any other grey rectangles.

Formally, given some run $\rho$, we denote by $\mathbb{V}_d$ the set of all pairs $(S, I)$ where $S$ is an $I$-vertex-stable-source components with $|I| \geqslant d$ in $\rho$; note that $\mathbb{V}_d \subseteq \mathbb{V}_1$ for every $d \geqslant 1$.

**Definition 17** *(Causal influence sets).* Let $(S, I) \in \mathbb{V}_1$ with $I = [a, b]$, $(S', I') \in \mathbb{V}_1$ with $I' = [a', b']$, and let $a' > b$. The causal influence set of $(S, I)$ and $(S', I')$ is $\mathsf{CS}((S, I), (S', I')) := \left\{ p_j \in S' \mid \exists p_i \in S \colon \mathrm{s}_i^b \rightsquigarrow \mathrm{s}_j^{a'} \right\}$.

The *majority influence* between $S$ and $S'$ guarantees that $S$ influences a set of nodes in $S'$, which is greater than any set influenced by VSSCs not already known by the processes in $S$ (and greater than or equal to any set influenced by VSSCs already known by the processes in $S$). Majority influence is hence a very natural way to discriminate between strong and weak influence between VSSCs.

**Definition 18** *(Majority influence).* Let $I = [a, b]$ and $I' = [a', b']$. For $(S, I) \in \mathbb{V}_{2D+1}$ and $(S', I') \in \mathbb{V}_{2D+1}$, we say that $(S, I)$ exercises a *majority influence* on $(S', I')$, denoted $(S, I) \hookrightarrow_{\mathtt{m}} (S', I')$ with $\hookrightarrow_{\mathtt{m}} \subseteq \mathbb{V}_{2D+1}^2$, if and only if for all $(S'', I'') \in \mathbb{V}_{D+1}$ with $\mathsf{CS}((S'', I''), (S, I)) = \emptyset$ it holds that $|\mathsf{CS}((S, I), (S', I'))| > |\mathsf{CS}((S'', I''), (S', I'))|$ and for all $(S'', I'') \in \mathbb{V}_{D+1}$ with $\mathsf{CS}((S'', I''), (S, I)) \neq \emptyset$ it holds that $|\mathsf{CS}((S, I), (S', I'))| \geqslant |\mathsf{CS}((S'', I''), (S', I'))|$.

The relation $\hookrightarrow_{\mathtt{m}}$ has the following properties:

**Lemma 17** *(Properties $\hookrightarrow_{\mathtt{m}}$). The majority influence relation is antisymmetric, acyclic and intransitive.*

**Proof.** Let $S$, $S'$, and $S''$ be three different VSSCs stable in the intervals $I = [a, b]$, $I' = [a', b']$, and $I'' = [a'', b'']$, respectively. W.l.o.g. assume $(S, I) \hookrightarrow_{\mathtt{m}} (S', I')$. This implies that $b < a'$ by the influence definition. Hence $a < b'$ which implies that $(S', I') \hookrightarrow_{\mathtt{m}} (S, I)$ does not hold. This proves that majority influence is antisymmetric and, by a transitive application of this argument, acyclicity. To prove intransitivity, observe that $(S, I) \hookrightarrow_{\mathtt{m}} (S', I')$ and $(S', I') \hookrightarrow_{\mathtt{m}} (S'', I'')$ would imply $\mathsf{CS}((S, I), (S'', I'')) > \mathsf{CS}((S', I'), (S'', I''))$ if $(S, I) \hookrightarrow_{\mathtt{m}} (S'', I'')$ also held, since no process in $S$ can be influenced by any process in $S'$. This contradicts $\mathsf{CS}((S', I'), (S'', I'')) \geqslant \mathsf{CS}((S, I), (S'', I''))$ required by $(S', I') \hookrightarrow_{\mathtt{m}} (S'', I'')$, however. □

With these preparations, we are now ready to specify a message adversary MAJINF($k$) given in Definition 19.

**Definition 19** *($k$-majority influence message adversary).* The message adversary MAJINF($k$) is the set of all sequences of communication graphs $(\mathcal{G}^r)_{r>0}$, where in every run there is a set $K \subseteq \mathbb{V}_{2D+1}$ with $|K| \leqslant k$ s.t. $\forall (S, I) \in \mathbb{V}_{2D+1} \setminus K \ \exists (S', I') \in \mathbb{V}_{2D+1}$ with $(S', I') \hookrightarrow_{\mathtt{m}} (S, I)$.

Informally, Definition 19 ensures that all but at most $k$ "initial" $I$-VSSCs with $|I| \geqslant 2D + 1$ are majority-influenced by some earlier $I'$-VSSCs with $|I'| \geqslant 2D + 1$ (see Fig. 3). Note carefully, though, that Definition 19 neither prohibits more than $k$ "initial" $I$-VSSCs with $|I| \leqslant 2D$ nor the partitioning of the system in more than $k$ simultaneous VSSCs.

We conclude this section with some straightforward stronger assumptions, which also imply Definition 19 and can hence be handled by the algorithm introduced in Section 7.3:

(i) Replacing majority influence in Definition 18 by majority intersection $|S \cap S'| > |S'' \cap S'|$, which is obviously the strongest form of influence.
(ii) Requiring $|S \cap S'| > |S'|/2$, i.e., a majority intersection with respect to the number of processes in $S'$. This could be interpreted as a changing VSSC, in the sense of "$S'$ is the result of changing a minority of processes in $S$". Although this restricts the rate of growth of VSSCs in a run, it would apply, for example, in case of random graphs where the giant component has formed [60,61].

### 7.3. Gracefully degrading consensus/k-set agreement

In this section, we provide a $k$-set agreement algorithm and prove that it works correctly under the message adversary $\mathrm{VSSC}_{D,H}(n, 3D + H) + \mathrm{MAJINF}(k)$, i.e., the conjunction of Definitions 15 and 19. Note that the algorithm needs to know $D$, but neither $n$ nor $H$. It consists of a "generic" $k$-set agreement algorithm, which relies on the network approximation algorithm of Section 5.1 for locally detecting vertex-stable source components and a function `GetLock` that extracts candidate decision values from history information. Our implementation of `GetLock` uses a vector-clock-like mechanism for maintaining "causally consistent" history information, which can be guaranteed to lead to proper candidate values thanks to $\mathrm{VSSC}_{D,H}(n, 3D + H) + \mathrm{MAJINF}(k)$.

In sharp contrast to classic $k$-set agreement algorithms, the algorithm is *$k$-universal*, i.e., the parameter $k$ does not appear in its code. Rather, the number of system-wide decision values is determined by the number of (certain) $2D + 1$-VSSCs occurring in the particular run. As a consequence, if the network partitions into $k$ weakly connected components, for example,[8] all processes in a component obtain the same decision value. On the other hand, if the network remains well-connected, the algorithm guarantees a unique decision value system-wide.

**Properties.** Our algorithm is in fact not only $k$-universal but even worst-case $k$-optimal, in the sense that (i) it provides at most $k$ decisions system-wide in all runs that are feasible for $\mathrm{VSSC}_{D,H}(n, 3D + H) + \mathrm{MAJINF}(k)$, and (ii) that there is at least one feasible run under $\mathrm{VSSC}_{D,H}(n, 3D + H) + \mathrm{MAJINF}(k)$ where no correct $k$-set agreement can guarantee less than $k$ decisions. (i) will be proved in Section 7.4, and (ii) follows immediately from the fact that a run consisting of $k$ isolated partitions is also feasible for $\mathrm{VSSC}_{D,H}(n, 3D + H) + \mathrm{MAJINF}(k)$. Our algorithm can hence indeed be viewed as a consensus algorithm that degrades gracefully to $k$-set agreement, for some $k$ determined by the actual network properties.

**Network approximation.** Like the consensus algorithm in Section 5, our $k$-set agreement algorithm consists of two reasonably independent parts, the network approximation algorithm Algorithm 1 and the $k$-set agreement core algorithm given in Algorithm 4. As in Section 5.2, we assume that the complete round $r$ computing step of the network approximation algorithm is executed just before the round $r$ computing step of the $k$-set algorithm, and that the round $r$ message of the former is piggybacked on the round $r$ message of the latter. Recall that this implies that the round $r$ computing step of the $k$-set core algorithm, which terminates round $r$, can already access the result of the round $r$ computation of the network approximation algorithm, i.e., its state at the end of round $r$.

**Core algorithm.** The general idea of our core $k$-set agreement algorithm in Algorithm 4 is to generate new decision values only at members of $2D + 1$-VSSCs, and to disseminate those values throughout the remaining network. Using the network approximation $A_i$, our algorithm causes process $p_i$ to make a transition from the initially *undecided* state to a *locked* state when it detects some minimal "stability of its surroundings", namely, its membership in some $D + 1$-VSSC $D$ rounds in the past (line 17). Note that the latency of $D$ rounds is inevitable here, since information propagation within a $D + 1$-VSSC may take up to $D$ rounds since it is $D$-bounded, as guaranteed by item (ii) in Definition 15. If process $p_i$, while in the locked state, observes some period of stability that is sufficient for locally inferring a consistent view among *all* VSSC members (which occurs when the $D + 1$-VSSC has actually extended to a $2D + 1$-VSSC), $p_i$ can safely make a transition to the *decided* state (line 24). The decision value is then broadcast in all subsequent rounds, and adopted by any not-yet decided process in the system that receives it later on (line 9). Note that $\mathrm{VSSC}_{D,H}(n, 3D + H)$ (Definition 15) guarantees that this will eventually happen.

Since locking is done optimistically, however, it may also happen that the $D + 1$-VSSC does not extend to a $2D + 1$-VSSC (or, even worse, is not recognized to have done so by some members) later on. In this case, $p_i$ makes a transition from the locked state back to the undecided state (line 22). Unfortunately, this possibility has severe consequences: Mechanisms are required that, despite possibly inconsistently perceived unsuccessful locks, ensure both (a) an *identical* decision value among all members of a $2D + 1$-VSSC who successfully detect this $2D + 1$-VSSC and thus reach the decided state, and (b) no more than $k$ different decision values originating from different $2D + 1$-VSSCs.

[8] It is important to note that the network properties required by our algorithm to reach $k$ decision values need *not* involve $k$ isolated partitions: Obviously, $k$ isolated partitions in the communication graph also imply $k$ source components, but $k$ source components do not imply a partitioning of the communication graph into $k$ weakly connected components — one process may still be connected to several components.

**Algorithm 4** $k$-universal $k$-set agreement algorithm, code for process $p_i$.

```
   Variables and Initialization:
1:  hist_i[*][*] := ∅ /* hist_i[j][r] holds p_i's estimate of the locks learned by p_j in round r */
2:  hist_i[i][0] := {({p_i}, x_i, 0)} /* virtual first lock (V(S) := {p_i}, v := x_i, τ_create := 0) at p_i */
3:  ℓ := ⊥ // most recent lock round, ⊥ if none
4:  decision_i := ⊥ // p_i's decision, ⊥ if undecided
   Emit round r messages:
5:  send ⟨ hist_i, decision_i ⟩ to all neighbors
   Receive round r messages:
6:  for all p_j in p_i's neighborhood N_i^r, receive ⟨ hist_j, decision_j ⟩
   Round r computation:
7:  if decision_i = ⊥ then
8:    if received any message m containing m.decision ≠ ⊥ then
9:      decide m.decision and set decision_i := m.decision
10:   else
        // update hist_i with hist_j received from neighbors
11:     for p_j ∈ N_i^r, where p_j sent hist_j do
12:       hist'_i := hist_i // remember current history
13:       for all non-empty entries hist_j[x][r'] of hist_j, x ≠ i do
14:         hist_i[x][r'] := hist_i[x][r'] ∪ hist_j[x][r']
          // locally add all newly learned locks:
15:       hist_i[i] := hist_i \ hist'_i
        // perform state transitions (undecided, locked, decided):
16:     mySource := InStableSource([r − 2D, r − D])
17:     if ℓ = ⊥ and mySource ≠ ∅ then
18:       ℓ := r − 2D
19:       lock := GetLock(mySource, ℓ)
20:       hist_i[i][r] := hist_i[i][r] ∪ lock // create new lock
21:     else if ℓ ≠ ⊥ and mySource = ∅ then
22:       ℓ := ⊥ // release unsuccessful lock
23:     else if ℓ ≠ ⊥ and InStableSource([ℓ, ℓ + 2D]) ≠ ∅ then
24:       decide lock.v and set decision_i := lock.v

25: function GetLock(S, r')
26:   Let R be the multiset ⋃_{p_j∈R, r''≤r'} hist_i[j][r'']
      Let mfrq(R) be the set of the most frequent elements in R
27:   Let mfrq_latest(R) := {x ∈ mfrq(R) | ∀y ≠ x ∈ mfrq(R): x.τ_create > y.τ_create}
28:   if |mfrq_latest(R)| = 1 then
29:     Let v be s.v of the single element s ∈ mfrq_latest(R)
30:     newLock := (R, v, r)
31:   else
32:     newLock := (R, max_{s∈R}{s.v}, r) // deterministic choice
33:   return newLock
```

Both goals are accomplished by a particular selection of the decision values (using function `GetLock`), which ultimately relies on an intricate utilization the network properties guaranteed by our message adversary $\text{VSSC}_{D,H}(n, 3D + H) + \text{MAJINF}(k)$ (Definitions 15 and 19): Our algorithm uses a suitable *lock history* data structure for this purpose, which is continuously exchanged and updated among all reachable processes. It is used to store sets of *locks* $L = (S, v, \tau_{\texttt{create}})$, which are created by every process that enters the locked state: $S$ is the vertex-set of the detected $D + 1$-VSSC, $v$ is a certain proposal value (determined as explained below), and $\tau_{\texttt{create}}$ is the round when the lock is created.

**Maintaining history.** In more detail, the lock history at process $p_i$ consists of an array $\text{hist}_i[j][r]$ that holds $p_i$'s (under)approximation of the locks process $p_j$ got to know in round $r$. It is maintained using the following simple update rules:

(i) *Local lock creation:* Apart from the single *virtual* lock $(\{p_i\}, x_i, 0)$ created initially by $p_i$ in line 2 (which guarantees a non-empty lock history right from the beginning), all regular locks created upon $p_i$'s transition from the undecided to the locked state are computed by the function `GetLock` in line 19. Any lock locally created at $p_i$ in round $r$ (that is, in the round $r$ computing step of the core $k$-set agreement algorithm that terminates round $r$) is of course put into $\text{hist}_i[i][r]$.

(ii) *Remote lock learning:* Since all processes exchange their lock histories, $p_i$ may learn about some lock $L$ created by process $p_x$ in round $r'$ from the lock history $\text{hist}_j[x][r']$ received from some $p_j$ later on. In this case, $L$ is just added to $\text{hist}_i[x][r']$ (line 14).

(iii) *Local lock learning:* In order to ensure that the lock histories of all members of a $2D + 1$-VSSC are eventually consistent, which will ensure identical decision values, *every* newly learned remote lock $L \in \text{hist}_i[x][r']$ obtained in (ii) is also added to $\text{hist}_i[i][r]$.

Note that the update rules (i)+(ii) resemble the ones of vector clocks [62].

Clearly, $\text{hist}_i[i][r']$ will always be accurate for current and past rounds $r' \leqslant r$, while $\text{hist}_i[j][r']$ may not always be up-to date, i.e., may lack some locks that are present in $\text{hist}_j[j][r']$. Nevertheless, if $p_i$ and $p_j$ are members of the same $I$-VSSC $S$ with $I = [r - 2D, r]$, Definition 8 ensures that $p_i$ and $p_j$ have consistent histories $\text{hist}_i[j][r']$ and $\text{hist}_j[i][r']$ at latest by (the end of) round $r' + D$, for any $r' \in [r - 2D, r - D]$. Hence, if $p_i$ creates a new lock $L$ when it detects, in its round

$r$ computing step, that it was part of a $D+1$-VSSC that was stable from $r-2D$ to $r-D$, it is ascertained that any other member $p_j$ will have locally learned the same lock $L$ in the same round $r$, provided that the $D+1$-VSSC in fact extended to a $2D+1$-VSSC.

**Consistent decisions.** The resulting consistency of the histories is finally exploited by the function $\texttt{GetLock}(S, \ell)$, which computes (the value of) a new local lock (line 19) created in round $r$. As its input parameters, it is provided with the members $S$ of the detected $D+1$-VSSC and its starting round $\ell = r-2D$. `GetLock` first determines a multiset $R$, which contains all locks locally known to the members $p_j \in S$ by round $r-2D$ (line 26). Note that the multiplicity of some lock $L = (S', v, r')$ in $R$ is just the number of members of $S$ who got to know $L$ by round $r-2D$, which is just $|\mathrm{CS}(S', S)|$ according to Definition 17. In order to determine a proper value for the new lock to be computed by `GetLock`, we exploit the fact that MAJINF($k$) (given in Definition 19) ensures majority influence according to Definition 18: If the set $\mathrm{mfrq}_{\texttt{latest}}(R)$, containing the most frequent locks in $R$ with the same maximal lock creation round, contains a single lock $L$ only, its value $L.v$ is used. Note that the restriction to the maximal lock creation date automatically filters unwanted, outdated locks that have merely been disseminated in preceding $2D+1$-VSSCs, see (1) below. Otherwise, i.e., if $\mathrm{mfrq}_{\texttt{latest}}(R)$ contains multiple candidate locks, a consistent deterministic choice, namely, the maximum among all lock values in $R$, is used (line 32). As a consequence, at most $k$ different decision values will be generated system-wide.

Given the various mechanisms employed in our algorithm and their complex interplay, the question about a more light-weight alternative solution that omits some of these mechanisms might arise. We will proceed with some informal arguments that support the necessity of some of the pillars of our solution, namely, (1) the preference of most recently created locks in `GetLock`, (2) the creation of a new lock at every transition to the locked state, and finally (3) the usage of an a priori unbounded data structure $\mathrm{hist}_i$. Although these arguments are also "embedded" in the correctness proof in the following section, they do not immediately leap to the eye and are hence provided explicitly here.

(1) The preference of most recently created lock in `GetLock`, which is done by selecting the set $\mathrm{mfrq}_{\texttt{latest}}(R)$ in line 28, defeats the inevitable "amplification" of the number of processes that got to know some "old" lock: All members of a $2D+1$-VSSC have finally learned *all* "old" locks that were only known to *some* of its members at the starting round of the VSSC initially. In terms of multiplicity in $R$, this would falsely make any such old lock a preferable alternative to the most recently created lock.

(2) Instead of creating new locks at every newly detected $D+1$-VSSC, it might seem sufficient to simply update the creation time of an old lock that (dominantly) influences a newly detected VSSC. This is not the case, however: Consider a hypothesized algorithm where new locks are only generated if no suitable old locks can be found in the current history, and assume a run where two VSSCs with vertex sets $S_1 = \{p_1, p_2\}$ and $S_3 = \{p_1, p_2\}$ that are both stable for $D+1$ rounds and two source components $S_2 = \{p_1, p_3\}$ and $S_4 = \{p_1, p_3\}$ that are stable for $2D+1$ rounds are formed. Let these VSSCs be such that $S_i$ is formed before $S_j$ if $i < j$ and let there be no influence among the processes of $\{p_1, p_2, p_3\}$, apart from their influence on each other when they are members of the same VSSC. First, let the processes of $S_1$ lock on some old lock $L'$. Then, assume that the processes of $S_2$ lock on some lock[9] $L \neq L'$, a lock not known in $S_1$. Since $S_3 = \{p_1, p_2\}$, if $S_3$ is sufficiently well connected, $p_1$ might lock on $L'$ in $S_3$, because $L'$ is known to both $p_1$ and $p_2$ while $L$ is known merely to $p_1$ at the start of $S_3$. Subsequently, this results in the situation in $S_4$ where there is neither a clear majority ($L'$ and $L$ are known to both members of $S_4$) nor a clear most recently adopted lock (for $p_1$, it seems that $L'$ is the most recent lock, while for $p_3$, it seems that $L$ is more recent). Consequently, in $S_4$, it is not clear whether to lock on $L.v$ or on $L'.v$. Nevertheless, the processes of $S_4$ should be able to determine that they must lock on $L$ and not on $L'$, since $S_2 \hookrightarrow_{\mathrm{m}} S_4$ holds in our example: $|\mathrm{CS}(S_1, S_2)| = 1$, $|\mathrm{CS}(S_1, S_4)| = 2$, $|\mathrm{CS}(S_2, S_4)| = 2$ and $|\mathrm{CS}(S_3, S_4)| = 1$. We can therefore conclude that merely adopting old locks is insufficient.

(3) Since the stabilization round $r_{ST}$, as implied by Definition 15, may be delayed arbitrarily, an unbounded number of $2D+1$-VSSCs can occur before $r_{ST}$. Since any of those might produce a critical lock, in the sense of exercising a majority influence upon some later $2D+1$-VSSC, no such lock can safely be deleted from $\mathrm{hist}_i$ of any $p_i$ after bounded time.

### 7.4. Correctness proof

In this final subsection, we will prove the following Theorem 11:

**Theorem 11.** *Algorithm 4 solves $k$-universal $k$-set agreement in a dynamic network under the message adversary $\mathrm{VSSC}_{D,H}(n, 3D+H)$ + MAJINF($k$), which is the conjunction of Definition 15 and Definition 19.*

The proof consists of a sequence of technical lemmas, which will allow us to establish all the properties of $k$-set agreement given in Section 3. First, validity according to Definition 4 is straightforward to see, as only the values of locks are ever considered as decisions (line 24). Values of locks, on the other hand, are initialized to the initial value of a process (line 2)

[9] This could occur, e.g., because $L$ is known to $p_3$ and has a more recent creation time than $L'$.

and later on always have values of previous locks assigned to them (lines 30 and 32). Note that the claimed $k$-universality is obvious, as the code of the algorithm does not involve $k$.

To establish termination, we start with Lemmas 18 to 20 that are related to setting locks at all members of vertex stable source components.

**Lemma 18.** *Apart from processes adopting a decision sent by another process, only processes part of a vertex stable source component with interval length greater than $D$ (resp. $2D$) lock (resp. decide).*

**Proof.** The if-statement in line 17 (resp. line 23) is evaluated to true only if `InStableSource` detects a stable member set $S$ in some interval $I$ of length $D+1$ (resp. of length $2D+1$) or larger, which implies by Corollary 2 that $S$ is indeed a $I$-VSSC with $|I| = D+1$ (resp. $|I| = 2D+1$). □

**Lemma 19.** *All processes part of a $I$-VSSC $S$ with $I = [a, b]$ and $|I| > 2D$, which did not start already before $a$, lock, i.e. set $\ell := a$, in round $a + 2D$.*

**Proof.** Because $S$ is $D$-bounded by Definition 15, Corollary 3 guarantees that `InStableSource`$(a, a+D)$ returns $S$ from round $a + 2D$ (of the $k$-set-algorithm) on, and that it cannot have done so already in round $a + 2D - 1$. Hence, $\ell = \perp$ in round $a + 2D$, the if-statement in line 17 is entered and $\ell := a$ is set in line 19. □

**Lemma 20.** *All processes part of a $I$-VSSC $S$ with $I = [a, b]$ and $|I| > 3D$, which did not start already before $a$, have decided by round $a + 3D$.*

**Proof.** It follows from Lemma 19 that all members of the VSSC $S$ set $\ell := a$ in round $a+2D$. As the VSSC remains stable also in rounds $a + 2D, \ldots, a + 3D$, line 22 will not be executed in these rounds, thus $\ell = a$ remains unchanged. Consequently, due to Corollary 3, the if-statement in line 23 will evaluate to true at the latest in round $\ell + 3D = a + 3D$, causing all the processes to decide via line 24 by round $a + 3D$ as asserted. □

**Lemma 21.** *The algorithm eventually terminates at all processes.*

**Proof.** Pick any process $p_j$. If $p_j$ is part of a source component during the stable interval, guaranteed by Definition 15, Lemma 20 ensures termination by $r_{ST} + 3D$ at the latest. Thus, we assume $p_j$ is not part of a source component during the stable interval. From Definition 10, it follows that there exists a causal chain of length at most $H$ to $p_j$ from some member $p_i$ of a VSSC after its termination. Therefore, it must receive the decide message and decide via line 9 by $r_{ST} + 3D + H$ at latest. □

Although we now know that all members of a VSSC that is vertex stable for at least $3D$ rounds will decide, we did not prove anything about their decision values yet. In the sequel, we will prove that they decide on the *same* value.

**Lemma 22.** *Given some $I$-VSSC $S$ with $I = [a, b]$ and $b \geqslant a + D$, in all rounds $x \in [a + D, b]$ it holds that $\forall p_i, p_j \in S$: $\bigcup_{r' \leqslant a} hist_i[j][r'] = \bigcup_{r' \leqslant a} hist_j[j][r']$*

**Proof.** Because $S$ is $D$-bounded, a message from round $a$ has reached every member of $S$ by round $a + D$. Moreover, no message sent by a process not in $S$ during $I$ can reach a member of $S$ during $I$ because $S$ is a source component. Therefore, since $\text{hist}_i$ is sent by each process $p_i$ in every round (line 5) and $p_i$ adds only newly learned entries to $\text{hist}_i$ (lines 15 and 20), all these updates of $\text{hist}_i$ during $I$, regarding any round $r' \leqslant a$, occur at the latest in round $a + D$. □

**Lemma 23.** *All processes of a $I$-VSSC $S$ of $\mathbb{V}_{2D+1}$ with $I = [a, b]$ adopt the same lock (and hence decide the same).*

**Proof.** Such a lock is created by $p_i \in S$ in round $a+2D$, when it recognizes $S$ as having been vertex-stable for $D+1$ rounds according to Lemma 19. As the lock (value) is computed based on $\text{hist}_i$ present in round $a+2D$, which is consistent among all VSSC members by Lemma 22, the lemma follows. □

Finally, we show that, given that the system satisfies Definition 19, there will be at most $k$ decision values in any run of Algorithm 4, which proves $k$-agreement: Since there are at most $k$ VSSCs of $\mathbb{V}_{2D+1}$ that are not majority-influenced by other VSSCs, it remains to show that any majority-influenced VSSC decides the same as the VSSC it is majority-influenced by. In order to do so, we will first establish a key property of our central data structure $\text{hist}_i$.

**Lemma 24.** *Given $I$-VSSC $S$ with, $I = [a, b]$, and $I'$-VSSC $R'$, $I' = [a', b']$, where $|I| > 2D$ and $|I'| \geqslant 1$, let $L$ be a lock known to all members of $S$ by $b$, i.e., for all $p_i \in S$ it holds that, by the end of round $b$, $L \in \bigcup_{r' \leqslant b} hist_i[i][r']$. For any process $p_j \in S'$, it holds that if there exists some $p_i \in S$, s.t. $s_i^b \rightsquigarrow s_j^{a'}$, then $L \in \bigcup_{r' \leqslant a'} hist_j[j][r']$.*

**Proof.** Assume there exists a $p_i \in S$ s.t. $s_i^b \rightsquigarrow s_j^{a'}$ but $L \notin \bigcup_{r' \leqslant a'} \text{hist}_j[j][r']$. The definition of $s_i^b \rightsquigarrow s_j^{a'}$ implies that there exists a causal chain from $p_i$ to $p_j$ that ends before $p_j$ becomes a part of $S'$. Since processes send their own history in every round according to line 5, every message in this causal chain consisted of a hist containing $L$ and thus $p_j$ put $L$ into its $\text{hist}_j[j][r]$ via line 14 if $\bigcup_{r' \leqslant r} \text{hist}_j[j][r']$ did not already contain $L$. □

**Lemma 25.** *Given 1-VSSC $S \in \mathbb{V}_{2D+1}$, $I = [a, b]$, and $I'$-VSSC $S' \in \mathbb{V}_{2D+1}$, $I' = [a', b']$, assume that the processes of $S$ created the (same) lock $L$ when locking. If $(S, I) \hookrightarrow_{\mathtt{m}} (S', I')$, then the processes in $S'$ will choose a lock $L'$ where $L.v = L'.v$ (and hence decide the same as the processes in $S$).*

**Proof.** From the definition of $\hookrightarrow_{\mathtt{m}}$ (Definition 18), it follows that no $I''$-VSSC of $\mathbb{V}_{D+1}$ has a larger influence set on $S'$ than $S$. By Lemma 18, this implies that no lock that was generated by some $I''$-VSSC in $\mathbb{V}_{D+1}$ can be known to more members of $S'$ than the lock $L$ generated by $S$. Since process $p_i$ puts only newly learned locks into $\text{hist}_i$ (lines 15 and 20), by Lemma 24, this means that in round $a'$ no "bad" lock $L_b$ is present in more elements of $R = \bigcup_{p_i \in R_{l'}, r' \leqslant a'} \text{hist}_i[i][r']$ than $L$. We now show that $L.\tau_{\mathtt{create}} > L_b.\tau_{\mathtt{create}}$ for all $L_b$ occurring in as many elements of $R$ as $L$ with $L_b \neq L$. Obviously, the only locks $L_b$ that could occur in as many elements of $R$ as $L$ are locks that have been in $\text{hist}_i$ of some $p_i \in S$ at the beginning of round $a$ already. Since for any such $L_b$, $L$ was created after $L_b$, by lines 30 and 32, we have that $L.\tau_{\mathtt{create}} > L_b.\tau_{\mathtt{create}}$, as claimed. Because in round $a' + 2D$, at all processes $p_i, p_j$ of $S'$, Lemma 22 implies that $\bigcup_{r' \leqslant a'} \text{hist}_i[j][r'] = \bigcup_{r' \leqslant a'} \text{hist}_j[i][r']$, when locking in round $a' + 2D$ according to Lemma 19, every $p_i$ of $S'$ will find $L$ as the unique most common lock in the elements of $R$ with maximal $\tau_{\mathtt{create}}$. This leads to the evaluation of the if-statement in line 28 to true and to the creation of a new lock $L'$, where $L'.v = L.v$ in line 30, as asserted. □

This completes the proof of Theorem 11.

## 8. Failure detectors

A convenient way to characterize consensus and $k$-set solvability in distributed systems where processes are (usually) subject to crash failures are *failure detectors* [63]. Well-known results for message passing systems where a majority of processes may crash are the weakest failure detector $(\Sigma, \Omega)$ (defined below) for consensus [64], and the necessary failure detector $\Sigma_k$ ($\Sigma = \Sigma_1$) for $k$-set agreement [59]. Note that, whereas the weakest failure detector for $k$-set agreement in message passing systems is still unknown, there are failure detectors like $\mathcal{L}_k$ [45] that are sufficiently strong for this purpose. These results imply that, from any solution that solves consensus resp. $k$-set agreement, it must be possible to implement $\Sigma$ and $\Omega$ resp. $\Sigma_k$. Conversely, if $\Sigma$ and $\Omega$ can be implemented in some system, then there are well-known algorithms for solving consensus in this system.

In this section, we follow the example of [23] and explore the relation between our message adversaries and the above failure detectors. It is important to note, though, that both $\text{VSSC}_{D,E}(d)$ and $\text{VSSC}_{D,H}(n, d) + \text{MAJINF}(k)$ are inherently incompatible with time-free failure detectors, as they involve explicit timing information, namely, the duration of the stability window. By contrast, the specifications of $\Omega$, $\Sigma$ and $\Sigma_k$ are time-free, in the sense that they only involve eventual properties for liveness. Therefore, we will consider only the eventually-forever variants $\text{VSSC}_{D,E}(\infty)$ and $\text{VSSC}_{D,H}(n, \infty) + \text{MAJINF}(k)$ of our message adversaries in the comparison below.

### 8.1. Failure detector basics

We recall that a crash failure means that a faulty process may stop to perform any computation step after some point during an execution, possibly in a way that causes only a subset of the processes to receive the message of the last broadcast. Given the time domain $\mathcal{T}$ of some system where processes are prone to crashes, for some given run, the function $F : \mathcal{T} \to 2^{\Pi}$ that maps each $t \in \mathcal{T}$ to the processes that are crashed by $t$ is called the failure pattern of the run. Processes in the set $\mathcal{C} = \Pi \setminus \bigcup_{t \in \mathcal{T}} F(t)$ are called *correct*.

In the case of a synchronous model with lock-step rounds, $\mathcal{T} = \mathbb{N}$. Let $\mathcal{AMP}_{n,x}$ denote the *asynchronous message passing model* where up to $x$ out of the overall $n = |\Pi|$ processes may crash; $x = n - 1$ characterizes the wait-free model. In $\mathcal{AMP}_{n,x}$, messages are delivered after a finite but unbounded time and processes do not operate in lock-step, hence $\mathcal{T} = \mathbb{R}$.

A failure detector is an oracle that can be queried by any process. Formally, a history $H$ with range $\mathcal{R}$ is a function $H : \Pi \times \mathcal{T} \to \mathcal{R}$. A failure detector $\mathcal{D}$ with range $\mathcal{R}$ maps a non-empty set of histories with range $\mathcal{R}$ to each failure pattern.

Two important failure detectors for consensus are $\Sigma$ and $\Omega$.

**Definition 20.** The eventual leader failure detector $\Omega$ has range $\Pi$. For each failure pattern $F$, for every history $H \in \Omega(F)$, there is a time $t \in \mathcal{T}$ and a correct process $p_j$ s.t. for every process $p_i$ for every $t' \geqslant t$, $H(p_i, t') = p_j$.

**Definition 21.** The quorum failure detector $\Sigma$ has range $2^{\Pi}$. For each failure pattern $F$, for every $H \in \Sigma(F)$, two properties hold: (1) for every $t, t' \in \mathcal{T}$ and $p_i, p_j \in \Pi$ we have $H(p_i, t) \cap H(p_j, t') \neq \emptyset$ and (2) there is a time $t \in \mathcal{T}$ s.t. for every process $p_i$, for every $t' \geqslant t$, $H(p_i, t') \subseteq \Pi \setminus \bigcup_{t \in \mathcal{T}} F(t)$.

We denote by $\mathcal{AMP}_{n,n-1}[fd:\mathcal{D}]$ the $\mathcal{AMP}_{n,n-1}$ model where processes have access to failure detector $\mathcal{D}$. We use $\mathcal{AMP}_{n,n-1} = \mathcal{AMP}_{n,n-1}[fd:\emptyset]$ to denote the absence of any failure detector. The combination $(\Sigma, \Omega)$ of $\Sigma$ and $\Omega$ is of particular importance, because it is the weakest failure detector for consensus [64] in $\mathcal{AMP}_{n,n-1}$, in the following sense: First, there is an algorithm that uses $(\Sigma, \Omega)$ for solving consensus. Second, let $\mathcal{D}$ be a failure detector s.t. consensus is solvable under $\mathcal{AMP}_{n,n-1}[fd:\mathcal{D}]$. Then, there is an algorithm $\mathcal{A}$ for $\mathcal{AMP}_{n,n-1}[fd:\mathcal{D}]$ that, for each failure pattern $F$, produces an output, denoted $out(p_i,t)$ for process $p_i$ at time $t$, s.t. setting $H(p_i,t) := out(p_i,t)$ defines a valid history $H$ of $(\Sigma, \Omega)$.

In order to relate such failure detector models to our message adversaries, which model dynamic link failures, 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 [23], 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.

Hence, we can define correct resp. faulty processes in our directed dynamic network model as follows:

**Definition 22.** Given an infinite sequence of communication graphs $\sigma$, process $p_i$ is *faulty* in a run with $\sigma$ if there is a round $r$ s.t., for some process $p_j$, for all $r' > r$: $\mathrm{s}_i^r \not\rightsquigarrow \mathrm{s}_j^{r'}$.

Let $\mathcal{C}(\sigma) = \left\{ p_i \in \Pi \mid \forall p_j \in \Pi, \forall r \in \mathbb{N}, \exists r' > r : \mathrm{s}_i^r \rightsquigarrow \mathrm{s}_j^{r'} \right\}$ denote the strongly correct (= non-faulty) processes in any run with $\sigma$.

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 23** *(Failure pattern).* The failure pattern associated with communication graph sequence $\sigma$ is a function $F_\sigma : \mathbb{N} \to 2^\Pi$ s.t. $p_i \in F_\sigma(r)$ if, and only if, for all processes $p_j \in \mathcal{C}(\sigma)$, for all $r' > r$: $\mathrm{s}_i^r \not\rightsquigarrow \mathrm{s}_j^{r'}$.

Hence, $F(r) \subseteq F(r+1)$ and, for any $\sigma$ of $\mathrm{VSSC}_{D,E}(\infty)$, $\mathcal{C}(\sigma) \neq \emptyset$ as the (infinitely vertex-stable) source component $S$ must satisfy $S = \mathcal{C}(\sigma) \neq \emptyset$.

We denote by $\mathcal{SMP}_n[adv:MA]$ the synchronous message passing model with $n$ processes where message loss is controlled by the adversary $MA$. In order to demonstrate how to relate this model to a crash failure model, we introduce the message adversary $CRASH(x)$, which guarantees that at least $n-x$ processes reach every other process infinitely often.

**Definition 24.** For $x < n$, we define $CRASH(x)$ as the set of those communication graph sequences $\sigma$ where $|\mathcal{C}(\sigma)| \geqslant n-x$.

Using a full-information protocol, we can transform a run of a synchronous model with the message adversary $CRASH(x)$ for $x < n/2$ to a run in asynchronous message passing with crashes:

**Corollary 4.** *For $x < n/2$, any run with graph sequence $\sigma$ of $\mathcal{SMP}_n[adv:CRASH(x)]$ can be transformed to a run in $\mathcal{AMP}_{n,x}[fd:\emptyset]$, which is indistinguishable for all simulated processes.*

**Proof.** Every process $p_i$ executes a simulator, which invokes the steps of the simulated process as follows: The simulator keeps track of all messages sent by the simulated process so far, and adds this history to every simulation message it sends. Consequently, any message sent by a strongly correct process in the run under $CRASH(x)$ is eventually delivered to all other processes. To ensure that this is also true for all the messages sent by not strongly correct processes, a process $p_i$ that has sent message $(m,i,j)$ to $p_j$ in its last simulated step is allowed to take its next simulated step only if $(m,i,j)$ is already known to (the simulator of) at least $n-x$ processes. If this never becomes true, the simulated $p_i$ does not execute further steps, i.e., is deliberately "crashed" by the simulation. Since $x < n/2$, there is always at least one strongly correct process among the $n-x$ processes that know $(m,i,j)$, which eventually disseminates this message to all processes in the system as needed.

Hence, it only remains to prove that the resulting simulation is consistent, i.e., that the simulated (non-atomic) send and receive operations are linearizable: Let $t_j$ be the time (round) when the simulated process $p_j$ is about to make the step where $(m,i,j)$ is processed, with $t_j = \infty$ if this is never the case (the simulator at $p_j$ never comes to know this message). Moreover, let $t_i'$ be the time when the simulated process $p_i$ is about to perform the next step after having sent $(m,i,j)$, with $t_i' = \infty$ if it never executes this next step (because it is "crashed"). Now, the send operation of $(m,i,j)$ is linearized to $t_{send} = \min_{p_j \in \Pi}\{t_j, t_i'\}$ (we assume here that $(m,i,.)$ is actually broadcast in the simulated process $p_i$'s computing step); if $t_{send} = \infty$, it is linearized to some arbitrary time $t_{send} = t_x$ after any (unsuccessful) receiver of $(m,i,.)$ (including $p_j$) that is not crashed by the simulation has performed its next step. The reception of $(m,i,j)$ is linearized at the time $t_j$ if $t_j < \infty$, or else at time $t_{send}$.

Since this linearization ensures the proper send-receive order, every run of this simulation in $\mathcal{SMP}_n[adv : CRASH(x)]$ is indeed indistinguishable for all processes from a run in $\mathcal{AMP}_{n,x}[fd : \emptyset]$. □

### 8.2. Consensus

We are now ready to explore the relation of our consensus message adversary $\text{VSSC}_{D,E}(\infty)$ to $\Sigma$ and $\Omega$: It will turn out that $\Omega$ can be implemented under $\text{VSSC}_{D,E}(\infty)$, but $\Sigma$ cannot.

In fact, $\Omega$ can even be implemented atop of the strictly stronger message adversary $\text{VSSC-PART}_{D,E}(\infty)$, under which consensus is impossible:

**Definition 25.** $\text{VSSC-PART}_{D,E}(\infty)$ contains those graph sequences where, for some round $r_{ST}$, there is $D$-bounded, $E$-influencing $I$-VSSC with $I = [r_{ST}, \infty)$

Put differently, $\text{VSSC-PART}_{D,E}(\infty)$ allows partitioning of the communication graph into multiple connected components for an arbitrary, finite number of rounds until some unique VSSC remains forever. Perhaps not surprisingly, this is insufficient to solve consensus:

**Lemma 26.** *Consensus is impossible under the message adversary VSSC-PART$_{D,E}(\infty)$.*

**Proof.** For simplicity, we will restrict our attention to the case $n = 2$; extending the proof for arbitrary $n$ is straightforward. Suppose some algorithm $\mathcal{A}$ solves consensus under this adversary. By termination and validity, there is some round $\tau$ where $\mathcal{A}$ lets $p_i$ decide $x_i$ in a run $\varepsilon$ starting from some initial configuration $C^0$ with the graph sequence $\sigma = (p_i \to p_j)_{r>0}$. Similarly, in the run $\varepsilon'$ that also starts from $C^0$ using $\sigma' = (p_i \leftarrow p_j)_{r>0}$, $\mathcal{A}$ will eventually let $p_j$ decide $x_j$. Now consider the run $\varepsilon''$ also starting from $C^0$ with sequence $\sigma'' = (p_i \quad p_j)_{r=1}^{\tau}(p_i \leftarrow p_j)_{r>\tau}$, where $(p_i \quad p_j)_{r=1}^{\tau}$ means that no message is successfully delivered in either direction in the first $\tau$ rounds. Clearly, until round $\tau$, $p_i$ will have exactly the same view in the run $\varepsilon$ and in the run $\varepsilon'$, denoted $\varepsilon \sim_{p_i} \varepsilon''$, thus $p_i$ decides $x_i$ in the run $\varepsilon''$. Similarly, $\varepsilon' \sim_{p_j} \varepsilon''$ until $\tau$, so $p_j$ decides $x_j$ in this run. Because $\sigma, \sigma', \sigma'' \in \text{VSSC-PART}_{D,E}(\infty)$, this contradicts the assumption that $\mathcal{A}$ solves consensus under this message adversary. □

However, the following lemma shows that $\text{VSSC-PART}_{D,E}(\infty)$ allows implementing $\Omega$.

**Lemma 27.** *$\mathcal{SMP}_n[adv : \text{VSSC-PART}_{D,E}(\infty)]$ allows to implement $\mathcal{AMP}_{n,n-1}[fd : \Omega]$.*

**Proof.** Consider an algorithm that outputs, at process $p_i$, the process with the largest identifier in the source component that was detected $E$ rounds ago, or itself if no such source component was detected. Clearly, this output is in the range of $\Omega$. Furthermore, since $\text{VSSC-PART}_{D,E}(\infty)$ guarantees that eventually some $D$-bounded, $E$-influencing source component $S$ remains the only VSSC forever, $S$ will be eventually detected by every process $p_i$ forever, and its member with the largest identifier will be written to the output of $p_i$ eventually forever as well. By Definition 22, no processes of $S$ is faulty, hence the specification of $\Omega$ is satisfied.

To simulate $\mathcal{AMP}$ with process crashes, exactly the same simulation as in [23, Sec.4.2] is used: Analogous to the simulation used in the proof of Corollary 4, a simulated process is only allowed to take its next step if all the messages sent in the previous step are already known by the simulator of the current output of $\Omega$, which (eventually) will be a strongly correct process. □

Finally, since all sequences of $\text{VSSC}_{D,E}(\infty)$ are contained in $\text{VSSC-PART}_{D,E}(\infty)$, it follows that $\Omega$ can indeed also be implemented under $\text{VSSC}_{D,E}(\infty)$.

We will now turn our attention to $\Sigma$: The following theorem shows that $\Sigma$ cannot be implemented atop of $\text{VSSC}_{D,E}(\infty)$.

**Lemma 28.** *$\mathcal{SMP}_n[adv : \text{VSSC}_{D,E}(\infty)]$ does not allow to implement $\mathcal{AMP}_{n,n-1}[fd : \Sigma]$.*

**Proof.** Again, 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 $p_i$, some algorithm $\mathcal{A}$ computes $out(p_i, r)$ s.t. for any admissible failure pattern $F$, $out \in \Sigma(F)$. Consider the graph sequence $\sigma = (p_i \to p_j)_{r\geqslant 1}$. Clearly, the failure pattern associated with $\sigma$ is $F_\sigma(r) = \{p_j\}$. Hence, in the run $\varepsilon$ starting from some initial configuration $C^0$ with sequence $\sigma$, there is some round $r'$ s.t. $out(p_i, r) = \{p_i\}$ for any $r > r'$ by Definition 21. Let $\sigma' = (p_i \to p_j)_{r=1}^{r'}(p_i \leftarrow p_j)_{r>r'}$. By similar arguments as above, in the run $\varepsilon'$ that starts from $C^0$ with sequence $\sigma'$, there is a round $r''$ such that $out(p_j, r) = \{p_j\}$ for any $r > r''$. Finally, for $\sigma'' = (p_i \to p_j)_{r=1}^{r'}(p_i \leftarrow p_j)_{r=r'+1}^{r''}(p_i \leftrightarrow p_j)_{r>r''}$, let $\varepsilon''$ denote the run starting from $C^0$ with graph sequence $\sigma''$. Until round $r'$, $\varepsilon'' \sim_{p_i} \varepsilon$, hence, as shown above, $out(p_i, r') = \{p_i\}$ in $\varepsilon''$. Similarly, until round $r''$, $\varepsilon'' \sim_{p_j} \varepsilon'$ and hence $out(p_j, r'') = \{p_j\}$

in $\varepsilon''$. Clearly, $\sigma, \sigma', \sigma'' \in \text{VSSC}_{D,E}(\infty)$ and $F_{\sigma''}(r) = \{\}$, that is, no process is faulty in $\sigma''$. However, in $\varepsilon''$, $out(p_i, r') \cap out(p_j, r'') = \emptyset$, a contradiction to Definition 21. □

The above result may come as a surprise, since the proof of the necessity of $\Sigma_k$ for $k$-set agreement (hence the necessity of $\Sigma = \Sigma_1$ for consensus) developed by Raynal et al. [59] 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 is no safety violation (i.e., a decision on a value that eventually leads to a violation of $k$-agreement) in any prefix of a run. This is not the case in $\mathcal{SMP}_n$, however, as processes may decide after a certain number of rounds also if no message is received. Hence, we cannot reuse their proof in our setting.

Taken together, Lemmas 26, 27, and 28 allow us to conclude the following:

(i) Since $\text{VSSC}_{D,E}(\infty)$ (not to speak of $\text{VSSC}_{D,E}(d)$, which is not compatible with failure detector specifications) does not allow to implement $(\Sigma, \Omega)$, we cannot derive consensus algorithms from $(\Sigma, \Omega)$-based solutions. And indeed, our consensus algorithm (Algorithm 2) is algorithmically very different.
(ii) The message adversaries SOURCE and QUORUM considered in [23], which allow to implement $(\Sigma, \Omega)$, are equivalent to $\text{VSSC}_{D,E}(\infty)$ in terms of consensus solvability, but strictly weaker in terms of sequence inclusion, i.e., $(\text{SOURCE}, \text{QUORUM}) \subset \text{VSSC}_{D,E}(\infty)$.

### 8.3. k-Set agreement

We start with the definitions of generalized failure detectors for the $k$-set agreement setting in crash-prone asynchronous message passing systems, using the notation introduced in Section 8.1.

**Definition 26.** The range of the failure detector $\Omega_k$ is all $k$-subsets of $2^\Pi$. 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 $t' \geqslant t$, $p_i \in \mathcal{C} : H(p_i, t') = LD$.

**Definition 27.** The failure detector $\Sigma_k$ has range $2^\Pi$. 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 $|S| = k+1$, $\exists p_i, p_j \in S : H(p_i, t) \cap H(p_j, t') \neq \emptyset$, (2) there is a time $t \in \mathcal{T}$ s.t. for every process $p_i$, for every $t' \geqslant t$: $H(p_i, t') \subseteq \mathcal{C}$.

$k$-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 $p_i$ has at least one process $p_j$ such that $\forall r' > r : \mathrm{s}_i^r \rightsquigarrow \mathrm{s}_j^{r'}$, but no $p_i$ usually reaches all $p_j \in \Pi$ here. Definition 22 hence implies that there is no correct process in this setting. Hence, we employ the following generalized definition:

**Definition 28.** Given a infinite graph sequence $\sigma$, let a *minimal source set* $S$ in $\sigma$ be a set of processes with the property that $\forall p_j \in \Pi, \forall r > 0$ there exists $p_i \in S, r' > r$ such that $\mathrm{s}_i^r \rightsquigarrow \mathrm{s}_j^{r'}$. The set of *weakly correct* processes $\mathcal{WC}(\sigma)$ of a sequence $\sigma$ is the union of all minimal source sets $S$ in $\sigma$.

This definition is a quite natural extension of correct processes in a model, which allows perpetual partitioning of the system. In particular, it is not difficult to show that $\mathcal{WC}(\sigma) \neq \emptyset$ for $\sigma \in \text{VSSC}_{D,H}(n, \infty) + MAJINF(k)$:

**Lemma 29.** *For every $\sigma \in \text{VSSC}_{D,H}(n, \infty) + MAJINF(k)$, it holds that $\mathcal{WC}(\sigma) \neq \emptyset$.*

**Proof.** By Definition 15, for any $\sigma \in \text{VSSC}_{D,H}(n, \infty)$, there is some non-empty, $H$-influencing set of $D$-bounded VSSCs $S_1, \ldots S_\ell$ from some round onwards in $\sigma$. According to Definition 28, $\bigcup_{i=1}^{\ell} S_i \subseteq \mathcal{WC}$. □

Based on this definition of weakly correct processes, it is possible to generalize some of our consensus-related results (obtained for $\Sigma$ and $\Omega$). First, we show that $\Sigma_k$ cannot be implemented, since $\text{VSSC}_{D,H}(n, \infty) + MAJINF(k)$ allows the system to partition into $k$ isolated components.

**Lemma 30.** *$\Sigma_k$ cannot be implemented under $\text{VSSC}_{D,H}(n, \infty) + MAJINF(k)$.*

**Proof.** For $k = 1$, we can rely on Lemma 28, as every $\sigma \in \text{VSSC}_{D,E}(\infty)$ is also admissible in $\text{VSSC}_{D,H}(n, \infty) + MAJINF(1)$. Hence, $\Sigma_1 = \Sigma$ cannot be implemented in $\text{VSSC}_{D,H}(n, \infty) + MAJINF(1)$.

The impossibility can be expanded to $k > 1$ by choosing some $\sigma$ 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 $\Sigma$ in $P$ (e.g., the message adversary could emulate the graph sequence used in Lemma 28 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 $p_i \in P$, it is indistinguishable whether any $p_j \in \widetilde{P} \setminus P$ is faulty in $\sigma$ or not, $p_i$ has to assume that every process $p_j \in \widetilde{P} \setminus P$ is faulty. Hence, for every $p_i \in P$, we must eventually have $out(p_i, r_i) \subseteq P$ for some sufficiently large $r_i$.

We now construct a set $S$ of $k+1$ processes that violates Definition 27: 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 $\forall P_j \in \widetilde{P} \setminus P, \forall p_i \in P_j, \exists r_i \colon out(p_i, r_i) \subseteq P_i$ and, by the construction of $S$, we have that $\forall p_i, p_j \in S$, $\exists r_i, r_j$ such that $out(p_i, r_i) \cap out(p_j, r_j) = \emptyset$. This set $S$ clearly violates Definition 27, as required. □

As for $\Omega_k$, we note that Lemma 27 reveals also that $\Omega_1 = \Omega$ can be implemented under VSSC-PART$_{D,E}(\infty)$. By contrast, however, $\Omega_k$ it is not implementable under VSSC$_{D,H}(n, \infty) + MAJINF(k)$ for $k > 1$:

**Lemma 31.** *For $k > 1$, $\Omega_k$ cannot be implemented under VSSC$_{D,H}(n, \infty) + MAJINF(k)$.*

**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 $\Omega_k$ under the adversary leads to a contradiction. The following graph sequences (a)–(e) are all admissible sequences under VSSC$_{D,H}(k, \infty)$ (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 26 and 28, $LD$ has to include $p_1$ in $\varepsilon_a$, $p_2$ in $\varepsilon_b$, and $p_3$ in $\varepsilon_c$. By Definition 26, in $\varepsilon_d$, 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 $\varepsilon_e$, 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 $\varepsilon_d$ and $\varepsilon_e$. □

## 9. Conclusions

We introduced a framework for modeling dynamic networks with directed communication links under generalized message adversaries that focus on vertex-stable source components. We presented a number of impossibility results and lower bounds for consensus, as well as an algorithm that solves consensus under the strongest message adversary known so far. Moreover, we made a significant step towards determining the solvability/impossibility border of general $k$-set agreement in our model: We provided several impossibility results and lower bounds, which also led us to the first gracefully degrading consensus/$k$-universal $k$-set agreement under fairly strong message adversaries proposed so far. Our results are complemented by relating our message adversaries to failure detectors. Our results show that the weakest resp. necessary failure detectors for consensus resp. $k$-set agreement cannot be implemented under our message adversaries.

Part of our future work is devoted to finding even stronger message adversaries and matching algorithms, as well as even stronger lower bounds, in an attempt to close the remaining gap.

## References

[1] F. Kuhn, R. Oshman, Dynamic networks: models and algorithms, SIGACT News 42 (1) (2011) 82–96.
[2] F. Kuhn, S. Schmid, R. Wattenhofer, Towards worst-case churn resistant peer-to-peer systems, Distrib. Comput. 22 (4) (2010) 249–267.
[3] M. Faezipour, M. Nourani, A. Saeed, S. Addepalli, Progress and challenges in intelligent vehicle area networks, Commun. ACM 55 (2) (2012) 90–100, https://doi.org/10.1145/2076450.2076470.
[4] J. Augustine, G. Pandurangan, P. Robinson, Fast byzantine agreement in dynamic networks, in: Proceedings PODC'13, 2013, pp. 74–83.
[5] W. Kiess, M. Mauve, A survey on real-world implementations of mobile ad-hoc networks, Ad Hoc Netw. 5 (3) (2007) 324–339, https://doi.org/10.1016/j.adhoc.2005.12.003.
[6] I. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci, Wireless sensor networks: a survey, Comput. Netw. 38 (4) (2002) 393–422, https://doi.org/10.1016/S1389-1286(01)00302-4.
[7] J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, Comput. Netw. 52 (12) (2008) 2292–2330, https://doi.org/10.1016/j.comnet.2008.04.002.
[8] F. Legendre, T. Hossmann, F. Sutton, B. Plattner, 30 years of wireless ad hoc networking research: what about humanitarian and disaster relief solutions? What are we still missing?, in: International Conference on Wireless Technologies for Humanitarian Relief, ACWR 11, IEEE, Amrita, India, 2011.
[9] C. Ware, J. Judge, J. Chicharo, E. Dutkiewicz, Unfairness and capture behaviour in 802.11 adhoc networks, in: IEEE International Conference on Communications. Global Convergence Through Communications, ICC 2000, 2000.

[10] A. Cerpa, J.L. Wong, M. Potkonjak, D. Estrin, Temporal properties of low power wireless links: modeling and implications on multi-hop routing, in: Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc'05, ACM, New York, NY, USA, 2005, pp. 414–425.
[11] V. Rajendran, K. Obraczka, J.J. Garcia-Luna-Aceves, Energy-efficient collision-free medium access control for wireless sensor networks, in: Proceedings of the 1st International Conference on Embedded Networked Sensor Systems, SenSys'03, ACM, New York, NY, USA, 2003, pp. 181–192.
[12] A. Cerpa, J.L. Wong, L. Kuang, M. Potkonjak, D. Estrin, Statistical model of lossy links in wireless sensor networks, in: Proceedings of the 4th International Symposium on Information Processing in Sensor Networks, IPSN'05, IEEE Press, Piscataway, NJ, USA, 2005, http://dl.acm.org/citation.cfm?id=1147685.1147701.
[13] U. Schilcher, C. Bettstetter, G. Brandner, Temporal correlation of interference in wireless networks with Rayleigh block fading, IEEE Trans. Mob. Comput. 11 (12) (2012) 2109–2120, https://doi.org/10.1109/TMC.2011.244.
[14] A. Goiser, S. Khattab, G. Fassl, U. Schmid, A new robust interference reduction scheme for low complexity direct-sequence spread-spectrum receivers: performance, in: Proceedings 3rd International IEEE Conference on Communication Theory, Reliability, and Quality of Service, CTRQ'10, Athens, Greece, 2010, pp. 15–21.
[15] M. Fussen, R. Wattenhofer, A. Zollinger, Interference arises at the receiver, in: International Conference on Wireless Networks, Communications and Mobile Computing, vol. 1, 2005, pp. 427–432.
[16] C. Newport, D. Kotz, Y. Yuan, R.S. Gray, J. Liu, C. Elliott, Experimental evaluation of wireless simulation assumptions, SIMULATION: Trans. Soc. Model. Simul. Int. 83 (9) (2007) 643–661, https://doi.org/10.1177/0037549707085632.
[17] F. Kuhn, N.A. Lynch, R. Oshman, Distributed computation in dynamic networks, in: STOC, 2010, pp. 513–522.
[18] F. Kuhn, R. Oshman, Y. Moses, Coordinated consensus in dynamic networks, in: Proceedings of the 30th Annual ACM SIGACT–SIGOPS Symposium on Principles of Distributed Computing, PODC'11, ACM, 2011.
[19] M. Maróti, B. Kusy, G. Simon, A. Lé deczi, The flooding time synchronization protocol, in: Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems, SenSys'04, ACM, New York, NY, USA, 2004, pp. 39–49.
[20] B. Awerbuch, Complexity of network synchronization, J. ACM 32 (4) (1985) 804–823, https://doi.org/10.1145/4221.4227.
[21] S. George, W. Zhou, H. Chenji, M. Won, Y.O. Lee, A. Pazarloglou, R. Stoleru, P. Barooah, Distressnet: a wireless ad hoc and sensor network architecture for situation management in disaster response, IEEE Commun. Mag. 48 (3) (2010) 128–136, https://doi.org/10.1109/MCOM.2010.5434384.
[22] S. Chaudhuri, More choices allow more faults: set consensus problems in totally asynchronous systems, Inf. Control 105 (1) (1993) 132–158.
[23] M. Raynal, J. Stainer, Synchrony weakened by message adversaries vs asynchrony restricted by failure detectors, in: Proceedings ACM Symposium on Principles of Distributed Computing, PODC'13, 2013, pp. 166–175.
[24] A. Casteigts, P. Flocchini, W. Quattrociocchi, N. Santoro, Time-varying graphs and dynamic networks, Int. J. Parallel Emergent Distrib. Syst. 27 (5) (2012) 387–408.
[25] M. Biely, P. Robinson, U. Schmid, Easy impossibility proofs for k-set agreement in message passing systems, in: Proceedings 15th International Conference on Principles of Distributed Systems, OPODIS'11, in: Springer LNCS, vol. 7109, 2011, pp. 299–312.
[26] F. Harary, G. Gupta, Dynamic graph models, Math. Comput. Modelling 25 (7) (1997) 79–87, https://doi.org/10.1016/S0895-7177(97)00050-2.
[27] Y. Afek, E. Gafni, Asynchrony from synchrony, in: D. Frey, M. Raynal, S. Sarkar, R. Shyamasundar, P. Sinha (Eds.), Distributed Computing and Networking, in: Lecture Notes in Computer Science, vol. 7730, Springer, Berlin, Heidelberg, 2013, pp. 225–239.
[28] N. Santoro, P. Widmayer, Time is not a healer, in: Proc. 6th Annual Symposium on Theor. Aspects of Computer Science, STACS'89, Paderborn, Germany, in: LNCS, vol. 349, Springer-Verlag, 1989, pp. 304–313.
[29] 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, Puerto Vallarta, Mexico, ACM Press, 1998, pp. 143–152.
[30] B. Charron-Bost, A. Schiper, The Heard-Of model: computing in distributed systems with benign faults, Distrib. Comput. 22 (1) (2009) 49–71, https://doi.org/10.1007/s00446-009-0084-6.
[31] M. Biely, U. Schmid, B. Weiss, Synchronous consensus under hybrid process and link failures, Theoret. Comput. Sci. 412 (40) (2011) 5602–5630, https://doi.org/10.1016/j.tcs.2010.09.032.
[32] J. Augustine, G. Pandurangan, P. Robinson, E. Upfal, Towards robust and efficient computation in dynamic peer-to-peer networks, in: Proceedings of the Twenty-Third Annual ACM–SIAM Symposium on Discrete Algorithms, SODA'12, SIAM, 2012, pp. 551–569.
[33] H.C. Chung, P. Robinson, J.L. Welch, Optimal regional consecutive leader election in mobile ad-hoc networks, in: Proceedings of the 7th ACM SIGACT/SIGMOBILE International Workshop on Foundations of Mobile Computing, FOMC'11, ACM, New York, NY, USA, 2011, pp. 52–61.
[34] U. Schmid, B. Weiss, I. Keidar, Impossibility results and lower bounds for consensus under link failures, SIAM J. Comput. 38 (5) (2009) 1912–1951, http://www.vmars.tuwien.ac.at/documents/extern/2554/paper.pdf.
[35] M. Biely, P. Robinson, U. Schmid, Agreement in directed dynamic networks, in: Proceedings 19th International Colloquium on Structural Information and Communication Complexity, SIROCCO'12, in: LNCS, vol. 7355, Springer-Verlag, 2012, pp. 73–84.
[36] É. Coulouma, E. Godard, A characterization of dynamic networks where consensus is solvable, in: Proceedings Structural Information and Communication Complexity – 20th International Colloquium, SIROCCO'13, in: LNCS, vol. 8179, Springer, 2013, pp. 24–35.
[37] M. Biely, P. Robinson, U. Schmid, M. Schwarz, K. Winkler, Gracefully degrading consensus and *k*-set agreement in directed dynamic networks, Revised selected papers, in: Third International Conference on Networked Systems, NETYS'15, Agadir, Morocco, in: Springer LNCS, vol. 9466, Springer International Publishing, 2015, pp. 109–124.
[38] M. Schwarz, K. Winkler, U. Schmid, Fast consensus under eventually stabilizing message adversaries, in: Proceedings of the 17th International Conference on Distributed Computing and Networking, ICDCN'16, ACM, New York, NY, USA, 2016, pp. 1–10.
[39] K. Winkler, M. Schwarz, U. Schmid, Consensus in directed dynamic networks with short-lived stability, CoRR, arXiv:1602.05852.
[40] F. Kuhn, N. Lynch, R. Oshman, Distributed computation in dynamic networks, in: ACM STOC, 2010, pp. 513–522.
[41] É. Coulouma, E. Godard, J.G. Peters, A characterization of oblivious message adversaries for which consensus is solvable, Theoret. Comput. Sci. 584 (2015) 80–90, https://doi.org/10.1016/j.tcs.2015.01.024.
[42] H.C. Chung, P. Robinson, J.L. Welch, Regional consecutive leader election in mobile ad-hoc networks, in: Proceedings of the DIALM–POMC Joint Workshop on Foundations of Mobile Computing, 2010, pp. 81–90.
[43] A. Sealfon, A.A. Sotiraki, Brief announcement: agreement in partitioned dynamic networks, in: Proceedings 28th International Symposium on Distributed Computing, DISC'14, in: Springer LNCS, vol. 8784, 2014, pp. 555–556.
[44] M. Biely, P. Robinson, U. Schmid, Solving k-set agreement with stable skeleton graphs, in: IPDPS Workshops, 2011, pp. 1488–1495, arXiv:1102.4423.
[45] M. Biely, P. Robinson, U. Schmid, The generalized loneliness detector and weak system models for *k*-set agreement, IEEE Trans. Parallel Distrib. Syst. 25 (4) (2014) 1078–1088, http://doi.ieeecomputersociety.org/10.1109/TPDS.2013.77.
[46] N.H. Vaidya, D.K. Pradhan, Degradable agreement in the presence of Byzantine faults, in: International Conference on Distributed Computing Systems, 1993, pp. 237–244, citeseer.nj.nec.com/vaidya92degradable.html.
[47] C. Dwork, D. Peleg, N. Pippenger, E. Upfal, Fault tolerance in networks of bounded degree, SIAM J. Comput. 17 (5) (1988) 975–988.
[48] M.K. Aguilera, W. Chen, S. Toueg, Using the heartbeat failure detector for quiescent reliable communication and consensus in partitionable networks, Theoret. Comput. Sci. 220 (1) (1999) 3–30, http://www.cs.cornell.edu/home/sam/FDpapers/TCS98final.ps.

[49] R. Ingram, P. Shields, J.E. Walter, J.L. Welch, An asynchronous leader election algorithm for dynamic networks, in: IPDPS, 2009, pp. 1–12.
[50] D. Peleg, Distributed Computing: A Locality-Sensitive Approach, SIAM Society for Industrial and Applied Mathematics Monographs on Discrete Mathematics and Applications, SIAM, Philadelphia, 2000.
[51] T. Elrad, N. Francez, Decomposition of distributed programs into communication-closed layers, Sci. Comput. Program. 2 (3) (1982) 155–173.
[52] K.J. Perry, S. Toueg, Distributed agreement in the presence of processor and communication faults, IEEE Trans. Softw. Eng. SE-12 (3) (1986) 477–482.
[53] T.D. Chandra, V. Hadzilacos, S. Toueg, The weakest failure detector for solving consensus, J. ACM 43 (4) (1996) 685–722, http://www.cs.cornell.edu/home/sam/FDpapers/CHT96-JACM.ps.
[54] I. Ben-Zvi, Y. Moses, Beyond Lamport's happened-before: on the role of time bounds in synchronous systems, in: N. Lynch, A. Shvartsman (Eds.), Distributed Computing, in: Lecture Notes in Computer Science, vol. 6343, Springer, Berlin/Heidelberg, 2010, pp. 421–436.
[55] J. Augustine, G. Pandurangan, P. Robinson, S. Roche, E. Upfal, Enabling efficient and robust distributed computation in highly dynamic networks, in: 56th Annual IEEE Symposium on Foundations of Computer Science, FOCS, 2015, pp. 350–369.
[56] S. Hoory, N. Linial, A. Wigderson, Expander graphs and their applications, Bull. Amer. Math. Soc. 43 (04) (2006) 439–562, https://doi.org/10.1090/S0273-0979-06-01126-8.
[57] N. Lynch, Distributed Algorithms, Morgan Kaufman, 1996.
[58] M.J. Fischer, N.A. Lynch, M.S. Paterson, Impossibility of distributed consensus with one faulty process, J. ACM 32 (2) (1985) 374–382.
[59] F. Bonnet, M. Raynal, On the road to the weakest failure detector for k-set agreement in message-passing systems, Theoret. Comput. Sci. 412 (33) (2011) 4273–4284, https://doi.org/10.1016/j.tcs.2010.11.007.
[60] S.N. Dorogovtsev, J.F.F. Mendes, A.N. Samukhin, Giant strongly connected component of directed networks, Phys. Rev. E 64 (2001) 025101, https://doi.org/10.1103/PhysRevE.64.025101.
[61] S. Janson, D.E. Knuth, T. Luczak, B. Pittel, The birth of the giant component, Random Structures Algorithms 4 (1993) 233–358.
[62] F. Mattern, Virtual time and global states of distributed systems, in: Parallel and Distributed Algorithms, North-Holland, 1989, pp. 215–226.
[63] T.D. Chandra, S. Toueg, Unreliable failure detectors for reliable distributed systems, J. ACM 43 (2) (1996) 225–267, http://www.cs.cornell.edu/home/sam/FDpapers/CT96-JACM.ps.
[64] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, V. Hadzilacos, P. Kouznetsov, S. Toueg, The weakest failure detectors to solve certain fundamental problems in distributed computing, in: Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing, PODC'04, ACM Press, 2004, pp. 338–346.