Gracefully Degrading Consensus and k-Set Agreement under Dynamic Link Failures

Manfred Schwarz[1], Kyrill Winkler[1], Ulrich Schmid[1], Martin Biely[2], and Peter Robinson[3]

[1] ECS Group, TU Wien, Austria, {mschwarz,kwinkler,s}@ecs.tuwien.ac.at
[2] EPFL, Switzerland, martin.biely@epfl.ch
[3] National University of Singapore, Singapore, robinson@comp.nus.edu.sg

Abstract. We consider synchronous directed dynamic networks, where the communication graphs may change in every round and links are not necessarily bidirectional. Since any non-trivial agreement problem is trivially impossible without constraints on the topology changes, determining sufficiently strong network assumptions is important for developing solution algorithms. On the other hand, for strong network assumptions, it may be difficult to guarantee that they indeed hold in a given dynamic network. Determining the exact solvability/impossibility border of agreement problems in dynamic networks is hence interesting both from a theoretical and a practical point of view. In this paper, we provide a significant step towards determining the solvability/impossibility border of general k-set agreement in synchronous dynamic networks with unidirectional links controlled by an omniscient message adversary: We provide impossibility results, which show that both some minimal temporal stability of the communication topology and some minimal information flow between temporally stable network components are mandatory. In addition, starting out from our impossibility results, we developed weak network assumptions that are sufficiently strong for solving k-set agreement, along with a corresponding gracefully degrading algorithm and its correctness proof.

Keywords: Dynamic networks, k-set agreement, weak network assumptions, impossibility results.

1 Introduction

Due to the increasing number of mobile ad-hoc networks and sensor networks, in a wide variety of different applications, much research has been and is being devoted to dynamic networks [19]. In sharp contrast to conventional wireline networks, the communication graphs of such systems are not static: Fading and interference phenomena [16], as well as mobility and duty-cycling for energy saving purposes, make the ability of any two processes to communicate with each other highly time-dependent. Moreover, capture effects and near-far problems [29], which affect the receiver side but not the sender side of a wireless link, make *asymmetric* communication links a reality.

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. Whereas the ability to reach *system-wide* consensus is of course optimal here, this requires strong constraints (including no partitioning) on the network topology and its evolution [6, 20] that are unlikely to hold in every dynamic network. Fortunately, weaker forms of agreement may be sufficient for certain applications. In case of determining communication schedules, 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. For such applications, suitably designed k-set agreement algorithms, 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 *k-uniform*, 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 increases under unfavorable conditions.

In this paper, we provide a major first step towards determining the exact solvability/impossibility border of general k-set agreement in synchronous systems made up of an (unknown) number n of processes, where the directed communication graphs in every round are controlled by an omniscient message adversary [1]: 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 ultimate goal of our research are network assumptions for every $1 \leqslant k < n$, which are both necessary and sufficient for solving k-set agreement. Knowing or at least approaching this border is interesting for several reasons: First, it is interesting from a theoretical point of view: k-set agreement has been a major target for the study of solvability in asynchronous systems with failure detectors since decades.[4] Second, striving for weak network assumptions is always advantageous w.r.t. the assumption coverage in real systems, as they are typically more likely to hold in a given dynamic network. Finally, a set of network assumptions close to the necessary and sufficient ones is needed for developing k-set agreement algorithms that indeed degrade *gracefully*: For a given communication scenario, such an algorithm should solve k-set agreement for the *smallest* k possible.

Major contributions and paper organization. In this paper, we tightly "enclose" necessary and sufficient assumptions for solving k-set agreement:

(1) We provide two (surprisingly strong) network assumptions, which are still too weak for solving k-set agreement. Our assumptions are based on the fundamental concept of *vertex-stable root components* (VSRC), presented along with our basic system model in Section 2, which are strongly connected components of $\mathcal{G}^r$ without incoming edges that consist of the same set of processes (with possibly varying interconnect) for a certain number of rounds. Note carefully that every directed graph has at least one root component. In Section 3, we show that even the restriction to at most k simultaneous VSRCs in every round is *not* sufficient for solving k-set agreement if just a single VSRC is vertex-stable for less than $n - k$ rounds: A generic reduction of k-set agreement to consensus introduced in [4], in conjunction with certain bivalence arguments, is used to construct a non-terminating run in this case. Moreover, we prove that *eventual* stability of *all* VSRCs is not enough for solving k-set agreement, not even when it is guaranteed that (substantially) less than k VSRCs exist simultaneously. The latter is a consequence of some "uncontrolled" partitioning over time, which could happen in dynamic networks.

(2) By slightly strengthening the above network properties, which are too weak for solving k-set agreement, we obtain a set of network assumptions that are sufficient for this purpose: As detailed in Section 4, *eventual stability* guarantees that at most k VSRCs become vertex-stable long enough simultaneously, while *majority influence* ensures that at most k of such VSRCs exist in a run that are "independent" of each other. Despite being rather weak properties, they allow to implement a k-uniform k-set agreement algorithm (that naturally respects partitions as well), which can be viewed as a gracefully degrading consensus algorithm.

Obviously, necessary and sufficient network conditions that determine the exact solvability/impossibility border of k-set agreement must lie somewhere in between (1) and (2).

Related work. Dynamic networks have been studied intensively in distributed computing (see the overview by Kuhn and Oshman [19] and the references therein). Besides work on peer-to-peer networks like [21], 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.

Agreement problems in dynamic networks with undirected communication graphs have been studied in [3, 13, 20]; agreement in directed graphs has been considered in [1, 6, 14, 25, 27]. Whereas [14, 27] considerably restrict the dynamicity of the communication graphs, e.g., by not allowing stabilizing behavior, which effectively causes them to belong to quite strong classes of network assumptions in the classification of Casteigts et al. [10], [6, 7] allows to solve consensus under very weak network assumptions. The leader election problem in dynamic networks has been studied in [13]. Afek and Gafni [1] introduced message adversaries, which make problems solvable in wait-free read-write shared memory systems also solvable in message-passing systems. Raynal and Stainer [25] looked at the relationship between round-based models and failure detectors.

Most of the above work is based on (variants of) the model introduced in [18], 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) Besides time varying graphs, as we consider them in this paper, several related alternative approaches have also been proposed in

[4] Despite all efforts, however, the weakest failure detector for message-passing k-set agreement is still unknown [9]. Interestingly, [25] revealed that there are relations between this classic model and dynamic networks.

the past: Moving omission failures [26], round-by-round fault detectors [15], the heard-of model [11] and the perception-based failure model [8].

In [6, 7], we introduced the system and basic network model also used in this paper for studying consensus in directed dynamic networks. In particular, we provided a consensus algorithm that works under the assumption that, in every round, the communication graph is both (i) weakly connected and (ii) contains a single root component (a strongly connected component without incoming links) that must eventually become vertex stable. Since these assumptions[5] do not guarantee bidirectional reachability system-wide, the model in [6] falls between the weakest and second weakest class of models defined in [10]. Related impossibility results reveal that there is no hope for successfully weakening this model further. Nevertheless, in larger-scale dynamic networks, it is unrealistic to assume that the above properties can always be guaranteed. Regarding k-set agreement in dynamic networks, we are not aware of any previous work except our previous paper [5], 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* root components. Note that this essentially implies a dynamic network with a static core. By contrast, in this paper, we allow the communication graphs to be fully dynamic.

We are also not aware of related work exploring gracefully degrading consensus or k-uniform 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 [28], where processes are allowed to also decide on a (fixed) default value in case of excessive faults. Aguilera et. al. [2] 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: On the one hand, we allow more different decisions, on the other hand, all correct processes are required to decide and every decisions must be the initial value of some process. Ingram et. al. [17] 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 [17] is guaranteed to successfully elect a leader only once the topology eventually stabilizes.

2 Model

The *k-set agreement problem* [12] is defined as follows: Each process p_i starts with an initial value x_i taken from some set $\mathcal{V}$, where $|\mathcal{V}| > k$, and must irrevocably decide on some y_i, such that the following properties hold in all runs:
(k-Agreement) At most k different decision values are obtained system-wide in any run.
(Validity) If $y_i = v$, then v is some p_j's initial value x_j.
(Termination) Eventually, every process p_i assigns a value to y_i.

Consensus is the special case of 1-set agreement. We call a k-set agreement algorithm *k-uniform*, if it does not require knowledge of k.

Computing model. We use the same synchronous dynamic network model as in [6, 7], which assumes a set $\Pi = \{p_1, \ldots, p_n\}$ of processes that know their own id i but not necessarily n. Processes execute an infinite number of rounds $r = 1, 2, \ldots$ (conceptually) in lock-step. In every round r, processes first broadcast a round r message of arbitrary content, determined by some message sending function, and then perform some deterministic local computation based on the received round r messages and their current (local) state. The actual communication in the system is determined by the sequence of communication graphs $\mathcal{G}^r$, $r > 0$, fixed by the adversary.[6] $\mathcal{G}^r$ contains a directed edge $(p \to q)$ from process p to q iff q receives p's round r broadcast in round r. The set $\mathcal{N}_p^r$ denotes p's (in-)*neighbors* in round r. We emphasize that p does not have any *a priori* knowledge of its neighbors, i.e., p 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.

[5] Note that the sufficient network assumptions for $k = 1$ obtained in Section 4 are implied by the (stronger) assumptions introduced for consensus in [6], but not vice versa.

[6] 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 [24] (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 $\mathcal{G}^r$.

Since $\mathcal{G}^r$ can range arbitrarily from n isolated nodes to a fully connected graph, there is no hope to solve k-set agreement without restricting the adversary to some extent. We will now introduce the notation used for specifying our constraints, which are collectively termed *network assumptions* in the sequel. Before presenting the formal definitions, we provide an informal overview of the cornerstones of our model: As in [6], we focus on the pivotal concept of root components, which are strongly connected components without incoming edges from processes outside the component. We require these root components to be vertex-stable, i.e., assume that their vertex sets (but not their interconnect) remain the same for a sufficiently large number of rounds. This ensures that all members can receive information from each other: Using a suitable notion of causal distance between processes, we introduce D-bounded vertex-stable root components (see Def. 2), which capture intra-root information propagation within D rounds. Similarly, H-network boundedness (Def. 3) will allow us to capture network-wide information propagation from root component(s) to all processes in the system.

Similarly to the classic "happened-before" relation [22], we say that a process p *causally influences* q *in round* r, denoted by $(p \overset{r}{\rightsquigarrow} q)$, iff either (i) q has an incoming edge $(p \to q)$ from p in $\mathcal{G}^r$, or (ii) if $q = p$, i.e., we assume that p always causally influences itself in a round. We say that there is a (causal) chain of length $k \geqslant 1$ starting from p in round r to q, denoted by $(p \overset{r[k]}{\rightsquigarrow} q)$, if there exists a sequence of not necessarily distinct processes $p = p_0, \ldots, p_k = q$ such that $p_i \overset{r+i}{\rightsquigarrow} p_{i+1}$ for $0 \leqslant i < k$. The *causal distance* $cd^r(p, q)$ at round r from process p to process q is the length of the shortest causal chain starting in p in round r and ending in q, formally $cd^r(p, q) := min\{k | (p \overset{r[k]}{\rightsquigarrow} q)\}$. We define $cd^r(p, p) = 1$ and $cd^r(p, q) = \infty$ if p never influences q after round r.

Definition 1 (Vertex-Stable Root Component). *A* root component R^r*, with set of vertices* $R \subseteq \Pi$*, is a* strongly connected component *(SCC) in* $\mathcal{G}^r$ *that has no incoming edges from other components, formally* $\forall p \in R^r, \forall q \in \mathcal{G}^r \colon (q \ \to \ p) \in \mathcal{G}^r \Rightarrow q \in R^r$.

An I-vertex-stable root component R^I *requires that the set of vertices of the strongly connected root component* R^x *in round* x *remains the same throughout all rounds* $x \in I = [r, s]$*,* $s \geqslant r$*, albeit the interconnection topology may change.*

We will abbreviate R^I as an I-VSRC or $|I|$-VSRC if only the length of I matters, and sometimes denote an I-VSRC R^I by its vertex set R if I is clear from the context. Note carefully that we assume $|I| = s - r + 1$ here, since I ranges from the *beginning* of round r to the *end* of round s; hence, $I = [r, r]$ is not empty but rather represents round r and has $|I| = 1$.

By contracting SCCs, it is easy to see that every weakly connected directed simple graph $\mathcal{G}$ has at least one root component. Hence, if $\mathcal{G}$ has k root components, it has at most k weakly connected components (with disjoint root components, but possibly overlapping remaining processes).

The important property of a VSRC R^I is that information is guaranteed to spread to all its vertices R if the interval I is large enough, as proved in [6, Lem. 2]: Defining the round x *causal diameter* $\varnothing^x(R^I)$ of a I-VSRC R^I as the largest round x causal distance $cd^x(p, q)$ between any pair of processes $p, q \in R$, formally, $\varnothing^x(R^I) := \max_{p,q \in R}\{cd^x(p, q)\}$, we can restate this result as follows:

Lemma 1 (Bound on causal diameter [6, Lemma 2]). *Given some* $I = [r, s]$ *and a VSRC* R^I *with* $|R| \geqslant 2$*: If* $s \geqslant r + |R| - 2$*, then* $\forall x \in [r, s - |R| + 2] \colon \varnothing^x(R^I) \leqslant |R| - 1$.

The following Def. 2 will be used in the sequel (cf. Asmp. 4) to concisely parameterize, via D, the worst-case information propagation in root components.

Definition 2 (D-bounded I-VSRC). *An* I*-vertex-stable root component* R^I *with* $I = [r, s]$ *is* D-bounded *if there exists some* $D \leqslant s - r + 1$ *such that for all but the last* $D - 1$ *rounds of* I *it is guaranteed that messages sent by any process in* R *reach all members of* R *within* I*, i.e.,* $\forall x \in [r, s - D + 1] \colon \varnothing^x(R^I) \leqslant D$.

Lem. 1 shows that every sufficiently long VSRC R^I is D-bounded for $D \geqslant |R| - 1$; all sufficiently long VSRCs are hence necessarily $n-1$-bounded. Choosing $D < n-1$ forces the adversary to speed-up information propagation accordingly.

In order to formalize information propagation from root components to the rest of the network, one has to account for the fact that a process q outside any root component may be reachable from *multiple* root

components. Note carefully that this allows to model dynamic networks that do not "cleanly" partition. Given a set $S^I = \{R_1^I, \dots, R_\ell^I\}$ of $\ell \geqslant 1$ I-vertex-stable root components, all vertex-stable in the same interval $I = [r, s]$, let the round x *network causal height* h^x be the maximum, taken over all processes $q \in \Pi$, of the minimal causal distance $cd^x(p, q)$ from *some* process $p \in \bigcup_{i=1}^{\ell} R_i^I$ in round x, formally $h^x(S^I) := \max_{q \in \Pi}\{\min_{p \in \cup_{i=1}^{\ell} R_i}\{cd^x(p, q)\}\}$. Def. 3 will be used in the sequel to guarantee that every process in the network receives a message from some member of at least one VSRC in $S^I = \{R_1^I, \dots, R_\ell^I\}$ within H rounds.

Definition 3 (H-network-bounded root components). *A set $S^I = \{R_1^I, \dots, R_\ell^I\}$ of $\ell \geqslant 1$ I-vertex-stable root components, for some interval $I = [r, s]$ with $|I| \geqslant H > 0$, is H-network-bounded, if $\forall x \in [r, s - H + 1] : h^x(S^I) \leqslant H$.*

Note that Def. 3 guarantees ($p \overset{x[H]}{\rightsquigarrow} q$) only for *some* but not for all $p \in R_i$. Moreover, p (and hence R_i) may be different for different starting rounds x in I.

The following Lem. 2 reveals that the network causal height h^x is bounded by $n-1$, provided $s-r \geqslant n-2$.

Lemma 2 (Bound on network causal height). *Suppose there is some interval $I = [r, s]$ where there is a set $S^I = \{R_1^I, \dots, R_\ell^I\}$ of exactly $\ell \geqslant 1$ I-vertex-stable root components. If $s \geqslant r + n - 2$ and $n \geqslant 2$, then S^I is H-network bounded for some $H \leqslant n - 1$.*

Proof. Let $P_0 = \bigcup_{i=1}^{\ell} R_i$ and fix any r' where $r \leqslant r' \leqslant s - n + 2$. Define, for each $i > 0$, the set $P_i = P_{i-1} \cup \{q : \exists q' \in P_{i-1} : q' \in \mathcal{N}_q^{r'+i-1}\}$. P_i is hence the set of processes q such that ($p \overset{r'[i]}{\rightsquigarrow} q$) holds for at least one $p \in P_0$. Using induction, we will show that $|P_k| \geqslant \min\{n, k + 1\}$ for $k \geqslant 0$. Induction start $k = 0 : |P_0| \geqslant \min\{n, 1\} = 1$ follows immediately from $P_0 \supseteq \{p_1, \dots, p_\ell\}$ with $\ell \geqslant 1$. Induction step $k \to k+1, k \geqslant 0$: First assume that already $|P_k| = n \geqslant \min\{n, k+1\}$; since $|P_{k+1}| \geqslant |P_k| = n \geqslant \min\{n, k+1\}$, we are done. Otherwise, consider round $r' + k$ and $|P_k| < n$: Since every node $q \in \Pi$ is in a weakly connected component containing at least one root in every round, hence also in $\mathcal{G}^{r'+k}$, there is a set of edges from processes in P_k to some non-empty set $L_k \subseteq \Pi \setminus P_k$. Hence, we have $P_{k+1} = P_k \cup L_k$, which implies $|P_{k+1}| \geqslant |P_k| + 1 \geqslant k + 1 + 1 = k + 2 = \min\{n, k + 2\}$ by the induction hypothesis. Thus, in order to guarantee $\Pi = P_k$ and thus $n = |P_k|$, choosing k such that $n = 1 + k$ and $k \leqslant s - r' + 1$ is sufficient. Since $s \geqslant r' + n - 2$, both conditions can be fulfilled by choosing $k = n - 1$. Moreover, due to the definition of P_k, it follows that for all $q \in \Pi$ there is some $p \in P_0$ with $cd^{r'}(p, q) \leqslant n - 1$, implying $h^{r'} \leqslant n - 1$. Since this holds for any $r' \leqslant s - n + 2$ following Def. 3, this implies Lem. 2.

According to Lem. 2, $H = n-1$ can always be guaranteed if a root component is vertex stable sufficiently long.

3 Impossibility Results

In this section, we establish necessary but not sufficient network assumptions for k-set agreement in directed dynamic networks. We accomplish this by proving that certain constraints do not allow to solve k-set agreement, using the generic impossibility theorem provided in [4, Thm. 1]. 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 α, β are *indistinguishable* (until decision) for a set of processes $\mathcal{D}$, denoted $\alpha \overset{\mathcal{D}}{\sim} \beta$, if for any $p \in \mathcal{D}$ it holds that p executes the same state transitions in α and in β (until it decides). Now consider a model of a distributed system $\mathcal{M} = \langle \Pi \rangle$ that consists of the set of processes Π 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. [4, 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 1 (k-Set Agreement Impossibility [4, 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 $D_1, \ldots, D_{k-1}$, and a set of distinct decision values $\{v_1, \ldots, v_{k-1}\}$. Moreover, let $\mathcal{D} = \bigcup_{1 \leqslant i < k} \mathcal{D}_i$ and $\overline{\mathcal{D}} = \Pi \setminus \mathcal{D}$. Consider the following two properties:*

(dec-$\mathcal{D}$) *For every set $\mathcal{D}_i$, value v_i was proposed by some $p \in \mathcal{D}$, and there is some $q \in \mathcal{D}_i$ that decides v_i.*
(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.[7] 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 $\mathcal{M}$.

Since excessive partitioning of the system into more than k root components makes k-set agreement trivially impossible, one natural assumption is to restrict the maximum number of root components per round in our system to k:

Assumption 1 *$\forall r > 0$, $\mathcal{G}^r$ contains at most k root components.*

Neither Asmp. 1 nor the stronger assumption given in Theorem 3 is sufficient for $1 \leqslant k < n-1$, however. To prove this, we use the generic Theorem 1 in conjunction with the impossibility of consensus in dynamic networks with a single root component that is vertex-stable only for $D-1$ subsequent rounds established in [7, Theorem 5], restated as Asmp. 2 resp. Theorem 2:

Assumption 2 (Consensus [6, Asmpt. 4]) *For any round r, there is exactly one root component $\mathcal{R}^r$ in $\mathcal{G}^r$. There are no other constraints on $\mathcal{G}^r$, except that there is some D and an interval of rounds $I = [r_{ST}, r_{ST} + D - 2]$ with an I-vertex stable root component $\mathcal{R}^I$, where there exists a unique $q \in \Pi$ with $\forall p \in \mathcal{R}, \forall r \in I : \mathtt{cd}^r(p, q) \leqslant D$, while for all $q' \in \Pi \setminus \{q\}$ we have $\forall p \in \mathcal{R}, \forall r \in I : \mathtt{cd}^r(p, q') \leqslant D - 1$.*

Theorem 2 (Consensus impossibility [6, Thm. 5]). *Assume that Assumption 2 is the only requirement for the communication graph topologies. Then consensus is impossible.*

The proof of the impossibility result in Theorem 3 utilizes the existence of a forever bivalent consensus run in a suitably chosen restricted system, which violates the termination condition of k-set agreement.

Theorem 3. *There is no algorithm that solves k-set agreement with $n > k+1$ processes under Asmp. 1, for any $1 \leqslant k < n-1$, even if there are $k-1$ root components $R_1, \ldots, R_{k-1}$ that are vertex-stable forever, and one root component R_k that is vertex-stable for not more than $n-k-1$ rounds.*

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

Proof. Suppose that there is a k-set algorithm $\mathcal{A}$ that works correctly under the assumptions of our theorem. For $k = 1$, Theorem 3 is equivalent to Theorem 2 for $D = n - k = n - 1$. To prove the theorem for $k > 1$, we will show that the conditions of the generic Theorem 1 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 dedices 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 root) 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} root component R_k, which is perpetually changing in every round, except for some interval of rounds $I = [r_{ST}, r_{ST} + \ell - 1]$, where $\ell = n - k - 1$, for some fixed r_{ST}. During this interval, let the topology of $\overline{\mathcal{D}}$ be such that there exists some $p \in R_k$ and some $q \in \overline{\mathcal{D}}$ with $\mathtt{cd}^{r_{ST}}(p, q) = n - k$. Since $|\overline{\mathcal{D}}| = n - k + 1$, such a topology (e.g. a chain with head p and tail q) adhering to the conditions of Theorem 2 for $D = n - k$ exists. 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 ρ as in ρ'. Such a run ρ 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 3 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 root 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 4 below that this is not the case for any $1 < k \leqslant n - 1$. We will again use the generic Theorem 1, this time in conjunction with the variant of the well-known impossibility of consensus with lossy links [26, 27] provided in Lem. 3, 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 3. *Let $\mathcal{M}' = \langle p, q\rangle$ 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 $r < r'$, $(p \to q) \in \mathcal{G}^r$ and/or $(q \to p) \in \mathcal{G}^r$, and no other edges incident with p or q are in $\mathcal{G}^r$, and (ii) for $r \geqslant r'$, there are no edges incident with p and q 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 [27, Lemma 3]. After r', this result continues to hold (and is even ensured by the classic lossy link impossibility [26]). Hence, consensus is indeed impossible in $\mathcal{M}'$.

Theorem 4. *Suppose that, in every run, there is a stabilization round r_{GST} such that, for all $r \geqslant r_{\text{GST}}$, it holds that $\mathcal{G}^r = \mathcal{G}^{r+1}$ and there are no other restrictions on the communication graphs apart from Asmp. 1. Then, there is no algorithm that solves k-set agreement for $1 < k < n$.*

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_{GST}, (i) the same set of $k-1$ root 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_{GST}, 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 the assumptions of our theorem, as the number of root components never exceeds k.

Moreover, we choose r_{GST} 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_{GST}, in which case they must eventually unilaterally decide, we can safely assume this.

We can now again employ the generic impossibility Theorem 1 in this modified setting. The proofs of properties (A), (B) and (D) remain essentially the same as in Theorem 3. It hence only remains to prove:
(C) Consensus is impossible in $\mathcal{M}' = \langle \overline{\mathcal{D}} \rangle$: This follows immediately from Lem. 3 with $r' = r_{\mathrm{GST}}$.

The following Theorem 5 reveals that even (considerably) less than k root components per round before stabilization and a single perpetually stable root component after stabilization are not sufficient for solving k-set agreement.

Theorem 5. *There is no algorithm that solves k-set agreement in our setting for $1 < k < n$ if the only restrictions on the communication graphs are that (A) every $\mathcal{G}^r$ contains at most $\lceil k/2 \rceil + 1$ root components, and (B) there exists a round r_{GST} such that $\mathcal{G}^r = \mathcal{G}$ for $r \geqslant r_{\mathrm{GST}}$, where $\mathcal{G}$ contains only a single root component.*

Proof. We show that, under the assumption that $\mathcal{A}$ exists, there is an adversarial strategy 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 $p, q \in \mathcal{D}_i$ 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 $(p \to q)$ or $(p \leftarrow q)$ or both are in $\mathcal{G}^r$, in a way that causes Lem. 3 to apply. Note carefully that any such $\mathcal{G}^r$ indeed satisfies assumption (A) of our theorem.

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 root after r_{GST}. 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 $p, q \in \mathcal{D}_i$ (and does not provide an incoming link from any other process, as before) in any $\mathcal{G}^r$. Since, for both p and q, this is again indistinguishable from the situation where they become the single root after r_{GST}, 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 Lem. 3 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 and q must hence decide on two *different* values. Moreover, since neither p nor q 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 and q 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 root components by two and preserving assumption (A) of the theorem, 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 5 reveals an interesting gap between 2-set agreement and 1-set agreement, i.e., consensus: It shows that 2-set agreement is impossible with $\lceil k/2 \rceil + 1 = 2$ root components per round before and a single fixed root component after stabilization. By contrast, if we reduce the number of root

components per round to a single one before stabilization (and still consider a single fixed root thereafter), even 1-set agreement becomes solvable [6].

As a final remark, we note that the results of Theorem 3 and Theorem 5 indeed implicitly define necessary network assumptions for solving k-set agreement: Every given network assumption, defined arbitrarily (even without referring to VSRCs etc.), that allows to implement k-set agreement must be such that it does not allow the adversary to choose graphs that adhere to the conditions stated in our impossibility theorems (more specifically, prohibit those executions that are used in our proofs).

4 Possibility Results

In this section, we provide network assumptions that are sufficient for k-set agreement. Although we do not claim that they are also necessary, they are weak enough to be considered close to the solvability/impossibility border.

To illustrate some of the ideas used in our network assumptions for general k-set agreement, we start with the simple case of $n-1$-set agreement (also called *set agreement*) first. Note that Theorem 3 does not apply here. To circumvent the impossibility result of Theorem 5, it suffices to strengthen the assumption of at most $n-1$ root components in every round (Asmp. 1) 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 root components consisting of a single process.

Assumption 3 (Sufficient Network Assumption for Set Agreement) *Let $R_1^{I_1}, \ldots, R_n^{I_n}$ be any set of n root components occurring in an execution, where $R_i^{I_i}$ consists of p_i only. Then, there must be two indices i, $j \neq i$ such that $R_i^{I_i}$* influences *$R_j^{I_j}$, denoted $R_i^{I_i} \hookrightarrow R_j^{I_j}$, in the sense that there exists a causal chain starting after I_i that ends before or at the beginning of I_j.*

It is easy to devise a set agreement algorithm that works correctly in a dynamic network that satisfies Asmp. 3, provided (a bound on) n is known: In Alg. 1, 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.

Algorithm 1 Set agreement algorithm for Asmp. 3.

```
    Set agreement algorithm, code for process p_i:
 1: v_i := x_i ∈ V // initial value
 2: y_i := ⊥
    Emit round r messages:
 3: send ⟨v_i, y_i⟩ to all
    Receive round r messages:
 4: receive ⟨v_j, y_j⟩ from all current neighbors
    Round r: computation:
 5: v_i := max{v_i, v_j : j ∈ N_{p_i}}
 6: if ∃j : (y_j ≠ ⊥) ∧ (y_i = ⊥) then
 7:     y_i := y_j
 8: if (N_{p_i} = ∅) ∧ (y_i = ⊥) then
 9:     y_i := v_i
10: if (r = n) ∧ (y_i = ⊥) then
11:     y_i := v_i; terminate
```

Theorem 6 (Correctness Alg. 1). *Alg. 1 solves $n-1$-set agreement in a dynamic network that satisfies Asmp. 3.*

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 $x_1 \leqslant x_2 \leqslant \ldots$, and let S^k be the set of different values (in y_i or, if still $y_i = \perp$, in v_i) present in the system at the

beginning of round $k \geqslant 1$; $S^1 = \{x_1, \ldots, x_n\}$ is the set of initial values. Obviously, $S^1 \supseteq S^2 \supseteq \ldots$, and since $n-1$-agreement is fulfilled if $|S^{n+1}| < n$, 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, $x_1 \notin S^2$ 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_ℓ 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 $x_i \notin S^{k+1}$ 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_ℓ's decision value x_ℓ and hence also cause $x_i \notin S^{i+1}$. 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: Asmp. 3 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 root 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.

Whereas the above solution is simple, it is apparent that Asmp. 3 is quite demanding. In particular, it forbids rounds where no messages are received and requires explicit knowledge of (a bound on) n. We now provide two other conditions (Asmps. 4 and 5), the conjunction of which will be sufficient for general k-set agreement. They are obtained by adding some additional properties to the necessary network conditions implied by our impossibility Theorems 3 and 5.[8]

To avoid non-terminating (i.e., forever undecided) executions as predicted by Theorem 3, we require the following *stable interval* constraint to hold:

Assumption 4 (Stable interval) *Let $H, D > 0$ be given. In every run, all vertex-stable root components R^I with $|I| \geqslant D+1$ are D-bounded. Moreover, there is some $k \geqslant 1$ and some interval $I = [r_{GST}, r_{GST}+d]$ with $d \geqslant 3D+H$, where $\ell \leqslant k$ H-network-bounded vertex-stable root components $R^I_1, \ldots, R^I_\ell$ exist simultaneously.*

The parameter D, which can always be safely set to $D = n-1$ by [6, Lemma 2], allows to adapt the model to the actual connectivity guaranteed in the VSRCs of a given dynamic network. Note that, since $D > 0$, rounds where no message is received are not forbidden here (in contrast to Asmp. 3).

In order to also circumvent executions violating the k-agreement property established by Theorem 5, we introduce the *majority influence* constraint given in Asmp. 5 below. Like Asmp. 3 for set agreement, it guarantees some (minimal) information flow between sufficiently long-lasting vertex-stable root components that exist at different times. As visualized in Fig. 1, it implies that the information available in any such VSRC originates in at most k "initial" VSRCs.

Formally, given some run ρ, we denote by $\mathbb{V}_d$ the set of all root components that are vertex-stable for at least d consecutive rounds in ρ. Let $R_{cur} \in \mathbb{V}_1$ be vertex-stable in $I_{cur} = [r_{cur}, s_{cur}]$ and $R_{suc} \in \mathbb{V}_1$ be vertex-stable in $I_{suc} = [r_{suc}, s_{suc}]$ with $r_{suc} > s_{cur}$; note that $\mathbb{V}_d \subseteq \mathbb{V}_1$ for every $d \geqslant 1$.

[8] An alternative way to derive sufficient network assumptions for, e.g., $n-2$-set agreement could be to generalize Asmp. 3: One could e.g. assume that at least two out of every set of $n-1$ different root components consisting of 1 or 2 processes are influenced by a common predecessor root component. Whereas this assumption does not require vertex stability of root components, it effectively ensures that information propagates not slower as in VSRCs. Owing to this fact, it also prohibits the existence of the node q in Asmp. 2 with causal distance D from p in the root component, thereby causing the proof of Theorem 3 to fail. Working out the details may turn out difficult, though: After all, unlike single-process roots, larger root components suffer from the problem that its members cannot always determine whether the root was a VSRC or not. Influence must hence be conservative, in the sense that it involves even *potential* 2-process roots.

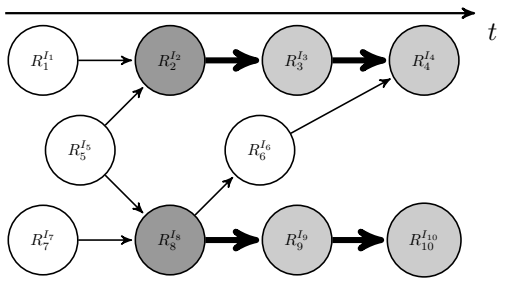

Fig. 1: VSRCs influencing each other in a run, for $k = 2$. Time progresses from left to right; all shaded nodes are stable for more than $2D$ rounds, white nodes are stable between $D + 1$ and $2D$ rounds. Thick arrows represent majority influence, thin arrows represent (weak) influence. At most two shaded nodes, depicted darkly shaded, may exist that are not majority-influenced by another shaded node.

Definition 4 ((Weak) Influence). *Given any two $R_{cur}^{I_{cur}}, R_{suc}^{I_{suc}} \in \mathbb{V}_1$, we say that some process $p \in R_{cur}^{I_{cur}}$* influences *some process $q \in R_{suc}^{I_{suc}}$ and write $p \hookrightarrow q$ with $\hookrightarrow \subseteq \Pi^2$ iff there exists a causal chain from p to q starting after I_{cur} that ends before or at the beginning of I_{suc}, i.e., $cd^{s_{cur}+1}(p, q) \leqslant r_{suc} - s_{cur}$. In this case, we also say that $R_{cur}^{I_{cur}}$ (weakly) influences $R_{suc}^{I_{suc}}$ and write $R_{cur}^{I_{cur}} \hookrightarrow R_{suc}^{I_{suc}}$, using the relation $\hookrightarrow \subseteq \mathbb{V}_1^2$ here.*

We will also need stronger notions of influence, which are based on the following Def. 5:

Definition 5 (Influence Sets). *Given any two $R_{cur}, R_{suc} \in \mathbb{V}_1$, their influence set is $IS(R_{cur}, R_{suc}) := \{q \in R_{suc} \mid \exists p \in R_{cur} \colon p \hookrightarrow q\}$.*

The *majority influence* between the nodes in R_{cur} and R_{suc} guarantees that R_{cur} influences a set of nodes in R_{suc}, which is greater than any set influenced by VSRCs not already known by the processes in R_{cur} (and greater than or equal to any set influenced by VSRCs already known by the processes in R_{cur}). Majority influence is hence a very natural way to discriminate between strong and weak influence between VSRCs, see Def. 7 below.

Definition 6 (Majority influence). *We say that a VSRC $R_{cur} \in \mathbb{V}_{2D+1}$ exercises a* majority influence *on a VSRC $R_{suc} \in \mathbb{V}_{2D+1}$, denoted $R_{cur} \hookrightarrow_{\mathtt{m}} R_{suc}$ with $\hookrightarrow_{\mathtt{m}} \subseteq \mathbb{V}_{2D+1}^2$, iff $\forall R \in \mathbb{V}_{D+1}$ with $IS(R, R_{cur}) = \emptyset$ it holds that $|IS(R_{cur}, R_{suc})| > |IS(R, R_{suc})|$ and $\forall R \in \mathbb{V}_{D+1}$ with $IS(R, R_{cur}) \neq \emptyset$ it holds that $|IS(R_{cur}, R_{suc})| \geqslant |IS(R, R_{suc})|$.*

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

Lemma 4 (Properties $\hookrightarrow_{\mathtt{m}}$). *The majority influence relation is antisymmetric, acyclic and intransitive.*

Proof. Let R, $\overline{R}$, and $\hat{R}$ be three different VSRCs stable in the intervals I, $\overline{I}$, and $\hat{I}$, resp. Since the VSRCs R and $\overline{R} \neq R$ are ordered in time according to their round intervals I and $\overline{I}$, which must be disjoint, no process in R can be influenced by any process in $\overline{R}$ if $R \hookrightarrow_{\mathtt{m}} \overline{R}$. Hence, $\overline{R} \hookrightarrow_{\mathtt{m}} R$ cannot hold, which implies both antisymmetry and, by a transitive application of this argument, acyclicity. To prove intransitivity, observe that $R \hookrightarrow_{\mathtt{m}} \overline{R}$ and $\overline{R} \hookrightarrow_{\mathtt{m}} \hat{R}$ would imply $\mathrm{IS}(R, \hat{R}) > \mathrm{IS}(\overline{R}, \hat{R})$ if $R \hookrightarrow_{\mathtt{m}} \hat{R}$ also held, since no process in R can be influenced by any process in $\overline{R}$. This contradicts $\mathrm{IS}(\overline{R}, \hat{R}) \geqslant \mathrm{IS}(R, \hat{R})$ required by $\overline{R} \hookrightarrow_{\mathtt{m}} \hat{R}$, however.

Definition 7 (Strong Influence). *We say that $R_{cur} \in \mathbb{V}_{2D+1}$* strongly influences *$R_{suc} \in \mathbb{V}_{2D+1}$ and write $R_{cur} \hookrightarrow_{\mathtt{m}} * R_{suc}$, where $\hookrightarrow_{\mathtt{m}} * \subseteq \mathbb{V}_{2D+1}^2$ is the transitive closure of $\hookrightarrow_{\mathtt{m}}$.*

Note carefully that $\hookrightarrow_{\mathtt{m}} *$ is antisymmetric by Lem. 4.

In addition to Asmp. 4, we require the following Asmp. 5 to hold in every execution. It just enhances the very limited information propagation that could occur in our model solely under Asmp. 4.

Assumption 5 (k-majority influence) *$\exists K \subseteq \mathbb{V}_{2D+1}$ with $|K| \leqslant k$ s.t. $\forall \overline{R} \in \mathbb{V}_{2D+1} \setminus K$ $\exists R \in \mathbb{V}_{2D+1}$ with $R \hookrightarrow_{\mathtt{m}} \overline{R}$.*

Informally speaking, Asmp. 5 ensures that all but at most k "initial" VSRCs in $\mathbb{V}_{2D+1}$ are majority-influenced by some earlier VSRC in $\mathbb{V}_{2D+1}$ (see Fig. 1). Note carefully, though, that Asmp. 5 neither prohibits partitioning of the system in more than k simultaneous VSRCs nor directly exhibits a k-quorum property, cf. the well-known quorum failure detector Σ_k [9] that is known to be necessary for k-set agreement: After

all, one could e.g. choose $k+1=3$ VSRC's $R_2^{I_2}$, $R_4^{I_4}$ and $R_7^{I_7}$ in Fig. 1 without finding any pair among those which are majority-influenced by a common predecessor VSRC!

However, if majority influence was replaced by strong influence according to Def. 7, such a quorum property could be easily established: Starting out from an arbitrary set of $k+1$ $2D+1$-VSRCs, we could go back along the (acyclic) majority influence relation until we end up in the set K guaranteed by Asmp. 5. If a k-set agreement algorithm relied on $2D+1$-VSRCs for decisions, this would guarantee that no more than k decision values (possibly fabricated in the "initial" $2D+1$-VSRCs) can be produced. Note that considering an alternative to Asmp. 5 that just ensures a k-quorum would not be sufficient here, unless acyclicity can be ensured as well. In any case, however, this indicates that our network assumptions are indeed fairly close to the k-set agreement solvability border.

5 Advanced Algorithms

In this section, we provide a k-set agreement algorithm and prove that it works correctly under the conjunction of Asmps. 4 and 5. Note that it needs to know D, but neither n nor H. The algorithm consists of a "generic" k-set agreement algorithm, which relies on a simple network approximation algorithm for locally detecting vertex-stable root 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 the above network assumptions.

In sharp contrast to classic k-set agreement algorithms, the algorithm is *k-uniform*, 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) $2D+1$-VSRCs occurring in the particular run. As a consequence, if the network partitions into k weakly connected components, for example,[9] 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. 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.

Our algorithm (see Alg. 2 for the pseudo-code) consists of two parts, a network approximation algorithm and the k-set agreement core algorithm. 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. Note carefully 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.

5.1 Network Approximation Algorithm

We use a marginally modified version[10] of the network approximation algorithm used for consensus in [6, 7], stated in the upper part of Alg. 2 for completeness. In every round, process p broadcasts its current *network estimate* A_p and fuses it with the network estimates received from its neighbors. A_p 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 received evidence that $(p_i \to p_j)$ was present. Given A_p, we use $A_p|t$ with $0 < t \leqslant r$ to denote the current estimate of $\mathcal{G}^t$ contained in A_p.

For ease of reference, we restate the major properties of the network approximation algorithm used in Alg. 2 for maintaining A_p from [7].

Initially, A_p consists of process p only, and is updated with new information in every round. As the information about q's neighbors in $\mathcal{G}^t$ might take many rounds to reach some process p (if it ever arrives at

[9] 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 root components, but k root components do not imply a partitioning of the communication graph into k weakly connected components — one process may still be connected to several components.

[10] Rather than returning TRUE, `InStableRoot()` now returns the processes in the detected SCC, and rather than returning FALSE it returns $\emptyset$.

p), however, $A_p|t$ will never be fully up-to-date, and as only reported edges are added to the approximation (but not all reports need to reach p), $A_p|t$ will be an under-approximation of $\mathcal{G}^t$. Consequently, calling `InStableRoot()` may return $\emptyset$ although p has been in an I-vertex stable root component, which happens when p has not received enough information about $\mathcal{G}^t$ in some round $t \in I$. Since all VSRCs that are relevant for our k-set agreement algorithm are D-bounded, however, p will learn that it has been in such a root component in round t by the end of round $t + D$, see Corollary 2 below.

Lemma 5 shows that $A_p|t$ underapproximates $\mathcal{G}^t$ in a way that consistently includes neighborhoods. Its proof uses a simple invariant, asserting $A_p|t = \langle\{p\}, \emptyset\rangle$ at the end of every round $r < t$.

Lemma 5 ([7, Lem. 6]). *If $A_p|t$ contains $(v \to w)$ at the end of some round r, then (i) $(v \to w) \in \mathcal{G}^t$, i.e., $A_p|t \subseteq \mathcal{G}^t$, and (ii) $A_p|t$ also contains $(v' \to w)$ for every $v' \in \mathcal{N}_w^t \subseteq \mathcal{G}^t$.*

Proof. We first consider the case where $r < t$, then at the end of round r $A_p|t$ is empty, i.e., there are no edges in $A_p|t$. 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 $A_p|t$ contains $(v \to w)$ at the end of round $r = t$, it follows from $A_q|t = \langle\{q\}, \emptyset\rangle$ at the end of every round $r < t$, for every $q \in \Pi$, that $w = p$, since p is the only processor that can have added this edge to its graph approximation. Clearly, it did so only when $v \in \mathcal{N}_p^t$, i.e., $(v \to w) \in \mathcal{G}^t$, and included also $(v' \to w)$ for every $v' \in \mathcal{N}_p^t$ 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 $A_q|t$ at the end of round r, in particular, for every $q \in \mathcal{N}_p^{r+1}$. If indeed $(v \to w)$ in $A_p|t$ at the end of round $r+1$, it must be contained in the union of round r approximations

$$U = (A_p|t) \cup \left(\bigcup_{q \in \mathcal{N}_p^{r+1}} A_q|t \right)$$

and hence in some $A_x|t$ ($x = q$ or $x = p$) at the end of round r. Note that the edges (labeled $r + 1$) added in round $r + 1$ to A_p are irrelevant for $A_p|t$ here, since $t < r + 1$.

Consequently, by the induction hypothesis, $(v \to w) \in \mathcal{G}^t$, thereby confirming (i). As for (ii), the induction hypothesis also implies that $(v' \to w)$ is also in this $A_x|t$. Hence, every such edge must be in U and hence in $A_p|t$ at the end of round $r + 1$ as asserted.

The next lemma shows that locally detecting $A_p|t$ to be strongly connected (in Line 14 of the network approximation of Alg. 2) implies that p is in the root component of round s. This result rests on the fact that $A_p|t$ underapproximates $\mathcal{G}^s$ (Lemma 5.(i)), but does so in a way that never omits an in-edge at any process $q \in V(A_p|t)$ (Lemma 5.(ii)).

Lemma 6. *If the graph $C_p|t$ (line 14) with $t < r$ is non-empty in round r, then p is member of $\mathcal{R}^t$, i.e., $p \in R$.*

Proof. For a contradiction, assume that $C_p|t$ is non-empty (hence $A_p|t$ is an SCC by Line 14), but $p \notin \mathcal{R}$. Since p is always included in any A_p by construction and $A_p|t$ underapproximates $\mathcal{G}^t$ by Lemma 5.(i), this implies that $A_p|t$ cannot be the root component of $\mathcal{G}^t$. Rather, $A_p|t$ must contain some process w that has an in-edge $(v \to w)$ in $\mathcal{G}^t$ that is not present in $A_p|t$. As w and hence some edge $(q \xrightarrow{t} w)$ is contained in $A_p|t$, because it is an SCC, Lemma 5.(ii) reveals that this is impossible.

From the description of `InStableRoot`(I) in Alg. 2, it is not too difficult to establish the following Corollary 1. It rests on the fact that $A_p|t$ underapproximates $\mathcal{G}^t$ but does not selectively omit in-edges of any process $q \in A_p|t$ ((Lem. 5).

Corollary 1 (cf. [7, Cor. 1]). *If the predicate* `InStableRoot`(I) *evaluates to $R \neq \emptyset$ at process p in round r, then $\forall t \in I$ where $t < r$, it holds that p is a member of R^t, i.e., $p \in R$.*

The following Corollary 2 reveals that, in a sufficiently long interval of rounds $I = [r, s]$ with an I-vertex-stable root component R^I, every member $p \in R$ detects its membership in the $[r, s - D]$-VSRC R^I with a latency of at most D rounds (i.e., at the end of round $r + D$).

Corollary 2 ([7, Cor. 2]). *Consider an interval of rounds $I = [r, s]$, with $|I| = s - r + 1 > D$, such that there is a D-bounded vertex-stable root component R^I. Then, from the end of round s on, a call to* `InStableRoot`$([r, s - D])$ *returns R at every process in R.*

Together, Corollaries 1 and 2 reveal that `InStableRoot`(.) *precisely* characterizes the caller's actual membership in the $[r, s - D]$-VSRC R^I in the communication graphs from the end of round s on.

5.2 Core k-set Agreement Algorithm

Algorithm 2 Our k-uniform k-set agreement algorithm, code for process p_i

NETWORK APPROXIMATION ALGORITHM:

1: $A_{p_i} := \langle V_{p_i}, E_{p_i} \rangle$ initially $(\{p_i\}, \emptyset)$
// weighted digraph without multi-edges and loops
Emit round r messages:
2: send $\langle A_{p_i} \rangle$ to all current neighbors
Round r: computation:
3: **for** $q \in \mathcal{N}^r_{p_i}$ and q sent message $\langle A_q \rangle$ in r **do**
4: **if** $\exists$ edge $e = (q \xrightarrow{T} p_i) \in E_{p_i}$ **then**
5: replace e with $(q \xrightarrow{T'} p_i)$ in E_{p_i} where $T' \leftarrow T \cup \{r\}$
6: **else**
7: add $e := (q \xrightarrow{\{r\}} p_i)$ to E_{p_i}
8: $V_{p_i} \leftarrow V_{p_i} \cup V_q$
9: **for** every pair of nodes $(p_u, p_v) \in V_{p_i} \times V_{p_i}$, $p_u \neq p_v$ **do**
10: **if** $T' = \bigcup \left\{ S \mid \exists q \in \mathcal{N}^r_{p_i} : (p_u \xrightarrow{S} p_v) \in E_q \right\} \neq \emptyset$ **then**
11: replace $(p_u \xrightarrow{T} p_v)$ in E_{p_i} with $(p_u \xrightarrow{T \cup T'} p_v)$;
add $(p_u \xrightarrow{T'} p_v)$ if no such edge exists

12: **function** `InStableRoot`(I)
13: Let $A_{p_i}|t$ be induced graph of $\left\{ (p_u \xrightarrow{T} p_v) \in E_{p_i} \mid t \in T \right\}$
14: Let $C_{p_i}|t$ be $A_{p_i}|t$ if it is strongly connected, or the empty graph otherwise.
15: **if** $\forall t_1, t_2 \in I : C_{p_i} := V(C_{p_i}|t_1) = V(C_{p_i}|t_2) \neq \emptyset$ **then**
16: return C_{p_i}
17: **else**
18: return $\emptyset$

CORE K-SET AGREEMENT ALGORITHM:

Variables and Initialization:
1: $\texttt{hist}_i[*][*] := \emptyset$ /* $\texttt{hist}_i[j][r]$ holds p_i's estimate of the locks learned by p_j in round r */
2: $\texttt{hist}_i[i][0] := \{(\{p_i\}, x_i, 0)\}$ /* virtual first lock $(V(R) := \{p_i\}, v := x_i, \tau_{\texttt{create}} := 0)$ at p_i */
3: $\ell := \perp$ // most recent lock round, $\perp$ if none
4: $\texttt{decision}_i := \perp$ // p_i's decision, $\perp$ if undecided

Emit round r messages:
5: send $\langle \texttt{hist}_i, \texttt{decision}_i \rangle$ to all neighbors
Receive round r messages:
6: for all p_j in p_i's neighborhood $\mathcal{N}^r_{p_i}$, receive $\langle \texttt{hist}_j, \texttt{decision}_j \rangle$
Round r computation:
7: **if** $\texttt{decision}_i = \perp$ **then**
8: **if** received any message m containing $m.\texttt{decision} \neq \perp$ **then**
9: decide $m.\texttt{decision}$ and set $\texttt{decision}_i := m.\texttt{decision}$
10: **else**
// update $\texttt{hist}_i$ with $\texttt{hist}_j$ received from neighbors
11: **for** $p_j \in \mathcal{N}^r_{p_i}$, where p_j sent $\texttt{hist}_j$ **do**
12: $\texttt{hist}'_i := \texttt{hist}_i$ // remember current history
13: **for** all non-empty entries $\texttt{hist}_j[x][r']$ of $\texttt{hist}_j$, $x \neq i$ **do**
14: $\texttt{hist}_i[x][r'] := \texttt{hist}_i[x][r'] \cup \texttt{hist}_j[x][r']$
// locally add all newly learned locks:
15: $\texttt{hist}_i[i] := \texttt{hist}_i \setminus \texttt{hist}'_i$
// perform state transitions (undecided, locked, decided):
16: $\texttt{myRoot} := \texttt{InStableRoot}(r - 2D, r - D)$
17: **if** $\ell = \perp$ and $\texttt{myRoot} \neq \emptyset$ **then**
18: $\ell := r - 2D$
19: $\texttt{lock} := \texttt{GetLock}(\texttt{myRoot}, \ell)$
20: $\texttt{hist}_i[i][r] := \texttt{hist}_i[i][r] \cup \texttt{lock}$ // create new lock
21: **else if** $\ell \neq \perp$ and $\texttt{myRoot} = \emptyset$ **then**
22: $\ell := \perp$ // release unsuccessful lock
23: **else if** $\ell \neq \perp$ and $\texttt{InStableRoot}[\ell, \ell + 2D] \neq \emptyset$ **then**
24: decide $\texttt{lock}.v$ and set $\texttt{decision}_i := \texttt{lock}.v$

25: **function** `GetLock`(R, r')
26: Let S be the multiset $\bigcup_{p_j \in R, r'' \leqslant r'} \texttt{hist}_i[j][r'']$
Let mfrq(S) be the set of the most frequent elements in S
27: Let $\text{mfrq}_{\texttt{latest}}(S) := \{x \in \text{mfrq}(S) \mid \forall y \neq x \in \text{mfrq}(S) \colon x.\tau_{\texttt{create}} > y.\tau_{\texttt{create}}\}$
28: **if** $|\text{mfrq}_{\texttt{latest}}(S)| = 1$ **then**
29: Let v be $s.v$ of the single element $s \in \text{mfrq}_{\texttt{latest}}(S)$
30: $\texttt{newLock} := (R, v, r)$
31: **else**
32: $\texttt{newLock} := (R, \max_{s \in S} \{s.v\}, r)$ // deterministic choice
33: **return** newLock

The general idea of our core k-set agreement algorithm, presented in the bottom/right part of Alg. 2, is to generate new decision values only at members of $2D + 1$-VSRCs, and to disseminate those values throughout the remaining network. Note that it needs to know D, but not H (nor n). Using the network approximation A_{p_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 $D + 1$-VSRC D rounds in the past (line 17). Note that the latency of D rounds is inevitable here, since information propagation within a $D + 1$-VSRC may take up to D rounds due to D-boundedness. 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* VSRC members (which occurs when the $D + 1$-VSRC has actually extended to a $2D + 1$-VSRC), 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).

Since locking is done optimistically, however, it may also happen that the $D+1$-VSRC does not extend to a $2D+1$-VSRC (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: Meachanisms are required that, despite possibly inconsistently perceived unsuccessful locks, ensure both (a) an *identical* decision value among all members of a $2D+1$-VSRC who successfully detect this $2D+1$-VSRC and thus reach the decided state, and (b) no more than k different decision values originating from different $2D+1$-VSRCs.

Both goals are accomplished by a particular selection of the decision values (using function `GetLock`), which ultimately relies on an intricate utilization of our network assumptions Asmps. 4 and 5: 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* $L = (R, v, \tau_{\texttt{create}})$, which are created by every process that enters the locked state: R is the vertex-set of the detected $D+1$-VSRC, v is a certain proposal value (determined as explained below), and $\tau_{\texttt{create}}$ is the round when the lock is created.

In more detail, the lock history at process p_i consists of an array $\texttt{hist}_i[j][r]$ 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 $\texttt{hist}_i[i][r]$.

(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 $\texttt{hist}_j[x][r']$ received from some p_j later on. In this case, L is just added to $\texttt{hist}_i[x][r']$ (line 14).

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

Note that the update rules (i)+(ii) resemble the ones of vector clocks [23].

Clearly, $\texttt{hist}_i[i][r']$ will always be accurate for current and past rounds $r' \leqslant r$, while $\texttt{hist}_i[j][r']$ may not always be up-to date, i.e., may lack some locks that are present in $\texttt{hist}_j[j][r']$. Nevertheless, if p_i and p_j are members of the same $2D+1$-VSRC R^I with $I = [r-2D, r]$, Def. 2 ensures that p_i and p_j have consistent histories $\texttt{hist}_i[j][r']$ and $\texttt{hist}_j[i][r']$ 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 $D+1$-VSRC 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 $D+1$-VSRC in fact extended to a $2D+1$-VSRC.

The resulting consistency of the histories is finally exploited by the function $\texttt{GetLock}(R, \ell)$, 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 R of the detected $D+1$-VSRC and its starting round $\ell = r-2D$. `GetLock` first determines a multiset S, which contains all locks locally known to the members $p_j \in R$ by round $r-2D$ (line 26). Note that the multiplicity of some lock $L = (R', v, r')$ in S is just the number of members of R who got to know L by round $r-2D$, which is just $|\mathrm{IS}(R', R)|$ according to Def. 5. In order to determine a proper value for the new lock to be computed by `GetLock`, we exploit majority influence according to Def. 6: If the set $\mathrm{mfrq}_{\texttt{latest}}(S)$, containing the most frequent locks in S 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 $2D+1$-VSRCs, see (1) below. Otherwise, i.e., if $\mathrm{mfrq}_{\texttt{latest}}(S)$ contains multiple candidate locks, a consistent deterministic choice, namely, the maximum among all lock values in S, is used (line 32).

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 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 $\texttt{hist}_i$. 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 $\text{mfrq}_{\texttt{latest}}(S)$ in line 28, defeats the inevitable "amplification" of the number of processes that got to know some "old" lock: All members of a $2D+1$-VSRC have finally learned *all* "old" locks that were only known to *some* of its members at the starting round of the VSRC initially. In terms of multiplicity in S, 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 $D+1$-VSRC, it might seem sufficient to simply update the creation time of an old lock that (dominantly) influences a newly detected VSRC. 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 VSRCs with vertex sets $R_1 = \{p_1, p_2\}$ and $R_3 = \{p_1, p_2\}$ that are both stable for $D+1$ rounds and two root components $R_2 = \{p_1, p_3\}$ and $R_4 = \{p_1, p_3\}$ that are stable for $2D+1$ rounds are formed. Let these VSRCs be such that R_i is formed before R_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 VSRC. First, let the processes of R_1 lock on some old lock L'. Then, assume that the processes of R_2 lock on some lock[11] $L \neq L'$, a lock not known in R_1. Since $R_3 = \{p_1, p_2\}$, if R_3 is sufficiently well connected, p_1 might lock on L' in R_3, because L' is known to both p_1 and p_2 while L is known merely to p_1 at the start of R_3. Subsequently, this results in the situation in R_4 where there is neither a clear majority (L' and L are known to both members of R_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 R_4, it is not clear whether to lock on $L.v$ or on $L'.v$. Nevertheless, the processes of R_4 should be able to determine that they must lock on L and not on L', since $R_2 \hookrightarrow_{\mathtt{m}} R_4$ holds in our example: $|\text{IS}(R_1, R_2)| = 1, |\text{IS}(R_1, R_4)| = 2, |\text{IS}(R_2, R_4)| = 2$ and $|\text{IS}(R_3, R_4)| = 1$. We can therefore conclude that merely adopting old locks is insufficient.

(3) Since the stabilization round r_{GST}, as implied by Asmp. 4, may be delayed arbitrarily, an unbounded number of $2D+1$-VSRCs can occur before r_{GST}. Since any of those might produce a critical lock, in the sense of exercising a majority influence upon some later $2D+1$-VSRC, no such lock can safely be deleted from $\texttt{hist}_i$ of any p_i after bounded time.

5.3 Correctness Proof

In this subsection, we will prove the following Theorem 7:

Theorem 7. *Alg. 2 solves k-uniform k-set agreement in a dynamic network that adheres to the conjunction of Asmp. 4 and Asmp. 5.*

The proof consists of a sequence of technical lemmas, which will finally allow us to establish all the properties of k-set agreement given in Section 2. First, *Validity* 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) and later on always have values of previous locks assigned to them (lines 30 and 32). Note that the claimed k-uniformity is obvious, as the code of the algorithm does not involve k.

To establish *Termination*, we start with some simple properties related to setting locks at all members of vertex stable root components.

Lemma 7. *Apart from processes adopting a decision sent by another process, only processes part of a vertex stable root 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 `InStableRoot` detects a stable member set R in some interval I of length $D+1$ (resp. of length $2D+1$) or larger, which implies by Corollary 1 that R^I is indeed a $D+1$-VSRC (resp. $2D+1$-VSRC).

Lemma 8. *All processes part of a vertex stable root $R^{[r,s]}$ with interval length greater than $2D$, which did not start already before r, lock, i.e. set $\ell := r$, in round $r + 2D$.*

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

Proof. Because $R^{[r,s]}$ is D-bounded by Asmp. 4, Corollary 2 guarantees that $\texttt{InStableRoot}(r, r+D)$ returns R from round $r+2D$ (of the k-set-algorithm) on, and that it cannot have done so already in round $r+2D-1$. Hence, $\ell = \perp$ in round $r+2D$, the if-statement in line 17 is entered and $\ell := r$ is set in line 19.

Lemma 9. *All processes part of a vertex stable root $R^{[r,s]}$ with interval length greater than $3D$, which did not start already before r, have decided by round $r+3D$.*

Proof. It follows from Lem. 8 that all members of the VSRC $R^{[r,s]}$ set $\ell := r$ in round $r+2D$. As the VSRC remains stable also in rounds $r+2D, \dots, r+3D$, line 22 will not be executed in these rounds, thus $\ell = r$ remains unchanged. Consequently, due to Corollary 2, the if-statement in line 23 will evaluate to true at the latest in round $\ell + 3D = r + 3D$, causing all the processes to decide via line 24 by round $r+3D$ as asserted.

Lemma 10. *The algorithm eventually terminates at all processes.*

Proof. For a contradiction, assume that there is $p_j \in \Pi$ which has not terminated after the stable interval guaranteed by Asmp. 4. This implies that p_j is not part of a root component during this stable interval, because Lem. 9 ensures termination by $r_{GST} + 3D$ at the latest for the latter. Hence, p_j did not get a decide message either. From Def. 3, it follows that there exists a causal chain of length at most H to p_j from some member p_i of a VSRC after its termination. Therefore, it must receive the decide message by $r_{GST} + 3D + H$ at latest.

Although we now know that all members of a VSRC 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 11. *Given some VSRC R^I with $I = [r, s]$ and $s \geqslant r + D$, in all rounds $x \in [r + D, s]$ it holds that $\forall p_i, p_j \in R\colon \bigcup_{r' \leqslant r} \texttt{hist}_i[j][r'] = \bigcup_{r' \leqslant r} \texttt{hist}_j[j][r']$*

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

Lemma 12. *All processes of a VSRCs R^I of $\mathbb{V}_{2D+1}$ with $I = [r, s]$ adopt the same lock (and hence decide the same).*

Proof. Such a lock is created by $p_i \in R$ in round $r+2D$, when it recognizes R^I as having been vertex-stable for $D+1$ rounds according to Lem. 8. As the lock (value) is computed based on $\texttt{hist}_i$ present in round $r+2D$, which is consistent among all VSRC members by Lem. 11, the lemma follows.

Finally, we show that, given that the system satisfies Asmp. 5, there will be at most k decision values in any run of Alg. 2. Since there are at most k VSRCs of $\mathbb{V}_{2D+1}$ that are not majority-influenced by other VSRCs, it remains to show that any majority-influenced VSRC decides the same as the VSRC it is majority-influenced by. In order to do so, we will first establish a key property of our central data structure $\texttt{hist}_i$.

Lemma 13. *Given $R_{cur}^{I_{cur}=[r_{cur},s_{cur}]}$, $R_{suc}^{I_{suc}=[r_{suc},s_{suc}]}$ with $|I_{cur}| > 2D$ and any $|I_{suc}| \geqslant 1$. Let L be a lock known to all members of R_{cur} by s_{cur}, i.e., for all $p_i \in R_{cur}$ it holds that, by the end of round s_{cur}, $L \in \bigcup_{r' \leqslant s_{cur}} \texttt{hist}_i[i][r']$. For any process $p_j \in R_{suc}$, it holds that if there exists some $p_i \in R_{cur}$, s.t. $p_i \hookrightarrow p_j$, then $L \in \bigcup_{r' \leqslant r_{suc}} \texttt{hist}_j[j][r']$.*

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

Lemma 14. *Given $R_{cur}^{I_{cur}=[r_{cur},s_{cur}]} \in \mathbb{V}_{2D+1}$ and $R_{suc}^{I_{suc}=[r_{suc},s_{suc}]} \in \mathbb{V}_{2D+1}$, assume that the processes of R_{cur} created the (same) lock L when locking. If $R_{cur}^{I_{cur}} \hookrightarrow_{\mathtt{m}} R_{suc}^{I_{suc}}$, then the processes of R_{suc} will choose a lock L' where $L.v = L'.v$ (and hence decide the same as the processes of R_{cur}).*

Proof. From the definition of $\hookrightarrow_{\mathtt{m}}$ (Def. 6), it follows that no VSRC R^I of $\mathbb{V}_{D+1}$ has a larger influence set on R_{suc} than R_{cur}. By Lem. 7, this implies that no lock that was generated by some R^I in $\mathbb{V}_{D+1}$ can be known to more members of R_{suc} than the lock L generated by R_{cur}. Since process p_i puts only newly learned locks into $\mathtt{hist}_i$ (lines 15 and 20), by Lem. 13, this means that in round r_{suc} no "bad" lock L_b is present in more elements of $S = \bigcup_{p_i \in R_{suc}, r' \leqslant r_{suc}} \mathtt{hist}_i[i][r']$ than L. We now show that $L.\tau_{\mathtt{create}} > L_b.\tau_{\mathtt{create}}$ for all L_b occuring in as many elements of S as L with $L_b \neq L$. Obviously, the only locks L_b that could occur in as many elements of S as L are locks that have been in $\mathtt{hist}_i$ of some $p_i \in R_{cur}$ at the beginning of round r_{cur} 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 $r_{suc}+2D$, at all processes p_i, p_j of R_{suc}, Lem. 11 implies that $\bigcup_{r' \leqslant r_{suc}} \mathtt{hist}_i[j][r'] = \bigcup_{r' \leqslant r_{suc}} \mathtt{hist}_j[i][r']$, when locking in round $r_{suc}+2D$ according to Lem. 8, every p_i of R_{suc} will find L as the unique most common lock in the elements of S 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.

6 Conclusions

In this paper, we made a significant step towards determining the solvability/impossibility border of k-set agreement in synchronous directed dynamic networks. We provided impossibility results, which imply a set of network assumptions that are necessary but not sufficient. By slightly strengthening the latter, we arrived at network assumptions that allow to implement a consensus algorithm that gracefully degrades to k-set agreement in case of non-favorable conditions.

References

1. Y. Afek and E. Gafni. Asynchrony from synchrony. In D. Frey, M. Raynal, S. Sarkar, R. Shyamasundar, and P. Sinha, editors, *Distributed Computing and Networking*, volume 7730 of *Lecture Notes in Computer Science*, pages 225–239. Springer Berlin Heidelberg, 2013.
2. M. K. Aguilera, W. Chen, and S. Toueg. Using the heartbeat failure detector for quiescent reliable communication and consensus in partitionable networks. *Theoretical Computer Science*, 220(1):3–30, June 1999.
3. J. Augustine, G. Pandurangan, P. Robinson, and 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, pages 551–569. SIAM, 2012.
4. M. Biely, P. Robinson, and 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), Springer LNCS 7109*, pages 299–312, 2011.
5. M. Biely, P. Robinson, and U. Schmid. Solving k-set agreement with stable skeleton graphs. In T. Kikuno and T. Tsuchiya, editors, *IPDPS Workshops*, pages 1488–1495. IEEE, 2011.
6. M. Biely, P. Robinson, and U. Schmid. Agreement in directed dynamic networks. In *Proceedings 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO'12)*, LNCS 7355, pages 73–84. Springer-Verlag, 2012.
7. M. Biely, P. Robinson, and U. Schmid. Agreement in directed dynamic networks. *arXiv:1204.0641*, Apr. 2012.
8. M. Biely, U. Schmid, and B. Weiss. Synchronous consensus under hybrid process and link failures. *Theoretical Computer Science*, 412(40):5602 – 5630, 2011. http://dx.doi.org/10.1016/j.tcs.2010.09.032.
9. F. Bonnet and M. Raynal. On the road to the weakest failure detector for k-set agreement in message-passing systems. *Theoretical Computer Science*, 412(33):4273 – 4284, 2011.
10. A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *IJPEDS*, 27(5):387–408, 2012.
11. B. Charron-Bost and A. Schiper. The Heard-Of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, Apr. 2009.
12. S. Chaudhuri. More *choices* allow more *faults*: Set consensus problems in totally asynchronous systems. *Information and Control*, 105(1):132–158, July 1993.

13. H. C. Chung, P. Robinson, and 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, pages 52–61, New York, NY, USA, 2011. ACM.
14. É. Coulouma and E. Godard. A characterization of dynamic networks where consensus is solvable. In *Proceedings Structural Information and Communication Complexity - 20th International Colloquium (SIROCCO'13), Springer LNCS 8179*, pages 24–35, 2013.
15. E. Gafni. Round-by-round fault detectors (extended abstract): unifying synchrony and asynchrony. In *Proceedings of the Seventeenth Annual ACM Symposium on Principles of Distributed Computing*, pages 143–152, Puerto Vallarta, Mexico, 1998. ACM Press.
16. A. Goiser, S. Khattab, G. Fassl, and 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)*, pages 15–21, Athens, Greece, June 2010.
17. R. Ingram, P. Shields, J. E. Walter, and J. L. Welch. An asynchronous leader election algorithm for dynamic networks. In *IPDPS*, pages 1–12, 2009.
18. F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *STOC*, pages 513–522, 2010.
19. F. Kuhn and R. Oshman. Dynamic networks: Models and algorithms. *SIGACT News*, 42(1):82–96, 2011.
20. F. Kuhn, R. Oshman, and Y. Moses. Coordinated consensus in dynamic networks. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Prin ciples of distributed computing*, PODC '11. ACM, 2011.
21. F. Kuhn, S. Schmid, and R. Wattenhofer. Towards worst-case churn resistant peer-to-peer systems. *Distributed Computing*, 22(4):249–267, 2010.
22. L. Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.
23. F. Mattern. Virtual time and global states of distributed systems. In *Parallel and Distributed Algorithms*, pages 215–226. North-Holland, 1989.
24. K. J. Perry and S. Toueg. Distributed agreement in the presence of processor and communication faults. *IEEE Transactions on Software Engineering*, SE-12(3):477–482, March 1986.
25. M. Raynal and J. Stainer. Synchrony weakened by message adversaries vs asynchrony restricted by failure detectors. In *Proceedings ACM Symposium on Principles of Distributed Computing (PODC'13)*, pages 166–175, 2013.
26. N. Santoro and P. Widmayer. Time is not a healer. In *Proc. 6th Annual Symposium on Theor. Aspects of Computer Science (STACS'89)*, LNCS 349, pages 304–313, Paderborn, Germany, Feb. 1989. Springer-Verlag.
27. U. Schmid, B. Weiss, and I. Keidar. Impossibility results and lower bounds for consensus under link failures. *SIAM Journal on Computing*, 38(5):1912–1951, 2009.
28. N. H. Vaidya and D. K. Pradhan. Degradable agreement in the presence of Byzantine faults. In *International Conference on Distributed Computing Systems*, pages 237–244, 1993.
29. C. Ware, J. Judge, J. Chicharo, and E. Dutkiewicz. Unfairness and capture behaviour in 802.11 adhoc networks. In *2000 IEEE International Conference on Communications. ICC 2000. Global Convergence Through Communications.*, 2000.