Noname manuscript No.
(will be inserted by the editor)

Agreement in Directed Dynamic Networks

Martin Biely · Peter Robinson · Ulrich Schmid

the date of receipt and acceptance should be inserted later

Abstract We study the fundamental problem of achieving consensus in a synchronous dynamic network, where an omniscient adversary controls the unidirectional communication links. Its behavior is modeled as a sequence of *directed* graphs representing the active (i.e. timely) communication links per round. We prove that consensus is impossible under some natural weak connectivity assumptions, and introduce vertex-stable root components as a—practical and not overly strong—means for circumventing this impossibility. Essentially, we assume that there is a short period of time during which an arbitrary part of the network remains strongly connected, while its interconnect topology keeps changing continuously. We present a consensus algorithm that works under this assumption, and prove its correctness. Our algorithm maintains a local estimate of the communication graphs, and applies techniques for detecting stable network properties and univalent system configurations. Our possibility results are complemented by several impossibility results and lower bounds, which reveal that our algorithm is asymptotically optimal.

This work has been supported by the Austrian Science Foundation (FWF) project P20529 and S11405. Peter Robinson has also been supported in part by Nanyang Technological University grant M58110000 and Singapore Ministry of Education (MOE) Academic Research Fund (AcRF) Tier 2 grant MOE2010-T2-2-082.

Martin Biely
EPFL, Switzerland. E-mail: martin.biely@epfl.ch

Peter Robinson
Nanyang Technological University, Division of Mathematical Sciences, Singapore. E-mail: peter.robinson@ntu.edu.sg

Ulrich Schmid
Embedded Computing Systems Group, Technische Universität Wien, Austria. E-mail: s@ecs.tuwien.ac.at

1 Introduction

Dynamic networks, instantiated, e.g., by (wired) peer-to-peer (P2P) networks, (wireless) sensor networks, mobile ad-hoc networks and vehicular area networks, are becoming ubiquitous nowadays. The primary properties of such networks are (i) sets of participants (called processes in the sequel) that are a priori unknown and maybe time-varying, and (ii) the absence of central control. Such assumptions make it very difficult to setup and maintain the basic system, and create particular challenges for the design of robust distributed services for applications running on such dynamic networks.

A natural approach to build robust services despite mobility, churn, failures, etc. of processes is to use distributed consensus to agree system-wide on (fundamental) parameters like schedules, frequencies, etc. Although system-wide agreement indeed provides a very convenient abstraction for building robust services, it inevitably rests on the ability to efficiently implement consensus in a dynamic network.

Doing this in *wireless* dynamic networks is particularly challenging, for several reasons: First, whereas wireline networks are usually adequately modeled by means of bidirectional links, this is not the case for wireless networks: Fading phenomenons and interference [14] are local effects that affect only the receiver of a wireless link. Since the receiver of the reverse link is located in a different place in the network, it is very likely that it faces very different levels of fading and interference. Thus, wireless links are more adequately modeled by means of pairs of unidirectional links, which are considered independent of each other.

Second, wireless networks are inherently broadcast. When a process transmits, then every other process within its transmission range will observe this transmission — either by legitimately receiving the message or as some form of interference. This creates quite irregular communication behavior, such as capture effects and near-far problems [26], where certain (nearby) transmitters may "lock" some receiver and thus prohibit the reception of messages from other senders. As a consequence, wireless links that work correctly at a given time may have a very irregular spatial distribution, and may also vary heavily with time.

Finally, taking also into account mobility of processes and/or peculiarities in the system design (for example, duty-cycling is often used to conserve energy in wireless sensor networks), it is obvious that static assumptions on the communication topology, as underlying classic models like unit disc graphs, are not adequate for wireless dynamic networks.

We hence argue that such dynamic systems can be modeled adequately only by means dynamically changing *directed* communication graphs. Since synchronized clocks are required already for basic communication in wireless systems, e.g., for transmission scheduling and sender/receiver synchronization, we also assume that the system is synchronous.

Main Contributions

Similar to Kuhn et al. [19], we consider consensus in a system modeled by means of a sequence of communication graphs, one for each round. In sharp contrast to existing work, our communication graphs are directed, and our rather weak connectivity assumptions do not guarantee bidirectional (multi-hop) communication between all processes.

(1) We prove that communication graphs that are weakly connected in every round are not sufficient for solving consensus, and introduce a fairly weak additional assumption that allows to overcome this impossibility. It requires that, in every round, there is exactly one (possibly changing) strongly connected component (called a *root component*) that has only out-going links to (some of) the remaining processes and can reach every process in the system via several hops. Since this assumption is still too weak for solving consensus, we add the requirement that, eventually, there will be a short interval of time where the processes in the root component remain the same, although the connection topology may still change. We coined the term *vertex-stable root component* (for some window of limited stability) for this requirement.
(2) We provide a consensus algorithm that works in this model, and prove its correctness. Our algorithm requires a window of stability that has a size of $4D$, where D is the number of rounds required to reach all processes in the network from any process in the vertex-stable root component. While in general D can be in $O(n)$, we show how to obtain an improved running time (in $O(\log n)$) when assuming certain expanding graph topologies.
(3) We show that any consensus and leader election algorithm has to know an a priori bound on D. Since the system size n is a trivial bound on D, this implies that there is no uniform algorithm, i.e., no algorithm unaware of the size of the network, that solves consensus in our model. This holds for deterministic and randomized (Las Vegas) algorithms. In addition, we establish a lower bound of D for the window of stability.
(4) We prove that neither reliable broadcast, atomic broadcast, nor causal-order broadcast can be implemented in our model without additional assumptions. Moreover, there is no algorithm that solves counting, k-verification, k-token dissemination, all-to-all token dissemination, and k-committee election.

2 Related Work

We are are not aware of any previous work on consensus in *directed* and *dynamic* networks with such weak connectivity requirements. This is also true for an earlier paper [5], where we assumed the existence of an underlying *static* skeleton graph (a non-empty common intersection of all communication graphs of all rounds), which had to include a *static* root component. By contrast, in this paper, we allow the graphs to be totally dynamic, except for a (sufficiently large) time window where the members (but not the topology!) of the root component are the same.

Dynamic networks have been studied intensively in distributed computing. Early work on this topic includes [1, 3]. One basic assumption that can be used to categorize research in dynamic networks is whether the set of processes is assumed to be fixed, or subject to churn (i.e., processes join and leave over time). The latter has mostly been considered in the area of peer-to-peer networks and the construction of overlays. We refer the interested reader to [20] for a more detailed treatment of related work in this area.

When the set of processes is considered to be fixed, dynamicity in the network is modeled by changes in the network topology. Over time several approaches to modeling dynamic connectivity in networks have been proposed. We will in the following focus on two lines of research that are closest to ours: work that models directly the underlying (evolving) communication graph, and approaches taken in the context of consensus.

Evolving graph models

There is a rich body of literature on dynamic graph models going back to [15], which also mentions for the first time modeling a dynamic graph as a sequence of static graphs, as we do. A survey on dynamic networks can be found in [17]. Recently, Casteigts et al. [7] have introduced a classification of the assumptions about the temporal properties of time varying graphs. We will (cf. Lemma 14) show that our assumption falls between two of the weakest classes considered, as we can only guarantee one-directional reachability.[1] We are not aware of other work considering such weak assumptions in the context of agreement.

Closest to our own work is that of Kuhn et al. [19], who also consider agreement problems in dynamic networks based on the model of [18]. This model is based on distributed computations organized in lock-step rounds, and states assumptions on the connectivity in each round as a separate (round) communication graph. While the focus of [18] is the complexity of aggregation problems in dynamics networks, [19] focuses on agreement problems; more specifically on the Δ-coordinated consensus problem, which extends consensus by requiring all processes to decide within Δ rounds of the first decision. In both papers, only undirected graphs that are *connected in every round* are considered. In terms of the classes of [7], the model of [18] is in one of the strongest classes (i.e. Class 10), which means that each process is always reachable by every other process. Since node failures are not considered, solving consensus is always possible in this model without additional assumptions, thus the focus of [19] is on Δ-coordinated and simultaneous consensus and its time complexity.

Dynamic networks have been studied in [2] in the context of "stable almost-everywhere agreement" (a variant of the "almost everywhere agreement" problem introduced by [10]), which weakens the classic consensus problem in the sense that a small linear fraction of processes might remain undecided. In the model of [2], the adversary can subject up to a linear fraction of nodes to churn per round; assuming that the network size remains stable, this means that up to εn nodes (for some small $\varepsilon > 0$) can be replaced by new nodes in every round. Moreover, changes to the (undirected) topology of the network are also under the control of the adversary. To avoid almost-everywhere agreement from becoming trivially unsolvable, [2] assume that the network is always an expander.

The leader election problem in dynamic networks has been studied in [9] where the adversary controls the mobility of nodes in a wireless ad-hoc network. This induces dynamic changes to the (undirected) network topology in every round and requires any leader election algorithm to take $\Omega(Dn)$ rounds in the worst case, where D is an upper bound on information propagation.

[1] Here, reachability does not refer to the graph-theoretic concept of reachability, but rather to the ability to eventually communicate information to another process.

Transmission Failure Models

Instead of considering a dynamic graph that defines which processes communicate in each round, an alternative approach is based on the (dual) idea of assuming a fully connected network of (potential) communication, and considering that communication in a round can fail. The notion of transmission failures was introduced by Santoro and Widmayer [24], who assumed dynamic transmission failures and showed that $n-1$ dynamic transmission failures in the benign case (or $n/2$ in case of arbitrary transmission failures) render any non-trivial agreement impossible. As it assumes unrestricted transmission failures (the (only) case considered in their proof are failures that affect all the transmissions of a *single* process), it does not apply to any model which considers perpetual mutual reachability of processes (e.g., [19]).

The HO-model [8] is also based on transmission failures. It relies on the collection of sets of processes a process *hears of* (i.e., receives a message from) in a round. Different system assumptions are modeled by predicates over this collection of sets. The HO-model is totally oblivious to the actual reason *why* some process does not hear from another one: Whether the sender committed a send omission or crashed, the message was lost during transmission or is simply late, or the receiver committed a receive omission. A version of the model also allowing communication to be corrupted is presented in [4]. Indeed, the HO-model is very close to our graph model, as an edge from p to q in the graph of round r corresponds to p being in the round r HO set of q.

The approach taken by Gafni [13] has some similarities to the HO-model (of which it is a predecessor), but is more focused on process failures than the work by Santoro and Widmayer. Here an oracle (a round-by-round failure detector) is considered to tell processes the set of processes they will be not be able to receive data from in the current round. Unlike the approaches discussed above, it explicitly states how rounds are implemented; nevertheless, the oracle abstracts away the actually reason for not receiving a message. So, like in the HO-model, the same device is used to describe failures and (a)synchrony.

Another related model is the perception based failure model [6, 25], which uses a sequence of perception matrices (corresponding to HO sets) to express failures of processes and links. As for communication failures, the impossibility result of Santoro and Widmayer is circumvented by putting separate restrictions on the number of outgoing and incoming links that can be affected by transmission failures [25]. Since transmission failures are counted on a per process/per round basis, agreement was shown to be possible in the presence of $O(n^2)$ total transmission failures per round.

The approach we used in [5] relied on restricting the communication failures per round in a way that secures the existence of a *static* skeleton graph (which exists in all rounds). In the terminology of this paper, [5] stipulates the existence of an ∞-interval vertex-stable root component. By contrast, Assumption 1 used in this paper is even weaker than eventual T-interval connectivity, as we never forbid the set of edges to change from one round to the next. That is, in our model, the intersection of the edge sets in two consecutive rounds can always be empty, even during the stable window.

3 Model and Preliminaries

We consider synchronous computations of a dynamic network of a fixed set of distributed processes Π with $|\Pi| = n \geqslant 2$. Processes can communicate with their current neighbors in the network by sending messages taken from some finite message alphabet $\mathcal{M}$.

In the following three subsections, we will present our computational model and define what it means to solve consensus in this model. In Section 4, we will introduce constraints that make consensus solvable in our model.

3.1 Computational Model.

Similar to the $\mathcal{LOCAL}$ model [23], we assume that processes organize their computation as an infinite sequence of lock-step rounds. For every $p \in \Pi$ and each round $r > 0$, let $S_p^r \in \mathcal{S}_p$ be the state of p at the beginning of round r; the initial state is denoted by $S_p^1 \in \mathcal{S}_p^1 \subset \mathcal{S}_p$. The round r computation of process p is determined by the following two functions that make up p's algorithm: The message sending function $M_p : \mathcal{S}_p \to \mathcal{M}$ determines the message m_p^r broadcast by p in round r, based on p's state S_p^r at the beginning of round r. We assume that some (possibly empty) message is broadcast in every round, to all (current!) neighbors of p. The neighborhood of a process in round r depends solely on the underlying communication graph of round r, which we define in Section 3.3. The transition function $T_p : \mathcal{S}_p \times 2^{(\Pi \times \mathcal{M})} \to \mathcal{S}_p$ takes p's state S_p^r at the beginning of round r and a set of pairs of process ids and messages μ_p^r. This set represents the round r messages[2] received by p from other processes in the system, and computes the successor state S_p^{r+1}. We assume that, for each process q, there is at most one $(q, m_q^r) \in \mu_p^r$ such that m_q^r is the message q sent in round r. Note that neither M_p nor T_p need to involve n, i.e., the algorithms executed by the processes may be uniform w.r.t. the network size n.

[2] We only consider messages sent in round r here, so we assume communication-closed [11] rounds.

3.2 Consensus

To formally introduce the consensus problem, we assume some ordered set V and consider the set of possible initial states $\mathcal{S}_p^1$ (of process p) to be partitioned into $|V|$ subsets $\mathcal{S}_p^1[v]$, with $v \in V$. When p starts in a state in $\mathcal{S}_p^1[v]$, we say that v is p's input value, denoted $v_p = v$. Moreover, we assume that, for each $v \in V$, there is a (sub-)set $\mathcal{D}_p[v] \subset \mathcal{S}_p$ of decided states that is closed under p's transition function, i.e., where T_p maps any state in this subset to this subset. We say that p has decided on v when it is in some state in $\mathcal{D}_p[v]$. When p performs a transition from a state outside of the set of decided states to the set of decided states, we say that p decides. We say that an *algorithm $\mathcal{A}$ solves consensus* if the following properties hold in every run of $\mathcal{A}$:

Agreement: If process p decides on x_p and q decides on x_q, then $x_p = x_q$.

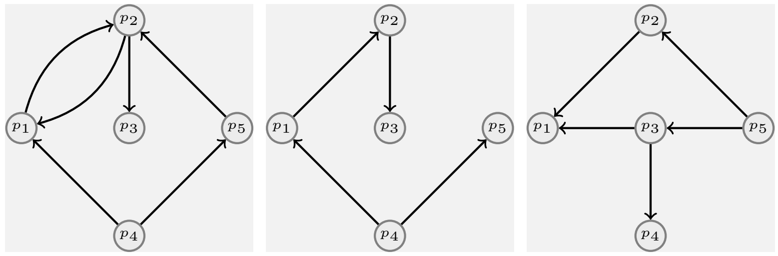
Validity: If a process decides on v, then v was proposed by some q, i.e., $v_q = v$.

Termination: Every process must eventually decide.

At a first glance, solving consensus might appear easier in our model than in the classic crash failure model, where processes simply stop executing the algorithm. This is not the case, however. As in [8], we model crash failures as follows: A process q that crashes in round r is equivalent to taking away all outgoing edges of q from round $r+1$ on. While q itself can still receive messages and perform computations, the remaining processes are not influenced by q from round r on.

3.3 Communication Model

The evolving nature of the network topology is modeled as an infinite sequence of simple directed graphs $\mathcal{G}^1, \mathcal{G}^2, \ldots$, which is fixed by an adversary having access to the processes' states. For each round r, we denote the *round r communication graph* by $\mathcal{G}^r = \langle V, E^r \rangle$, where each node of the set V is associated with one process from the set of processes Π, and where E^r is the set of directed edges for round r, such that there is an edge from p to q, denoted as $(p \to q)$, *iff* q receives p's round r message (in round r). Figure 1 shows a sequence of graphs for a network of 5 processes, for rounds 1 to 3. For any (sub)graph G, we will use the notation $V(G)$



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

Fig. 1:

and $E(G)$ to refer to the set of vertices respectively edges of G, i.e., it always holds that $G = \langle V(G), E(G)\rangle$.

For deterministic algorithms, a run is completely determined by the input values assigned to the processes and the sequence of communication graphs. Therefore, the fact that the adversary knows only the initial values does not pose a limitation to its power.

To simplify the presentation, we will denote a process and the associated node in the communication graph by the same symbols and omit the set from which it is taken if there is no ambiguity. We will henceforth write $p \in \mathcal{G}^r$ and $(p \to q) \in \mathcal{G}^r$ instead of $p \in V(G)$ resp. $(p \to q) \in E^r$.

The *neighborhood of* p *in round* r is the set of processes $\mathcal{N}_p^r$ that p receives messages from in round r, formally, $\mathcal{N}_p^r = \{q \mid (q \to p) \in \mathcal{G}^r\}$.

Similarly to the classic notion of "happened-before" introduced in [21], we say that a process p *(causally) influences process* q *in round* r, expressed by $(p \overset{r}{\rightsquigarrow} q)$ or just $(p \rightsquigarrow q)$ if r is clear from the context, iff either (i) $p \in \mathcal{N}_q^r$, or (ii) if $q = p$, i.e., we assume that p always 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, graphically denoted by $(p \overset{r[k]}{\rightsquigarrow} q)$or simply $(p \overset{[k]}{\rightsquigarrow} q)$, if there exists a sequence of not necessarily distinct processes $p = p_0, \ldots, p_k = q$ such that p_i influences p_{i+1} in round $r + i$, for all $0 \leqslant i < k$. If k is irrelevant, we just write $(p \overset{r}{\rightsquigarrow} q)$ or just $(p \rightsquigarrow q)$ and say that p (in round r) causally influences q.

The *causal distance* $\mathtt{cd}^r(p, q)$ 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, $\mathtt{cd}^r(p, q) := \min\{k \mid (p \overset{r[k]}{\rightsquigarrow} q)\}$. Note that we assume $\mathtt{cd}^r(p, p) = 1$. The following Lemma 1 shows that the causal distance in successive rounds cannot arbitrarily decrease.

Lemma 1 (Causal distance in successive rounds) *For every round* $r \geqslant 1$ *and every two processes* $p, q \in \Pi$, *it holds that* $\mathtt{cd}^{r+1}(p, q) \geqslant \mathtt{cd}^r(p, q) - 1$. *As a consequence, if* $\mathtt{cd}^r(p, q) = \infty$, *then also* $\mathtt{cd}^{r+1}(p, q) = \infty$.

Proof Since $(p \rightsquigarrow p)$ in every round r, the definition of causal distance trivially implies $\mathtt{cd}^r(p, q) \leqslant 1 + \mathtt{cd}^{r+1}(p, q)$. □

Note that, in contrast to the similar notion of dynamic distance defined in [17], the causal distance in *directed* graphs is not necessarily symmetric. Moreover, if the adversary chooses the graphs $\mathcal{G}^r$ such that not all nodes are strongly connected, the causal distance can even be infinite. In fact, even if $\mathcal{G}^r$ is strongly connected for round r (but not for rounds $r' > r$), $\mathtt{cd}^r(p, q)$ can be infinite. As we will not consider the whole communication graph to be strongly connected in this paper, we make use of the notation of *strongly connected components (SCC)*. We write $\mathcal{C}_p^r$ to denote the (unique) SCC of $\mathcal{G}^r$ that contains process p in round r or simply $\mathcal{C}^r$ if p is irrelevant.

It is apparent that $\mathtt{cd}^r(p, q)$ and $\mathtt{cd}^r(q, p)$ may be infinite even if $q \in \mathcal{C}_p^r$. In order to be able to argue (meaningfully) about the maximal length of causal chains within an SCC, we also need some "continuity property" over rounds. This leads us to the crucial concept of an I*-vertex-stable strongly connected component*, denoted as $\mathcal{C}^I$: It requires that the set of vertices of a strongly connected component $\mathcal{C}$ remains stable throughout all rounds in the nonempty interval I. Its topology may undergo changes, but must form an SCC in every round. Formally, $\mathcal{C}^I$ being vertex-stable during I requires that $\forall p \in \mathcal{C}^I, \forall r \in I : V(\mathcal{C}_p^r) = V(\mathcal{C}^I)$. The important property of $\mathcal{C}^I$ is that information is guaranteed to spread to all vertices of $\mathcal{C}^I$ if the interval I is large enough (cf. Lemma 3).

Let the *round* r *causal diameter* $D^r(\mathcal{C}^I)$ of a vertex-stable SCC $\mathcal{C}^I$ be the largest causal distance $\mathtt{cd}^r(p, q)$ for any $p, q \in \mathcal{C}^I$. The *causal diameter* $D(\mathcal{C}^I)$ of a vertex-stable SCC $\mathcal{C}^I$ in I is the largest causal distance $\mathtt{cd}^x(p, q)$ starting at any round $x \in I$ that "ends" in I, i.e., $x + \mathtt{cd}^x(p, q) - 1 \in I$. If there is no such causal distance (because I is too short), $D(\mathcal{C}^I)$ is assumed to be infinite. Formally, for $I = [r, s]$ with $s \geqslant r$,[3]

$$D(\mathcal{C}^I) = \min\{\max\{D^x(\mathcal{C}^I) \mid x \in [r, s] \text{ and } x + D^x(\mathcal{C}^I) - 1 \leqslant s\} \cup \{\infty\}\}.$$

If $\mathcal{C}^I$ consists only of one process, then we obviously have $D(\mathcal{C}^I) = 1$. The following Lemma 2 establishes a bound for $D(\mathcal{C}^I)$ also for the general case.

Lemma 2 (Bound on causal diameter) *Let a vertex-stable SCC $\mathcal{C}^I$ for some $I = [r, s]$ be given and let $|\mathcal{C}^I| \geqslant 2$ be the number of processes in $\mathcal{C}^I$. If $s \geqslant r + |\mathcal{C}^I| - 2$, then $D(\mathcal{C}^I) \leqslant |\mathcal{C}^I| - 1$.*

Proof Fix some process $p \in \mathcal{C}^I$ and some r' where $r \leqslant r' \leqslant s - |\mathcal{C}^I| + 2$. Let $\mathcal{P}_0 = \{p\}$, and define for each $i > 0$ the set $\mathcal{P}_i = \mathcal{P}_{i-1} \cup \{q : \exists q' \in \mathcal{P}_{i-1} : q' \in \mathcal{N}_q^{r'+i-1}\}$. $\mathcal{P}_i$ is hence the set of processes q such that $(p \overset{r'[i]}{\rightsquigarrow} q)$ holds. Using induction, we will show that $|\mathcal{P}_k| \geqslant \min\{|\mathcal{C}^I|, k+1\}$ for $k \geqslant 0$. Induction start $k = 0$: $|\mathcal{P}_0| \geqslant \min\{|\mathcal{C}^I|, 1\} = 1$ follows immediately from $\mathcal{P}_0 = \{p\}$. Induction step $k \to k+1$, $k \geqslant 0$: First assume that already $|\mathcal{P}_k| = |\mathcal{C}^I| \geqslant \min\{|\mathcal{C}^I|, k+1\}$; since $|\mathcal{P}_{k+1}| \geqslant |\mathcal{P}_k| = |\mathcal{C}^I| \geqslant \min\{|\mathcal{C}^I|, k+1\}$, we are done. Otherwise, consider round $r' + k$ and $|\mathcal{P}_k| < |\mathcal{C}^I|$: It follows from strong connectivity of $\mathcal{G}^{r'+k} \cap \mathcal{C}^I$ that there is a set of edges from processes in $\mathcal{P}_k$ to some non-empty set $\mathcal{L}_k \subseteq \mathcal{C}^I \backslash \mathcal{P}_k$. Hence, we have $\mathcal{P}_{k+1} = \mathcal{P}_k \cup \mathcal{L}_k$, which implies $|\mathcal{P}_{k+1}| \geqslant |\mathcal{P}_k| + 1 \geqslant k + 1 + 1 = k + 2 = \min\{|\mathcal{C}^I|, k+2\}$ by the induction hypothesis.

Thus, in order to guarantee $\mathcal{C}^I = \mathcal{P}_k$ and thus $|\mathcal{C}^I| = |\mathcal{P}_k|$, choosing k such that $|\mathcal{C}^I| = 1 + k$ and $k \leqslant s - r' + 1$ is sufficient. Since $s \geqslant r' + |\mathcal{C}^I| - 2$, both conditions can be fulfilled by choosing $k = |\mathcal{C}^I| - 1$. Moreover, due to the definition of $\mathcal{P}_k$, it follows that $\mathtt{cd}^{r'}(p, q) \leqslant |\mathcal{C}^I| - 1$ for all $q \in \mathcal{C}^I$. Since this holds for any p and any $r' \leqslant s - |\mathcal{C}^I| + 2$, we finally obtain $|\mathcal{C}^I| - 1 \geqslant D^{r'}(\mathcal{C}^I)$ and hence $|\mathcal{C}^I| - 1 \geqslant D(\mathcal{C}^I)$, which completes the proof of Lemma 2.

Given this result, it is tempting to assume that, for any vertex-stable SCC $\mathcal{C}^I$ with finite causal diameter $D(\mathcal{C}^I)$, any information propagation that starts at least $D(\mathcal{C}^I) - 1$ rounds before the final round of I will reach all processes in $\mathcal{C}^I$ within I. This is not generally true, however, as the following example for $I = [1, 3]$ and a vertex-stable SCC of four processes shows: If $\mathcal{G}^1$ is the complete graph whereas $\mathcal{G}^2 = \mathcal{G}^3$ is a ring, $D(\mathcal{C}^I) = 1$, but information propagation starting at round 2 does not finish by the end of round 3. However, the following Lemma 3 gives a bound on the latest starting round that guarantees this property.

[3] Since I ranges from the beginning of r to the end of s, we define $|I| = s - r + 1$.

Lemma 3 (Information propagation) *Suppose that $\mathcal{C}^I$ is an I-vertex-stable strongly connected component of size $\geqslant 2$ that has $D(\mathcal{C}^I) < \infty$, for $I = [r, s]$, and let x be the maximal round where $x + D^x(\mathcal{C}^I) - 1 \leqslant s$. Then,*

(i) for every $x' \in [r, x]$, it holds that $x' + D^{x'}(\mathcal{C}^I) - 1 \leqslant s$ and $D^{x'}(\mathcal{C}^I) \leqslant D(\mathcal{C}^I)$ as well, and
(ii) $x \geqslant \max\{s - |\mathcal{C}^I| + 2, r\}$.

Proof Since $D(\mathcal{C}_I) < \infty$, the maximal round x always exists. Lemma 1 reveals that for all $p, q \in \mathcal{C}^I$, we have $x - 1 + \mathtt{cd}^{x-1}(p, q) - 1 \leqslant x + \mathtt{cd}^x(p, q) - 1 \leqslant s$, which implies $x' + \mathtt{cd}^{x'}(p, q) - 1 \leqslant s$ for every x' where $r \leqslant x' \leqslant x$ and proves (i). The bound given in (ii) follows immediately from Lemma 2.

Since we will frequently require a vertex-stable SCC $\mathcal{C}^I$ that guarantees bounded information propagation also for late starting rounds, we introduce the following Definition 1.

Definition 1 (D-bounded I-vertex-stable SCC) An I-vertex-stable SCC $\mathcal{C}^I$ with $I = [r, s]$ is *D-bounded* if $D \geqslant D^I(\mathcal{C}^I)$ and $D^{s-D+1}(C^I) \leqslant D$.

4 Required Connectivity Properties

Up to now, we did not provide any guarantees on the connectivity of the network, the lack of which makes consensus trivially impossible.[4] In this section, we will add some weak constraints on the adversary that circumvent this impossibility. Obviously, we want to avoid requesting strong properties of the network topology (such as stating that $\mathcal{G}^r$ is strongly connected in every round r), as this would reduce the applicability of our results in real networks.

As a first attempt, we could assume that, in every round r, the communication graph $\mathcal{G}^r$ is weakly connected. This, however, turns out to be too weak. Even if the adversary choses a *static* topology, it is easy to see that consensus remains impossible: Consider for example the graph that is partitioned into 3 strongly connected components $\mathcal{C}_0$, $\mathcal{C}_1$, and $\mathcal{C}_2$ such that there are only outgoing edges from $\mathcal{C}_0$ respectively $\mathcal{C}_1$ pointing to $\mathcal{C}_2$, whereas $\mathcal{C}_2$ has no outgoing edges. If all processes in $\mathcal{C}_0$ start with 0 and all processes in $\mathcal{C}_1$ start with 1, this yields a contradiction to agreement: For $i \in \{0, 1\}$, processes in $\mathcal{C}_i$ can never learn the value $1 - i$, thus, by an easy indistinguishability argument, it follows that processes in $\mathcal{C}_0$ and $\mathcal{C}_1$ must decide on conflicting values.

[4] The adversary can simply choose the empty set for the set of edges in every round.

In order to define constraints that rule out the existence of $\mathcal{C}_0$ and $\mathcal{C}_1$ as above, the concept of *root components* proves useful: Let $\mathcal{R}^r \subseteq \mathcal{G}^r$ be an SCC that has no incoming edges from any $q \in \mathcal{G}^r \setminus \mathcal{R}^r$. We say that $\mathcal{R}^r$ is a *root component* in round r, formally: $\forall p \in \mathcal{R}^r\ \forall q \in \mathcal{G}^r : (q \rightarrow p) \in \mathcal{G}^r \Rightarrow q \in \mathcal{R}^r$. For example, in Figure 1a, process p_4 forms a root component by itself, while processes p_1 and p_2 form a SCC that is not a root component since it has incoming edges.

The following Lemma 4 establishes some simple facts on $\mathcal{G}^r$.

Lemma 4 *Any $\mathcal{G}^r$ contains at least one and at most n root components (isolated processes), which are all disjoint. If $\mathcal{G}^r$ contains a single root component $\mathcal{R}^r$, then $\mathcal{G}^r$ is weakly connected, and there is in fact a directed (out-going) path from every $p \in \mathcal{R}^r$ to every $q \in \Pi$ in $\mathcal{G}^r$.*

Proof We first show that every weakly connected directed simple graph G has at least one root component. To see this, contract every SCC (including its connecting internal edges) to a single vertex. The resulting graph G' is a directed acyclic graph (DAG) (and of course still weakly connected). By the general properties of DAGs, G' has at least one vertex v that has only out-going edges. By construction, v corresponds to a root component in the original graph G. Since $\mathcal{G}^r$ has at least 1 and at most n weakly connected components, the first statement of our lemma follows.

To prove the second statement for the case where there is a single root component, we first observe that $\mathcal{G}^r$ must be weakly connected: If there were $\geqslant 2$ isolated components, we would get at least two root components by the above result.

To prove that there is a directed path from every $p \in \mathcal{R}^r$ to every $q \in \Pi$ in $\mathcal{G}^r$, assume the contrary: There is some $q \in \Pi$ such that there is no path from any $p \in \mathcal{R}^r$ to q (since all processes in $\mathcal{R}^r$ are strongly connected, we can safely assume this). Since $\mathcal{G}^r$ is weakly connected, there are node-disjoint undirected paths connecting q and some $p \in \mathcal{R}^r$. Traversing any such path starting from p must lead to some vertex $s \neq q$ where a reverse edge $(s' \rightarrow s)$ lying on this path ends, since an out-going path from p to q would exist otherwise.

Now consider the Graph G obtained by removing all these edges $(s' \rightarrow s)$ from $\mathcal{G}^r$. By construction, in G, all vertices s lie in the same weakly connected component $C_{\mathcal{R}^r}$ in which $\mathcal{R}^r$ lies, and all vertices s' lie in the same weakly connected component C_q in which q lies, and $C_{\mathcal{R}^r} \neq C_q$. Hence, C_q has at least one root component $\mathcal{R}_q$, and since we only removed out-going edges from C_q in the transition from $\mathcal{G}^r$ to G, $\mathcal{R}_q$ is also a root component in $\mathcal{G}^r$. This contradicts the unique root component $\mathcal{R}^r$ of $\mathcal{G}^r$, however.

Returning to the consensus impossibility example for weakly connected graphs above, it is apparent that the two components $\mathcal{C}_0$ and $\mathcal{C}_1$ are indeed both root components. Since consensus is not solvable in this case, we assume in the sequel that there is at most *a single* root component in $\mathcal{G}^r$, for any round r. We already know (cf. [5]) that this assumption makes consensus solvable if the topology (and hence the root component) is static. Since we are interested in *dynamic* networks, however, we assume in this paper that the root component may change throughout the run, i.e., the (single) root component $\mathcal{R}^r$ of $\mathcal{G}^r$ might consist of a different set of processes in every round round r. Figure 1 shows a sequence of graphs where there is exactly one root component in every round. It is less straightforward to reason about the solvability of consensus in this case. We will establish in Section 7, consensus is again impossible to solve without further constraints.

As root components are special cases of strongly connected components, we define an *I-vertex-stable root component* $\mathcal{R}^I$ as an I-vertex-stable strongly connected component that is a root component in every round $r \in I$. Clearly, all the definitions and results for vertex-stable components carry over to vertex-stable root components.

Restricting our attention to the case where exactly one vertex-stable root component $\mathcal{R}^I$ exists, it immediately follows from Lemma 4 that information of any process in $\mathcal{R}^I$ propagates to all nodes in Π if I is large enough. More specifically, we can extend our notions of causal diameter of a vertex-stable SCC to the whole network: The *round r network causal diameter D^r* is the largest $\mathtt{cd}^r(p,q)$ for any $p \in \mathcal{R}^r$ and any $q \in \Pi$. Similarly to the causal diameter of a vertex-stable component of an interval, we define the *network causal diameter D^I* for an interval I as the largest round x, $x \in I$, network causal diameter that also (temporally) falls within I, i.e., satisfies $x + D^x - 1 \in I$ and hence $x + \mathtt{cd}^x(p,q) - 1 \in I$ for any $p \in \mathcal{R}^r$ and any $q \in \Pi$.

The following versions of Lemma 2 and 3 for root components and their causal influence on the whole network can be established by proofs almost identical to the ones of their SCC versions:

Lemma 5 (Bound on network causal diameter) *Assume that there is a single vertex-stable root component $\mathcal{R}^I$ for some $I = [r,s]$. If $s \geqslant r + n - 2$, then $D^I \leqslant n - 1$.*

Note that Lemma 5 considers the worst case where the network topologies can even correspond to a line.

This assumption might be too pessimistic for many real world networks. By using graph topologies that allow fast information spreading, we can get much better causal network diameters like $D^I \in O(\log n)$ (cf. Section 6.2).

Lemma 6 (Network information propagation) *Assume that there is a single vertex-stable root component $\mathcal{R}^I$ in $I = [r, s]$ with network causal diameter $D^I < \infty$. Let x be the maximal round where $x + D^x - 1 \leqslant s$. Then,*
(i) for every $x' \in [r, x]$, it also holds that also $x' + D^{x'} \leqslant s$ and $D^{x'} \leqslant D^I$, and
(ii) $x \geqslant \max\{s - n + 2, r\}$.

As in the case of I-vertex-stable SCCs we also define the D-bounded variant of root components, which are central to our model.

Definition 2 (D-bounded I-vertex-stable root components) A vertex-stable root component $\mathcal{R}^I$ in $I = [r, s]$, $s \geqslant r$, is D-bounded if $D \geqslant D^I$ and $D^{s-D+1} \leqslant D$.

Note that a plain I-vertex-stable root component with $I \geqslant n - 1$ is always D-bounded for $D = n - 1$, recall Lemma 6. Our definition also allows some smaller choice of D, however.

We will show in Section 7 that the following Assumption 1 is indeed very weak, in the sense that many problems considered in distributed computing remain unsolvable.

Assumption 1 *For any round r, there is exactly one root component $\mathcal{R}^r$ in $\mathcal{G}^r$, and all vertex-stable root components $\mathcal{R}^I$ with $|I| \geqslant D$ are D-bounded. Moreover, there exists an interval of rounds $I = [r_{ST}, r_{ST}+d]$, with $d > 4D$, such that there is a D-bounded I-vertex-stable root component.*

5 The Local Network Approximation Algorithm

Initially, every process p has no knowledge of the network — it only knows its own input value. Any algorithm that correctly solves consensus must guarantee that, when p makes its decision, it either knows that its value has been/will be adopted by all other processes or it has agreed to take over some other process' decision value. In either case, process p needs to locally acquire knowledge about the information propagation in the network. As we have seen, p's information is only guaranteed to propagate throughout the network if p is in a I-vertex stable root component with finite network causal diameter D^I. Thus, for p to locally acquire knowledge about information propagation, it has to acquire knowledge about the (dynamic) communication graph. (We will discuss this point in more detail in Section 7.)

We allow p to achieve this by means of Algorithm 1, which essentially gathers as much local information on $\mathcal{G}^s$ as possible, for every past round s. Every process p keeps track of its current graph approximation in variable[5] A_p, which initially consists of process p, without any edges, and is broadcast and updated in every round. Ultimately, every process p will use A_p to determine whether it has been inside a vertex-stable root component for sufficiently many rounds. To this end, Algorithm 1 provides predicate `inStableRoot`(I), which returns true iff p has been in the I-vertex stable root component. The edges of A_p are labeled with a set of rounds constructed as follows: Since p can learn new information only via incoming messages, it updates A_p, whenever $q \in \mathcal{N}_p^r$, by adding $(q \xrightarrow{\{r\}} p)$ if q is p's neighbour for the first time, or updating the label of the edge $(q \xrightarrow{U} p)$ to $(q \xrightarrow{U \cup \{r\}} p)$ (Lines 5 and 7). Moreover, from q, p also receives A_q and uses it to update its own knowledge: The loop in Line 9 ensures that p has an edge $(v \xrightarrow{T \cup T'} w)$ for each $(v \xrightarrow{T'} w)$ in A_q, where T is the set of rounds previously known to p. Given A_p, we will denote the information contained in A_p about round s by $A_p|s$. More specifically, $A_p|s$ is the graph induced by the set of edges

$$E_p|s = \left\{ e = (v \to w) \mid \exists T \supseteq \{s\} : (v \xrightarrow{T} w) \in A_p \right\}.$$

It is important to note that our Assumption 1 is too weak to guarantee that eventually the graph $A_p|s$ will ever exactly match the actual $\mathcal{G}^s$ in some round s. In fact, there might be a process q that does not have any incoming links from other processes, throughout the entire run of the algorithm. In that case, q cannot learn anything about the remaining network, i.e., A_q will permanently be the singleton graph.

To simplify the presentation, we have refrained from purging outdated information from the network approximation graph. Our consensus algorithm only queries `inStableRoot` for intervals that span at most the last $4D + 1$ rounds, i.e., any older information can safely be removed from the approximation graph, yielding a message complexity that is bounded by a polynomial in n.

5.1 Proof of Correctness

[5] We denote the value of a variable v of process p in round r (before the round r computation finishes) as v_p^r; we usually suppress the superscript when it refers to the current round.

Algorithm 1 *Local Network Approximation (Process p)*

Provides predicate `inStableRoot()`.

Variables and Initialization:

1: $A_p := \langle V_p, E_p \rangle$ initially $(\{p\}, \emptyset)$ // weighted digraph without multi-edges and loops

Emit round r messages:

2: send $\langle A_p \rangle$ to all current neighbors

Round r: computation:

3: **for** $q \in \mathcal{N}_p^r$ and q sent message $\langle A_q \rangle$ in r **do**
4:   **if** $\exists$ edge $e = (q \xrightarrow{T} p) \in E_p$ **then**
5:     replace e with $(q \xrightarrow{T'} p)$ in E_p where $T' \leftarrow T \cup \{r\}$
6:   **else**
7:     add $e := (q \xrightarrow{\{r\}} p)$ to E_p
8:   $V_p \leftarrow V_p \cup V_q$

9: **for** every pair of nodes $(p_i, p_j) \in V_p \times V_p$, $p_i \neq p_j$ **do**
10:   **if** $T' = \bigcup \left\{ S \mid \exists q \in \mathcal{N}_p^r : (p_i \xrightarrow{S} p_j) \in E_q \right\} \neq \emptyset$ **then**
11:     replace $(p_i \xrightarrow{T} p_j)$ in E_p with $(p_i \xrightarrow{T \cup T'} p_j)$; add $(p_i \xrightarrow{T'} p_j)$ if no such edge exists

12: **predicate** `inStableRoot`(I)
13:   Let $A_p|s = (V_p^s, \left\{ (p_j \xrightarrow{T} p_i) \in E_p \mid s \in T \right\})$
14:   Let $C_p|s$ be $A_p|s$ if it is strongly connected, or the empty graph otherwise.
15:   return TRUE *iff* for all $s_1, s_2 \in I$: $V(C_p|s_1) = V(C_p|s_2) \neq \emptyset$

The following Lemma 7 reveals that $A_p|t$ underapproximates $\mathcal{G}^t$ is a way that consistently includes neighborhoods. Its proof uses a trivial invariant that is obvious from the code, which says that $A_p|t = \langle \{p\}, \emptyset \rangle$ at the end of every round $r < t$.

Lemma 7 *For every round $r \geqslant t$, if $A_p|t$ contains $(v \to w)$ at the end of round r, then*

(i) $(v \to w) \in \mathcal{G}^t$, i.e., $A_p|t \subseteq \mathcal{G}^t$,
(ii) $A_p|t$ also contains $(v' \to w)$ for every $v' \in \mathcal{N}_w^t \subseteq \mathcal{G}^t$.

Proof The proof is by induction on $r \geqslant t$.

Induction start $r = t$: 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

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

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.

Note that $A_p|t$ can be non-empty only for $t < r$, as round r information is added to A_p^r (by p) only at the end of round r. The range of validity of Lemma 7 could be extended to cover $r < t$ as well, since $A_p|t = \langle \{p\}, \emptyset \rangle$ at the end of every round $r < t$.

The next lemma shows that locally detecting a strongly connected component $C_p|s \subseteq A_p|s$ (in Line 14 of Algorithm 1) implies that p is in the root component of round s. This result rests on the fact that $A_p|s$ underapproximates $\mathcal{G}^s$ (Lemma 7.(i)), but does so in a way that never omits an in-edge at any process $q \in V(A_p|s)$ (Lemma 7.(ii)).

Lemma 8 *If the graph $C_p|s$ with $s < r$ is non-empty (hence an SCC) in round r, then $p \in \mathcal{R}^s$.*

Proof For a contradiction, assume that $C_p|s$ is non-empty (hence $A_p|s$ is an SCC by Line 14), but $p \notin \mathcal{R}^s$. Since p is always included in any A_p by construction and $A_p|s$ underapproximates $\mathcal{G}^s$ by Lemma 7.(i), this implies that $A_p|s$ cannot be the root component of $\mathcal{G}^s$. Rather, $A_p|s$ must contain some process w that has an in-edge $(v \to w)$ in $\mathcal{G}^s$ that is not present in $A_p|s$. As w and hence some edge $(q \xrightarrow{s} w)$ is contained in $A_p|s$, because it is an SCC, Lemma 7.(ii) reveals that this is impossible.

From this lemma and the description of predicate `inStableRoot`(I) in Algorithm 1, we get the following corollary.

Corollary 1 *If the predicate* `inStableRoot`(I) *evaluates to* TRUE *at process p in round r, then $\forall s \in I$ where $s < r$, it holds that $p \in \mathcal{R}^s$.*

The following Lemma 9 proves that, in a sufficiently long $I = [r, s]$ with a I-vertex-stable root component $\mathcal{R}^I$, every member p of $\mathcal{R}^I$ detects an SCC for round r (i.e., $C_p|r \neq \emptyset$) with a latency of at most D rounds (i.e., at the end of rounds $r + D$). Informally speaking, together with Lemma 8, it asserts that if there is an I-vertex-stable root component $\mathcal{R}^I$ for a sufficiently long intervals I, then a process p observes $C_p|r \neq \emptyset$ from the end of round $r + D$ on iff $p \in \mathcal{R}$.

Lemma 9 *Consider an interval of rounds $I = [r, s]$, such that there is a D-bounded I-vertex-stable root component $\mathcal{R}^I$ and assume $|I| = s - r + 1 > D \geqslant D(\mathcal{R}^I)$. Then, from round $r + D$ onwards, we have $C_p|r = \mathcal{R}^I$, for every process in $p \in \mathcal{R}^I$.*

Proof Consider any $q \in \mathcal{R}^I$. At the beginning of round $r+1$, q has an edge $(q' \xrightarrow{T} q)$ in its approximation graph A_q with $r \in T$ iff $q' \in \mathcal{N}_q^r$. Since processes always merge all graph information from other processes into their own graph approximation, it follows from the definition of a D-bounded I-vertex-stable root component in conjunction with the fact that $r + 1 \leqslant s - D + 1$ that every $p \in \mathcal{R}^I$ has these in-edges of q in its graph approximation by round $r+1+D-1$. Since $\mathcal{R}^I$ is a vertex-stable root-component, it is strongly connected without in-edges from processes outside $\mathcal{R}^I$. Hence $C_p|r = \mathcal{R}^I$ from round $r + D$ on, as asserted.

Corollary 2 *Consider an interval of rounds $I = [r, s]$, with $|I| = s - r + 1 > D \geqslant D(\mathcal{R}^I)$, such that there is a D-bounded vertex-stable root component $\mathcal{R}^I$. Then, from the end of round s on, predicate* `inStableRoot`*$([r, s-D])$ evaluates to* TRUE *at every process in $\mathcal{R}^I$.*

6 The Consensus Algorithm

The underlying idea of our consensus algorithm is to use flooding to forward the largest proposed value to everyone. However, as Assumption 1 does not guarantee bidirectional communication between every pair of processes, flooding is not sufficient: The largest proposal value could be known only to a single process that never has outgoing edges. Therefore, we let "privileged" processes, namely, the ones in a vertex-stable root component, try to impose their largest proposal values on the other processes. In order to do, so we use the well-known technique of locking a unique value. Processes only decide on their locked value once they are sure that every other process has locked this value as well. Since Assumption 1 guarantees that there will be one root component such that the processes in the root component can communicate their locked value to all other processes in the system they will eventually succeed.

Our consensus algorithm will hence be built atop of the network approximation algorithm. More specifically, in order to detect that a process is currently privileged, i.e., in a vertex-stable root component, we can rely on Corollary 1 and use the `inStableRoot` predicate provided by Algorithm 1. To be able to do so, round r of Algorithm 1 is executed before round r of Algorithm 2, and messages sent in round r by both algorithms are packed together in a single message. Since Corollary 2 revealed that `inStableRoot` has a delay of up to D rounds to detect that a process is in the vertex-stable root component of some interval of rounds, however, our algorithm (conservatively) looks back D rounds in the past for locking and privilege detection.

In more detail, Algorithm 2 proceeds as follows: Initially, processes do not have locked any value, therefore $lockRound_p = 0$ and $locked =$ FALSE. Processes try to detect whether they are privileged by evaluating the condition in Line 10. When this condition is true in some round ℓ, they lock the current value (by setting $locked_p =$ TRUE and $lockRound$ to the current round), unless $locked_p$ is already TRUE. Note that our locking mechanism does not actually protect the value against being overwritten by a larger value being also locked in ℓ; it locks out only those values that have older locks $l < \ell$.

When the process m that had the largest value in the root component of round ℓ detects that it has been in a vertex-stable root component in all rounds ℓ to $\ell + D$ (Line 20), it can decide on its current value. As all other processes in that root component must have had m's value imposed on them, they can decide as well. After deciding, a process stops participating in the flooding of locked values, but rather (Line 6) floods the network with $\langle$DECIDE$, x\rangle$. Since the time window guaranteed by Assumption 1 is large enough to allow every process to receive this message, all processes will eventually decide.

6.1 Proof of Correctness

Lemma 10 (Validity) *Every decision value is the input value of some process.*

Proof Processes decide either in Line 12 or in Line 21. When a process decides via the former case, it has received a $\langle$DECIDE$, x_q\rangle$ message, which is sent by q iff q has decided on x_q in an earlier round. In order to prove the theorem, it is thus sufficient to show that processes can only decide on some process' input value when they decide in Line 21, where they decide on their current estimate x_p. Let the round of this decision be r. This value is either p's initial value, or was updated in some round $r' \leqslant r$ in Line 14 from a value received by way of one of its neighbors' $\langle lockRound, x\rangle$ message. In order to send such a message, q must have had $x_q = x$ at the beginning of round r', which in turn means that x_q

was either q's initial value, or q has updated x_q after receiving a message in some round $r_q < r$. By repeating this argument, we will eventually reach a process that sent its initial value, since no process can have updated its decision estimate prior to the first round.

The following Lemma 11 states a number of properties maintained by our algorithm when the first process p has decided. Essentially, they say that there has been a vertex-stable root component for at least $2D+1$ rounds centered around the lock round ℓ (but not earlier), and asserts that all processes in that root component chose the same lock round ℓ.

Lemma 11 *Suppose that some process p decides in some round r, no decisions occurred before r, and $\ell = lockRound_p^r$,*[6] *then*

(i) p is in the vertex-stable root component $\mathcal{R}^I$ with $I = [\ell - D - 1, \ell + D]$,
(ii) $\ell + D \leqslant r \leqslant \ell + 2D$,
(iii) $\mathcal{R}^I \neq \mathcal{R}^{\ell-D-2}$, and
(iv) all processes in $\mathcal{R}^I$ executed Line 18 in round ℓ, and no process in $\Pi \setminus \mathcal{R}^I$ can have executed Line 18 in a round $\geqslant \ell$.

Proof Item (i) follows since Line 15 has been continuously TRUE since round ℓ and from Lemma 8. As for item (ii), $\ell + D \leqslant r$ follows from the requirement of Line 20, while $r \leqslant \ell + 2D$ follows from (i) and the fact that by Lemma 9 the requirement of Line 20 cannot be, for the first time, fullfilled strictly after round $\ell + 2D$ (but already by $\ell + 2D$). From Lemma 9, it also follows that if $\mathcal{R}^{\ell-D-2} = \mathcal{R}^I$, then the condition in Line 15 would return true already in round $\ell - 1$, thus locking in round $\ell - 1$. Since p did not lock in round $\ell - 1$, (iii) must hold. Finally, from (i), (iii), and Lemma 9, it follows that every other process in $\mathcal{R}^I$ also has `inStableRoot`$([\ell - D - 1, \ell - D])$ = TRUE in round ℓ. Moreover, due to (iii), `inStableRoot`$([\ell - 1 - D - 1, \ell - 1 - D])$ = FALSE in round $\ell - 1$, which causes all the processes in $\mathcal{R}^I$ (as well as those in $\Pi \setminus \mathcal{R}^I$) to set *lockRound* to 0. Since `inStableRoot`$([\ell' - D - 1, \ell' - D])$ cannot become true for any $\ell' \geqslant \ell$ at a process $q \in \Pi \setminus \mathcal{R}^I$, as $C_q|r = \emptyset$ for any $r \in I$, (iv) also holds.

The following Lemma 12 asserts that if a process decides, then it has successfully imposed its proposal value on all other processes.

Lemma 12 (Identical proposal values) *Suppose that process p decides in Line 21 in round r and that no other process has executed Line 21 before r. Then, for all q, it holds that $x_q^r = x_p^r$.*

[6] We denote the value of a variable v of process p in round r (before the round r computation finishes) as v_p^r; we usually suppress the superscript when it refers to the current round.

Proof Using items (i) and (iv) in Lemma 11, we can conclude that p was in the vertex-stable root component $\mathcal{R}$ of rounds $\ell = lockRound_p^r$ to $\ell + D$ and that all processes in $\mathcal{R}$ have locked in round ℓ. Therefore, in the interval $[\ell, \ell + D]$, ℓ is the maximal value of *lockRound*. More specifically, all processes q in $\mathcal{R}$ have $lockRound_q = \ell$, whereas all processes s in $\Pi \setminus \mathcal{R}$ have $lockRound_s < \ell$ during these rounds by Lemma 11.(iv).. Let $m \in \mathcal{R}$ have the largest proposal value $x_m^\ell = x_{max}$ among all processes in $\mathcal{R}$. Since m is in $\mathcal{R}$, there is a causal chain of length at most D from m to any $q \in \Pi$. Since no process executed Line 21 before round r, no process will send DECIDE messages in $[\ell, \ell+D]$. Thus, all processes continue to execute the update rule of Line 14, which implies that x_{max} will propagate along the aforementioned causal path to q.

Theorem 1 *Let r_{ST} be the first round where Assumption 1 holds. Algorithm 2 in conjunction with Algorithm 1 solves consensus by round $r_{ST} + 4D + 1$.*

Proof Validity holds by Lemma 10. Considering Lemma 12, we immediately get Agreement: Since the first process p that decides must do so via Line 21, there are no other proposal values left in the system.

Observe that, so far, we have not used the liveness part of Assumption 1. In fact, Algorithm 2 is always safe in the sense that Agreement and Validity are not violated, even if there is no vertex-stable root component.

We now show the Termination property. By Corollary 2, we know that every process in $p \in \mathcal{R}$ evaluates the predicate `inStableRoot`$([r_{ST}, r_{ST} + 1])$ = TRUE in round $\ell = r_{ST} + D + 1$, thus locking in that round. Furthermore, Assumption 1 and Corollary 2 imply that at the latest in round $d = \ell + 2D$ every process $p \in \mathcal{R}$ will evaluate the condition of Line 20 to TRUE and thus decide using Line 21. Thus, every such process p will send out a message $m = \langle \text{DECIDE}, x_p \rangle$. By the definition of D and Assumption 1, we know that the round d network causal diameter satisfies $D^d(\Pi) \leqslant D$, such that every $q \in \Pi$ will receive the DECIDE message at the latest in round $d + D = \ell + 3D = r_{ST} + 4D + 1$.

6.2 Improved Time Complexity

We now discuss how to guarantee a smaller causal network diameter, by considering additional properties on the topologies of the network. We will use expander graphs, which are a frequently used technique for designing robust and fault-tolerant networks (e.g. [10]). An (undirected) graph $\mathcal{G}$ is an *α-vertex expander* if, for all sets $S \subset V$ of size $\leqslant |\mathcal{G}|/2$, it holds that $\frac{|\mathcal{N}(S)|}{|S|} \geqslant \alpha$,

Algorithm 2 Solving Consensus; code for process p

Simultaneously run Algorithm 1.

Variables and Initialization:

$x_p \in \mathbb{N}$ initially v_p
$locked_p, decided_p \in \{\texttt{false}, \texttt{true}\}$ initially `false`
$lockRound_p \in \mathbb{Z}$ initially 0

Emit round r messages:

if $decided_p$ **then**
  send $\langle \text{DECIDE}, x_p \rangle$ to all neighbors
else
  send $\langle lockRound_p, x_p \rangle$ to all neighbors

Round r computation:

if not $decided_p$ **then**
  if received $\langle \text{DECIDE}, x_q \rangle$ from any neighbor q **then**
    $x_p \leftarrow x_q$
    decide on x_p and set $decided_p \leftarrow \texttt{true}$
  else // p only received $\langle lock_q, x_q \rangle$ messages (if any):
    $(lockRound_p, x_p) \leftarrow \max\left\{(lock_q, x_q) \mid q \in \mathcal{N}_p^r \cup \{p\}\right\}$ // lexical order in max
    if `inStableRoot`$([r-D-1, r-D])$ **then**
      if (**not** $locked_p$) **then**
        $locked_p \leftarrow \texttt{true}$
        $lockRound_p \leftarrow r$
      else
        if `inStableRoot`$([lockRound_p, lockRound_p + D])$ **then**
          decide on x_p and set $decided_p \leftarrow \texttt{true}$
    else // `inStableRoot`$([r-D-1, r-D])$ returned FALSE
      $locked_p \leftarrow \texttt{false}$

where $\mathcal{N}(S)$ is the set of neighbors of S in $\mathcal{G}$. Such graphs exist and can be constructed explicitly (cf. [16]).

For a vertex set S and a round r, we consider the set $\mathcal{N}_+^r(S)$ of nodes outside of S that are reachable from S and the set of nodes $\mathcal{N}_-^r(S)$ that can reach S in r.

Assumption 2 (Expander Topologies) *Suppose that Assumption 1 holds and consider the corresponding I-vertex stable root component $\mathcal{R}^I$ with interval $I = [b, e]$. For all $r \in I$ and some fixed constant α, the following conditions hold for for all sets $S \subseteq V(\mathcal{G}^r)$:*

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

(b) If $|S| \leqslant n/2$ and $\mathcal{R}^I \subseteq S$, then $\frac{|\mathcal{N}_+^r(S)|}{|S|} \geqslant \alpha$.

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

Lemma 13 *There are sequences of graphs where Assumptions 1 and 2 are satisfied simultaneously and, for any such run, there is an interval I during which there exists an $O(\log n)$-bounded I-vertex stable root component.*

Proof We will first argue that *directed* graphs exist that simultaneously satisfy Assumptions 1 and 2. Consider $r \in I$ and the simple undirected graph $\bar{\mathcal{U}}$ that is the union of an α-vertex expander on $\mathcal{R}^I$ and an α-vertex expander on $V(\mathcal{G}^r)$. Replacing every edge $e \in E(\bar{\mathcal{U}})$ with 2 oppositely oriented directed edges guarantees Properties (a)-(c), however, Assumption 1 might not hold for I, since $\mathcal{R}^I$ will no longer be a root component if $|\mathcal{R}^I| < n$. To remedy this, we drop all directed edges pointing to $\mathcal{R}^I$ from the remaining graph, which leaves the graph weakly connected and Properties (a)-(c) intact. We stress that the actual topologies chosen by the adversary might be quite different from this construction, which merely serves us to show the existence of such graphs.

We prove that Assumption 2 gives a network causal diameter in $O(\log n)$. For $i \geqslant 1$, let $\mathcal{P}^i \subseteq \mathcal{R}^I$ be the set of processes q in $\mathcal{R}^I$ such that $(p \overset{b[i]}{\rightsquigarrow} q)$, and $\mathcal{P}^0 = \{p\}$. We first show that $D^I(\mathcal{R}^I) \in O(\log n)$. The result is trivial if $|\mathcal{R}^I| \in O(\log n)$, thus assume that $|\mathcal{R}^I| \in \Omega(\log n)$ and consider some process $p \in \mathcal{R}^I$. For round b, Property (a) yields $|\mathcal{P}^1| \geqslant |\mathcal{P}^0|(1+\alpha)$. In fact, for all i where $|\mathcal{P}^i| \leqslant |\mathcal{R}^I|/2$, we can apply Property (a) to get $|\mathcal{P}^{i+1}| \geqslant |\mathcal{P}^i|(1+\alpha)$, hence $|\mathcal{P}^i| \geqslant (1+\alpha)^i$. Let k be the smallest value such that $|\mathcal{P}^{k+1}| > |\mathcal{R}^I|/2$. By the above, this is true for any $k > \frac{\log(|\mathcal{R}^I|/2)}{\log(1+\alpha)} \in O(\log n)$. Now consider any q at round $b+2k$ and define $\mathcal{Q}^{i-1} \subset \mathcal{R}^I$ as the set of nodes that causally influence the set $\mathcal{Q}^i$ in round $b+i$, for $\mathcal{Q}^{2k} = \{q\}$. Again, by Property (a), we get $|\mathcal{Q}^{i-1}| \geqslant |\mathcal{Q}^i|(1+\alpha)$, so $|\mathcal{Q}^i| \geqslant (1+\alpha)^{2k-i}$ for any i where $|\mathcal{Q}^{i-1} \leqslant |\mathcal{R}^I|/2$ and hence $|\mathcal{Q}^k| > |\mathcal{R}^I|/2$. Since $\mathcal{P}^{k+1} \cap \mathcal{Q}^k \neq \emptyset$, it follows that every $p \in \mathcal{R}^I$ influences every $q \in \mathcal{R}^I$ within $2k \in O(\log n)$ rounds.

Finally, to see that $D^I \in O(\log n)$, we use Property (b): At any round $r \in [b, e-2k'+1]$, we know that any process $p \in \mathcal{R}^I$ has influenced at least $n/2$ nodes by round $r+k'$ where $k' > \log_{1+\alpha}(n/2) \in O(\log n)$. Property (c) allows us to reason along the same lines as for the sets $\mathcal{Q}^{i-1}$ above. That is, p will influence every $q \in \Pi$ by round $r+2k'$, which completes the proof.

Corollary 3 *Suppose that Assumptions 1 and 2 hold. Then, running Algorithms 1 and 2 solves consensus by round $r_{ST} + O(\log n)$.*

7 Impossibilities and Lower Bounds

In this section, we will prove that our basic Assumption 1, in particular, the existence of a stable window (of a certain minimal size) and the knowledge of an upper bound D on the causal network diameter, are crucial for making consensus solvable. Moreover, we will show that it is not unduly strong, as many problems considered in distributed systems in general (and dynamic networks in particular) remain unsolvable.

First, we relate our Assumption 1 to the classification of [7]. Lemma 14 reveals that it is stronger than one of the two weakest classes, but also weaker than the next class.

Lemma 14 (Properties of root components) *Assume that there is at most one root component $\mathcal{R}^r$ in every $\mathcal{G}^r$, $r > 0$. Then, (i) there is at least one process p such that $\mathtt{cd}^1(p,q)$ is finite for all $q \in \Pi$, and this causal distance is in fact at most $n(n-2)+1$. Conversely, for $n > 2$, the adversary can choose topologies where (ii) no process p is causally influenced by all other processes q, i.e., $\nexists p\ \forall q\colon \mathtt{cd}^1(q,p) < \infty$.*

Proof Since we have infinitely many rounds in a run but only finitely many processes, there is at least one process p in $\mathcal{R}^r$ for infinitely many r. Let $r_1, r_2, \ldots$ be this sequence of rounds. Moreover, let $\mathcal{P}_0 = \{p\}$, and define for each $i > 0$ the set $\mathcal{P}_i = \mathcal{P}_{i-1} \cup \{q : \exists q' \in \mathcal{P}_{i-1} : q' \in \mathcal{N}_q^{r_i}\}$.

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

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

The converse statement (ii) follows directly from considering a static star, for example, i.e., a communication graph where there is one central process c, and for all r, $\mathcal{G}^r = \langle \Pi, \{c \to q | q \in \Pi \setminus \{c\}\}\rangle$. Clearly, c cannot be causally influenced by any other process, and $q \not\rightsquigarrow q'$ for any $q, q' \neq q \in \Pi \setminus \{c\}$. On the other hand, this topology satisfy Assumption 1, which includes the requirement of at most one root component per round.

Next, we examine the solvability of several broadcast problems under Assumption 1. Although there is a strong bond between some of these problems and consensus in traditional settings, they are *not* implementable under our assumptions—basically, because there is no guarantee of (eventual) bidirectional communication.

We first consider reliable broadcast, which requires that when a correct process broadcasts m, every correct process eventually delivers m. Suppose that the adversary chooses the topologies $\forall r:\ \mathcal{G}^r = \langle \{p,q,s\}, \{p \to q, q \to s\}\rangle$, which matches Assumption 1. Clearly, q is a correct process in our model. Since p never receives a message from q, p can trivially never deliver a message that q broadcasts.

We now turn our attention to the various problems considered in [18], which are all impossible to solve under Assumption 1. More specifically, we return to the static star considered in the proof of Lemma 14. Clearly, the local history of any process is independent of the size n. Therefore, the problems of counting, k-verification, and k-committee election are all impossible. For the token dissemination problems, consider that there is a token that only $p \neq c$ has. Since no other process ever receives a message from p, token dissemination is impossible.

Theorem 2 *Suppose that Assumption 1 is the only restriction on the adversary in our model. Then, neither* reliable broadcast*, *atomic broadcast*, nor* causal-order broadcast *can be implemented. Moreover, there is no algorithm that solves* counting*,* k-verification*,* k-token dissemination*,* all-to-all token dissemination*, and* k-committee election*.*

7.1 Knowledge of a Bound on the Network Causal Diameter

Theorem 3 *Consider a system where Assumption 1 holds and suppose that processes do not know an upper bound D on the network causal diameter (and hence do not know n). Then, there is no deterministic algorithm that solves consensus.*

Proof Assume for the sake of a contradiction that there is such an algorithm $\mathcal{A}$. Consider a run $\alpha(v)$ of $\mathcal{A}$ on a communication graph G that forms a (very large) static directed line rooted by process p and ending in process q. Process p has initial value $v \in \{0,1\}$, all other processes have initial value 0. Clearly, algorithm $\mathcal{A}$ must allow p to decide on v by the end of round κ, where κ is a constant (independent of D and n; clearly, we assume that n is large enough to guarantee $n-1 > \kappa$). Next, consider a run $\beta(v)$ of $\mathcal{A}$ that has the same initial states as $\alpha(v)$, and communication graphs $(H_r)_{r>0}$

that, during rounds $[1, \kappa]$, are also the same as in $\alpha(v)$ (defining what happens after round κ will be defered). In any case, since $\alpha(v)$ and $\beta(v)$ are indistinguishable for p until its decision round κ, it must also decide v in $\beta(v)$ at the end of round κ.

However, since $n > \kappa + 1$, q has not been causally influenced by p by the end of round κ. Hence, it has the same state $S_p^{\kappa+1}$ both in $\beta(v)$ and in $\beta(1-v)$. As a consequence, it cannot have decided by round κ: If q decided v, it would violate agreement with p in $\beta(1-v)$. Now assume that run $\beta(.)$ is actually such that the stable window occurs later than round κ, i.e., $r_{ST} = \kappa + 1$, and that the adversary just swaps the direction of the line then: For all H^k, $k \geqslant \kappa+1$, q is the root and p is the last process of the resulting topology. Observe that the resulting $\beta(v)$ still satisfies Assumption 1, since q itself forms the only root component. Now, q must eventually decide on some value v' in some later round κ', but since q has been in the same state at the end of round κ in both $\beta(v)$ and $\beta(1-v)$, it is also in the same state in round κ' in both runs. Hence, its decision contradicts the decision of p in $\beta(1-v')$.

It is straightforward to see that Theorem 3 also holds for any randomized (Las Vegas) algorithm, i.e., a randomized algorithm that achieves consensus in every run. For this, we consider a variant of the state transition function (cf. Section 3) that, in every round, additionally takes as input the outcome of some finite number of unbiased coin flips for determining the successor state. We can adapt the above proof by replacing κ with the earliest round where p decides with probability $\pi > 0$; by Validity, p's possible decision must be v. Since $\alpha(v)$ and $\beta(v)$ are indistinguishable for p, the probability that it also decides (on v) in $\beta(v)$ is exactly π. However, q cannot have a nonzero probability to decide on any value, as deciding on v' in $\beta(1-v')$ would violate agreement.

Corollary 4 *Assume that Assumption 1 holds and suppose that processes do not know an upper bound on the network causal diameter. Then, there is no randomized algorithm that solves consensus in every run.*

7.2 Impossibility of Leader Election with Unknown Network Causal Diameter

We now use a more involved indistinguishability argument to show that a weaker problem than consensus, namely, deterministic leader election is also impossible to solve in our model. The classic leader election problem (cf. [22]) assumes that, eventually, exactly one process irrevocably elects itself as leader (by entering a special ELECTED state) and every other process elects itself as non-leader (by entering the NON-ELECTED state). Non-leaders are not required to know the process id of the leader.

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

Theorem 4 *Consider a system where Assumption 1 holds and suppose that processes do not know an upper bound D on the network causal diameter (and hence do not know n). Then, there is no deterministic or randomized (Las Vegas) algorithm that solves leader election.*

Proof Consider the execution $\alpha(w, \mathcal{G})$ of $\mathcal{A}$ in a static unidirectional line (i.e. $\forall r \geqslant 1 : \mathcal{G}^r = \mathcal{G}$), headed by process p with id w: Since p has only a single out-going edge and does not know n, it cannot know whether it has neighbors at all. As it is alone in the single-vertex graph consisting of p only, it must elect itself as leader in any $\alpha(w, \mathcal{G})$, after some $T(w)$ rounds (we do not restrict $\mathcal{A}$ to be time-bounded).

Let w and z be two arbitrary different process ids, and let $T(w)$ resp. $T(z)$ be the termination times in the execution $\alpha(w, \mathcal{G})$ resp. $\alpha(z, \mathcal{G})$ on the line; let $T = \max\{T(w), T(z)\}$.

We now build a system consisting of $n = 2T + 3$ processes arranged in a clockwise unidirectional ring, assembled from a line $\mathcal{G}_p$ of $T + 1$ processes headed by p (with id w), a process r, and a line $\mathcal{G}_q$ of $T + 1$ processes headed by q (with id z) and ending in process s.

Now consider an execution β, which proceeds as follows: For the first T rounds, the communications graph is the clockwise unidirectional ring; its root component is hence the entire ring. Starting from round $T + 1$ on, process r forms the single vertex root component, which feeds the two lines G_q (in clockwise direction) and G_p (in counter-clockwise direction). Note that there is neither edge $(p \to s)$ nor $(s \to p)$ from round $T + 1$ on.

Let ℓ be the process that is elected leader in β. We distinguish 2 cases:

1. If $\ell \in \mathcal{G}_q \cup \{r\}$, then consider the execution β_p that is exactly like β, except that there is no edge $(s \to p)$ during the first T rounds: p with id w is the single root component here. Clearly, for p, the execution β_p is indistinguishable from $\alpha(w, \mathcal{G}_p)$ during the

first $T(w) \leqslant T$ rounds, so it must elect itself leader. However, since no process t in $\mathcal{G}_q \cup \{r\}$ (including $t = \ell$) is causally influenced by p during the first T rounds, t has the same state after round T (and all later rounds) in β_p as in β. Consequently, ℓ also elects itself leader in β_p as it does in β, which is a contradiction.

2. On the other hand, if $\ell \in \mathcal{G}_p$, we consider the execution β_q, which is exactly like β, except that there is no edge $(r \to q)$ during the first T rounds: q with id z is the single root component here. Clearly, for q, the execution β_q is indistinguishable from $\alpha(z, \mathcal{G}_q)$ during the first $T(z) \leqslant T$ rounds, so it must elect itself leader. However, since no process t in $\mathcal{G}_p \cup \{r\}$ (including $t = \ell$) is causally influenced by q during the first T rounds, t has the same state after round T (and all later rounds) in β_q as in β. Consequently, ℓ also elects itself leader β_q as it does in β, which is again a contradiction.

Analogously to Corollary 4, we can extend the above argument to randomized (Las Vegas) algorithms by defining $T(w)$ resp. $T(z)$ to be the first point in time when p enters the ELECTED state with nonzero probability. This completes the proof of Theorem 4.

7.3 Impossibility of consensus with too short intervals

Our goal in this section is to show that it is necessary to have root components that are vertex stable long enough to flood the network. That is, w.r.t. Assumption 1, we need I to be in the order of D. To this end, we first introduce the following alternative Assumption 3, which requires a window of only D:

Assumption 3 *For any round r, there is exactly one root component $\mathcal{R}^r$ in $\mathcal{G}^r$. Moreover, there exists a D and an interval of rounds $I = [r_{ST}, r_{ST} + D]$, such that there is an I-vertex stable root component $\mathcal{R}^I$, such that $D^I \leqslant D$.*

In order to show that Assumption 3 is necessary, we further shorten the interval: Some process could possibly not be reached within $D - 1$ rounds, but would be reached if the interval was D rounds. Processes could hence *withold* information from each other, which causes consensus to be impossible [25]. To simplify the proofs, we consider a stronger variant, where there is exactly one such process q, that is not reached within $D - 1$ rounds, from any process in $\mathcal{R}^I$. Note that, since it is exactly one, we have that in executions where the interval is actually D rounds, it will be reached. Thus we consider the following assumption:

Assumption 4 *For any round r, there is exactly one root component $\mathcal{R}^r$ in $\mathcal{G}^r$. Moreover, there exists a D and an interval of rounds $I = [r_{ST}, r_{ST} + D - 1]$, such that there is an I-vertex stable root component $\mathcal{R}^I$, and there exists a unique $q \in \Pi$ such that $\forall p \in \mathcal{R}^I, \forall r \in I : \mathtt{cd}^r(p, q) \geqslant D$, while for all $q' \in \Pi \setminus \{q\}$ we have $\forall p \in \mathcal{R}^I, \forall r \in I : \mathtt{cd}^r(p, q') \leqslant D - 1$.*

Note that, those executions that fulfill Assumption 1, and where $D^I = D$ also fulfill Assumption 4. But the latter also allows executions where the longest interval with a vertex stable interval is (much) shorter than $4D$.

In the sequel we assume that the adversary has to fix the start of I and the set of processes in the root component $\mathcal{R}^r$ of every round r before the beginning of the execution. every round r before the beginning of the execution. Note that this does not strengthen the adversary, and hence does not weaken our impossibility result.

Lemma 15 *Assume some fixed I and $\mathcal{R}^I$, such that Assumption 4 holds. If two univalent configurations C' and C'' at the beginning of round r differ only in the state of one process p, they cannot differ in valency.*

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

Our choice of the round graphs depends on the following two cases: (i) p is in $\mathcal{R}^r$ and $r \in I$ or (ii) otherwise. In the second case, we assume that the adversary choses $\mathcal{R}^s = \{q\} \neq \{p\}$ for all rounds $s \geqslant r$. We can thus consider a sequence of graphs $\mathcal{G}^s$, for $s \geqslant r$, such that $\mathtt{cd}^s(q, p) = D$ which obviously fulfills Assumption 4. But for q (and all other processes except p), e' and e'' are indistinguishable.

In case (i), by our Assumption 4, there is some q, such that the information that p sends in round r does not arrive at some specific q within $I = [a, b]$. Now assume that the adversary choses $\mathcal{R}^s = \{q\}$ for all $s > b$. Clearly, for process q, the sequence of states in the extension starting from C' and C'' is the same. Therefore, the two runs are indistinguishable to q also in this case.

Lemma 16 *Consider a round r configuration C, then for any two round r graphs $\mathcal{G}'$ and $\mathcal{G}''$, there is a k such that we can find a sequence of graphs $\mathcal{G}', \mathcal{G}_1, \ldots \mathcal{G}_i \ldots \mathcal{G}''$*

each with a single root component, where any two consecutive graphs differ only by at most one edge. Moreover, if $\mathcal{G}'$ and $\mathcal{G}''$ have the same root component $\mathcal{R}$ so do all $\mathcal{G}_i$.

Proof First, we consider two cases with respect to the root components $\mathcal{R}'$ and $\mathcal{R}''$: (a) $\mathcal{R}' \cap \mathcal{R}'' = \emptyset$, (b) $\mathcal{R}' \cap \mathcal{R}'' \neq \emptyset$. Moreover, for the second part of the proof, we also consider a special case of (b): (b') $\mathcal{R}' = \mathcal{R}''$.

For case (b) (and thus also for (b')), we consider $\mathcal{G}_1 = \mathcal{G}'$. For case (a), we construct $\mathcal{G}_1$ from $\mathcal{G}'$ as follows: Let $p' \in \mathcal{R}'$ and $p'' \in \mathcal{R}''$, then $\mathcal{G}_1$ has the same edges as $\mathcal{G}'$ plus $a = p'' \to p'$, thus $\mathcal{R}_1 \supseteq \mathcal{R}' \cup \{p''\}$ (recall that p'' must be reachable from $\mathcal{R}'$ already in $\mathcal{G}'$). So, now we have that in both cases $\mathcal{G}'$ and $\mathcal{G}_1$ differ in at most one edge. Moreover, there is a nonempty intersection between $\mathcal{R}_1$ and $\mathcal{R}''$.

In the first phase of our construction (which continues as long as $E'' \setminus E_i \neq \emptyset$), we construct $\mathcal{G}_{i+1}$ from $\mathcal{G}_i$, $i \geqslant 1$, by choosing one edge $e = (v \to w)$ from $E'' \setminus E_i$ and let $\mathcal{G}_{i+1}$ have the same edges as $\mathcal{G}_i$ plus e. Clearly, $\mathcal{G}_i$ and $\mathcal{G}_{i+1}$ differ in at most one edge. Moreover, when adding an edge, we cannot add an additional root component, so as long as we add edges we will have that $\mathcal{G}_{i+1}$ has a single root component $\mathcal{R}_{i+1} \supset R'$.

When we reach a point in our construction where $E'' \setminus E_i = \emptyset$, the first phase ends. As $\mathcal{G}_i$ now contains all the edges in $\mathcal{G}''$, i.e., $E_i \supset E''$ and we have $\mathcal{R}_i \supset R''$. In the second phase of the construction, we remove edges. To this end, we choose one edge $e = (v \to w)$ from $E_i \setminus E''$, and construct $\mathcal{G}_{i+1}$ from $\mathcal{G}_i$ by removing e. Again we have to show that there is only one root component. Since we never remove an edge in E'', $\mathcal{G}_i$ always contains a directed path from some $x \in \mathcal{R}''$ to both v and w that only uses edges in E''. As $e \notin E''$, this also holds for $\mathcal{G}_{i+1}$. Since there is only one root component in $\mathcal{G}''$, this implies that there is only one in $\mathcal{G}'$.

Let $\mathcal{G}_j$ be the last graph constructed in the first phase, and $\mathcal{G}_k$ the last graph constructed in the second phase. It is easy to see that $E_k = E_j \setminus (E_j \setminus E'')$, which implies that $E_k = E''$ and hence $\mathcal{G}_k = E''$. This completes the proof of the first part of the Lemma.

To see that the second part also holds, we consider case (b') in more detail and show by induction that $\mathcal{R}_{i+1} = \mathcal{R}_i = \mathcal{R}$. For the base case we recall that $\mathcal{G}_1 = G'$ and thus $\mathcal{R}_1 = R'$. For the induction step, we consider first that the step involves adding an edge $e = (v \to w)$. Adding an edge can only modify the root component when $v \notin \mathcal{R}_i$ and $w \in \mathcal{R}_i$. Since such an edge e is not in E'' (as it has the same root component as E'), we cannot select it for addition, so the root component does not change. If on the other hand the step from $\mathcal{G}_i$ to $\mathcal{G}_{i+1}$ involves removing the edge $e = (v \to w)$, then we only need to consider the case where $v \in \mathcal{R}_i$. (If $v \notin \mathcal{R}_i$, then also $w \notin \mathcal{R}_i$ so the root component cannot change by removing e.) But since we never remove edges from E'', this implies that even after removing e there is still a path from v to w, so the root component cannot have changed.

Theorem 5 *Assume that Assumption 4 is the only requirement for the graph topologies. Then consensus is impossible.*

Proof We follow roughly along the lines of the proof of [25, Lemma 3] and show per induction on the round number, that an algorithm A cannot reach a univalent configuration until round r.

For the base case, we consider binary consensus only and argue similar to [12] but make use of our stronger Validity property: Let C_x^0 be the initial configuration, where the processes with the x smallest ids start with 1 and all others with 0. Clearly, in C_0^0 all processes start with 0 and in C_n^0 all start with 1, so the two configurations are 0- and 1-valent, respectively. To see that for some x C_x^0 must be bivalent, consider that this is not the case, then there must be a C_x^0 that is 0-valent while C_{x+1}^0 is 1-valent. But, these configurations differ only in p_{x+1}, and so by Lemma 15 they cannot be univalent with different valency.

For the induction step we assume that there is a bivalent configuration C at the beginning of round $r-1$, and show that there is at least one such configuration at the beginning of round r. We proceed by contradiction and assume all configurations at the beginning of round r are univalent. Since C is bivalent and all configurations at the beginning of r are univalent, there must be two configurations C' and C'' at the beginning of round r which have different valency. Clearly, C' and C'' are reached from C by two different round $r-1$ graphs $\mathcal{G}' = \langle \Pi, E' \rangle$ and $\mathcal{G}'' = \langle \Pi, E'' \rangle$. Lemma 16 shows that there is a sequence of graphs that can be applied to both C' and C''. Further, note that each pair of subsequent graphs in this sequence differ only in one link $v \to w$, so the resulting configurations differ only in the state of w. Moreover, if the root component in $\mathcal{G}'$ and $\mathcal{G}''$ is the same, all graphs in the sequence also have the same root component. Since the valency of C' and C'' was assumed to be different, there must be two configurations $\overline{C}'$ and $\overline{C}''$ in the sequence of configurations that have different valency and differ only in the state of one process, say p. Applying Lemma 15 to $\overline{C}'$ and $\overline{C}''$ again produces a contradiction, and so not all successors of C can be univalent.

8 Conclusion

We introduced a novel framework for modeling dynamic networks with directed communication links, and introduced a weak connectivity assumption that makes consensus solvable. Without such assumptions, consensus is trivially impossible in such systems as some processes can withhold their input values until a wrong decision has been made. We presented an algorithm that achieves consensus under this assumption, and showed several impossibility results and lower bounds that reveal that our algorithm is asymptotically optimal.

References

1. Yehuda Afek, Eli Gafni, and Adi Rosen. The slide mechanism with applications in dynamic networks. In *ACM PODC*, pages 35–46, 1992.
2. John Augustine, Gopal Pandurangan, Peter Robinson, and Eli 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.
3. Baruch Awerbuch, Boaz Patt-Shamir, David Peleg, and Michael E. Saks. Adapting to asynchronous dynamic networks. In *STOC'92*, pages 557–570, 1992.
4. Martin Biely, Bernadette Charron-Bost, Antoine Gaillard, Martin Hutle, André Schiper, and Josef Widder. Tolerating corrupted communication. In *Proceedings of the 26th ACM Symposium on Principles of Distributed Computing (PODC'07)*, pages 244–253. ACM, August 2007.
5. Martin Biely, Peter Robinson, and Ulrich Schmid. Solving k-set agreement with stable skeleton graphs. In *IPDPS Workshops*, pages 1488–1495, 2011.
6. Martin Biely, Ulrich Schmid, and Bettina Weiss. Synchronous consensus under hybrid process and link failures. *Theoretical Computer Science*, 412(40):5602 – 5630, 2011.
7. Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. In Hannes Frey, Xu Li, and Stefan Rührup, editors, *ADHOC-NOW*, volume 6811 of *Lecture Notes in Computer Science*, pages 346–359. Springer, 2011.
8. Bernadette Charron-Bost and André Schiper. The Heard-Of model: computing in distributed systems with benign faults. *Distributed Computing*, 22(1):49–71, April 2009.
9. Hyun Chul Chung, Peter Robinson, and Jennifer 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. ACM, 2011.
10. Cynthia Dwork, David Peleg, Nicholas Pippenger, and Eli Upfal. Fault tolerance in networks of bounded degree. *SIAM J. Comput.*, 17(5):975–988, 1988.
11. Tzilla Elrad and Nissim Francez. Decomposition of distributed programs into communication-closed layers. *Science of Computer Programming*, 2(3):155–173, 1982.
12. Michael J. Fischer, Nancy A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, April 1985.
13. Eli 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. ACM Press, 1998.
14. Alois Goiser, Samar Khattab, Gerhard Fassl, and Ulrich 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, June 2010.
15. F. Harary and G. Gupta. Dynamic graph models. *Mathematical and Computer Modelling*, 25(7):79 – 87, 1997.
16. Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(04):439–562, August 2006.
17. F. Kuhn and R. Oshman. Dynamic networks: Models and algorithms. *SIGACT News*, 42(1):82–96, 2011.
18. Fabian Kuhn, Nancy Lynch, and Rotem Oshman. Distributed computation in dynamic networks. In *ACM STOC*, pages 513–522, 2010.
19. Fabian Kuhn, Rotem Oshman, and Yoram Moses. Coordinated consensus in dynamic networks. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, PODC '11. ACM, 2011.
20. Fabian Kuhn, Stefan Schmid, and Roger Wattenhofer. Towards worst-case churn resistant peer-to-peer systems. *Distributed Computing*, 22(4):249–267, 2010.
21. Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. *Commun. ACM*, 21(7):558–565, 1978.
22. Nancy Lynch. *Distributed Algorithms*. Morgan Kaufman Publishers, Inc., San Francisco, USA, 1996.
23. David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Mathematics ans Applications, Philadelphia, 2000.
24. Nicola Santoro and Peter 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, February 1989. Springer-Verlag.
25. Ulrich Schmid, Bettina Weiss, and Idit Keidar. Impossibility results and lower bounds for consensus under link failures. *SIAM Journal on Computing*, 38(5):1912–1951, 2009.
26. 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.