FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Agreement Algorithms in Directed Dynamic Networks

DISSERTATION

submitted in partial fulfillment of the requirements for the degree of

Doktor der Technischen Wissenschaften

by

DI Manfred Schwarz, B. Sc.
Registration Number 0725898

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Prof. Ulrich Schmid

The dissertation has been reviewed by:

Emmanuel Godard	Jesper Larsson Träff

Vienna, 30th April, 2018

Manfred Schwarz

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Agreement Algorithms in Directed Dynamic Networks

DISSERTATION

zur Erlangung des akademischen Grades

Doktor der Technischen Wissenschaften

eingereicht von

DI Manfred Schwarz, B. Sc.
Matrikelnummer 0725898

an der Fakultät für Informatik
der Technischen Universität Wien
Betreuung: Prof. Ulrich Schmid

Diese Dissertation haben begutachtet:

Emmanuel Godard

Jesper Larsson Träff

Wien, 30. April 2018

Manfred Schwarz

Erklärung zur Verfassung der Arbeit

DI Manfred Schwarz, B. Sc.
Hauptstrasse 163, 3400 Kierling, Österreich

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 30. April 2018

Manfred Schwarz

Acknowledgements

First and foremost, I would like to thank my advisor Ulrich Schmid. He did not only spark my research interest but also channeled it in the direction of distributed algorithms[1], which ultimately led to the topic of this thesis. Furthermore, I am grateful for providing a relaxed but stimulating working environment, which facilitated numerous fruitful discussions with my collaborators and colleagues.

I am also very much in debt to my colleague Kyrill Winkler, with whom I developed the foundations for most of our joint publications and, hence, my and his forthcoming PHD thesis. Without collaborations none of our results would exist.

Furthermore, I would like to thank all the other collaborators and colleagues Martin Perner, Martin Zeiner, Jürgen Maier, Florian Huemer, Matthias Függer, Thomas Nowak and Heinz Deinhart for all the discussions, their feedback on presentations and the meals shared.

Finally, I would like to thank my family and friends for being there when I struggled and always reminding me to take a step back to get a new perspective if problems seemed insurmountable.

[1]The work received funding from the FWF projects ADynNet (P28182), RISE (S11405) and SIC (P26436).

Danksagung

Zuallererst möchte ich mich bei meinem Betreuer Ulrich Schmid bedanken. Nicht nur dafür dass er mein Interesse an der Forschung geweckt hat, sondern auch, dass er mich an das Thema Verteilte Algorithmen[2] herangeführt hat, welche jetzt Gegenstand meiner Arbeit sind. Des Weiteren möchte ich Ihm dafür danken, dass er immer bemüht war und ist ein entspanntes und inspirierendes Arbeitsumfeld zu schaffen, welches zahlreiche fruchtvolle Diskussionen mit meinen Kollegen stimuliert hat.

Ich bin außerdem sehr dankbar für die Zusammenarbeit mit meinem Kollegen Kyrill Winkler, mit dem ich die Basis für die meisten unserer gemeinsamen Publikationen und daher letztlich für unsere beiden Dissertationen geschaffen habe. Ohne diese Zusammenarbeit würde keines unserer Resultate existieren.

Des Weiteren möchte ich mich bei meinen anderen Kollegen und Kollaborateuren Martin Perner, Martin Zeiner, Jürgen Maier, Florian Huemer, Matthias Függer, Thomas Nowak und Heinz Deinhart für all die Diskussionen, die Verbesserungsvorschläge für Präsentation und die gemeinsamen Mittagsessen bedanken.

Abschließend möchte ich noch meiner Familie und meinen Freunden danken, dass sie für mich da waren, wenn ich gestrauchelt bin und mich immer daran erinnert haben, dass ein wenig Abstand hilft um eine neue Perspektive auf unüberwindbar scheinende Probleme zu bekommen.

[2]Die Arbeit wurde finanzielle von den FWF Projekten ADynNet (P28182), RISE (S11405) und SIC (P26436) unterstützt.

Abstract

This dissertation explores algorithmic solutions for some prominent agreement problems in the field of distributed computing. Such problems are interesting as they are the basis for many practical distributed systems. Agreement problems include clock synchronization, symmetry breaking or solving coordination problems where participants with conflicting inputs have to agree on a common output. Whereas most of the existing work studies systems where the failure assumption is a crash of one or more participants, this thesis focuses on synchronous dynamic networks with communication failures controlled by a message adversary. With the emergence of wireless systems, ad-hoc networks and sensor networks, systems consisting of large possibly unknown number of network nodes connected by undirected network links become ubiquitous. Results include optimal solutions for the consensus problem, a gracefully degrading solution for the more general k-set agreement problem, and lower bounds for the asymptotic consensus problem. The thesis closes with relating synchronous systems subject message adversaries to asynchronous systems with failure detectors, which inspired the use of a suitable definition of message adversary simulations.

Kurzfassung

Diese Dissertation erforscht algorithmische Lösungen für wichtige Konsensusprobleme in verteilten Systemen. Diese Probleme sind besonders interessant, da sie die Basis für viele praktische Anwendungen in verteilten Systeme darstellen. Beispiele sind Uhren synchronisation oder das Lösen von Symmetrie- und Koordinationsproblemen bei denen Teilnehmer mit widersprüchlichen Eingangswerten sich auf einen gemeinsamen Ausgangswert einigen müssen. Da die bereits existierende Literatur hauptsächlich Fehlermodelle mit ausfallenden Teilnehmern studieren, beschäftigt sich diese Arbeit mit synchronen dynamischen Netzwerken mit Kommunikationsfehlern, die von einem Message Adversary kontrolliert werden. Mit dem vermehrten Auftreten von wireless Systemen, ad-hoc Netzwerken und Sensornetzwerken werden Systeme, die aus einer großen einer möglicherweise unbekannten Anzahl von Teilnehmern und über gerichtete, unzuverlässige Datenverbindungen kommunizieren immer häufiger. Die Resultate umfassen optimale Lösungen für das Consensus Problem, eine degradierende Lösunge für das allgemeinere k-set Agreement Problem und untere Schranken für das Asymptotic Consensus Problem. Die Arbeit schließt mit der Untersuchung einer Beziehung zwischen synchronen Systemen unter Message Adversaries und asynchronen Systemen mit Fehlerdetektoren, die schlussendlich zu einer sinnvollen Definition von Message Adversary-Simulation geführt hat.

Contents

CHAPTER 1

Introduction

In this dissertation, we will explore agreement problems in directed dynamic networks. Agreement problems in general are interesting as they are at the basis for many distributed systems, with the most prominent example being clock synchronization. Other forms are needed for symmetry breaking in deadlock scenarios or solving coordination problems in distributed systems, where participants with possible conflicting inputs have to agree on a common output.

Directed dynamic networks (DDNs) abstract both asynchronous and synchronous systems via so called time-varying graphs. This thesis is focused on exploring different avenues in lock-step synchronous DDNs where processes execute their computation synchronously. Therefore, the behavior of the system can be abstracted via communication closed rounds where every round consist of a message sending, message receiving and a computation step. Whether a process communicates with another process or not in a round is determinated by a directed communication graph, which may change from round to round.

Commonly studied agreement problems are consensus, where every process has to decide on the same output, k-set agreement, where the decisions are a set of at most k possible different values, and asymptotic consensus, where the output values have to converge to a common limit. These problems are well researched in systems with reliable communication and process crashes as the sole failure type.

With the emergence of wireless systems, ad-hoc networks and sensor networks, this failure assumption is not sufficient. In those systems, a typical failure is not a crash of some process but rather communication loss between processes, which results in time varying communication or joining and leaving of processes during runtime. Moreover, peculiarities of wireless communication make communication between process asymmetrical. Hence, directed dynamic networks perfectly model communication in such systems. Note that lock-step synchrony is trivially implemented atop of synchronized clocks, which are easily implemented in such systems.

Moreover such systems are particularly in need of algorithms that solve distributed computing problems like consensus or k-set agreement to compensate for the missing central control. Hence models and algorithms for solving agreement problems in DDNs are important.

The formal entity that abstracts message loss in DDNs is called a *message adversary* (MA), and solving any kind of agreement problem can be seen as a two player game between the algorithm (and the algorithm designer) and the message adversary. Obviously, without any restriction of the adversary, no non-trivial distributed computing problem can be solved: After all, it could

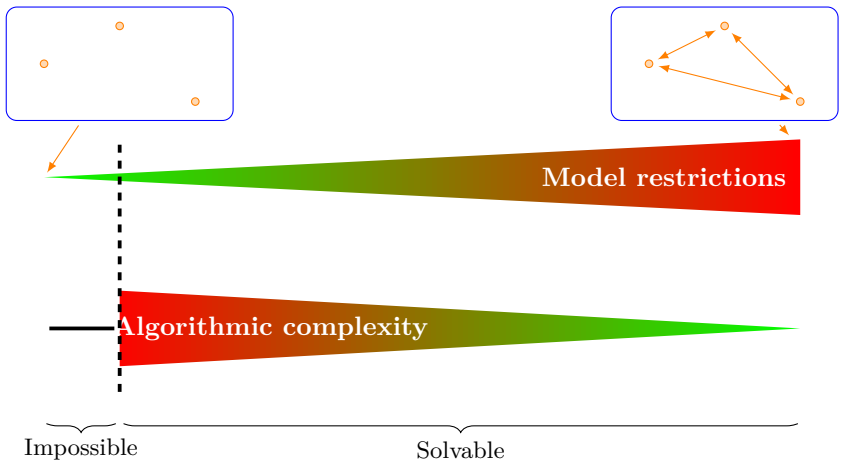


Figure 1.1: The graphic depicts the trade-off between restricting the possible communication graphs and the complexity of a solution algorithm. The dashed line represents the point where finding a solution is impossible. The two graphs on either end symbolize the extreme cases of possible communication graphs for a DDN with 3 process.

just prevent any communication in the system. In addition, more difficult distributed computing problems usually require more severe restrictions of the adversary w.r.t. its ability to prevent communication between processes. On the other hand, there is usually a trade-off between model coverage and algorithmic complexity, as depicted in Figure 1. The model coverage quantifies how well a MA captures the possible behaviors of a real model.

This thesis is structured along increasingly more complex types of message adversaries, and will present results and approaches for solving different instances of agreement problems under these MAs.

Outline of the thesis.

This thesis is based on the the following papers:

- M. Schwarz, K. Winkler, U. Schmid, M. Biely, P. Robinson. (2014). Brief announcement: Gracefully degrading consensus and k-set agreement under dynamic link failures. Proceedings of the Annual ACM Symposium on Principles of Distributed Computing. . 10.1145/2611462.2611506.
- M. Biely, P. Robinson, U. Schmid, M. Schwarz, and K. Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. In Revised selected papers Third International Conference on Networked Systems (NETYS'15), Springer LNCS 9466, pages 109–124, Agadir, Morocco, 2015. Springer International Publishing.
- M. Biely, P. Robinson, U. Schmid, M. Schwarz, and K. Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. Theoretical Computer Science,volume 726, pages 41-77, 2018.
- M. Schwarz, K. Winkler, and U. Schmid. Fast consensus under eventually stabilizing message adversaries. In Proceedings of the 17th International Conference on Distributed Computing and Networking, ICDCN'16, pages 7:1-7:10, New York, NY, USA, 2016. ACM.

- K. Winkler, M. Schwarz, and U. Schmid. Consensus in directed dynamic networks with short-lived stability. CoRR, abs/1602.05852, 2016(submitted).
- M. Függer, T. Nowak, and M. Schwarz. Brief announcement: Lower bounds for asymptotic consensus in dynamic networks. 31st International Symposium on Distributed Computing, DISC 2017,pages 51:1-51:3, October 16-20, 2017, Vienna, Austria
- M. Függer, T. Nowak, and M. Schwarz. Tight Bounds for Asymptotic and Approximate Consensus. ACM Symposium on Principles of Distributed Computing (PODC 2018).
- U. Schmid and M. Schwarz. On the strongest message adversary for consensus in directed dynamic networks (submitted).

Most of the these papers have been written in close collaboration with my colleague Kyrill Winkler. Although it is impossible to clearly identify a main author for every result, it is fair to say that my main contributions (and hence the focus of this thesis) are on the algorithmic side, whereas Kyrill's main contributions (documented in his forthcoming thesis [107]) are mostly on impossibility results.

To relate the different chapters easily to the corresponding paper we will start every chapter with a quick explanation of the chapter structure and reference the paper on which it is based, if possible.

Structure of the thesis.

- In Chapter 2, we start out with a detailed exploration of dynamic networks in general and provide the formal definitions of a model for the DDNs considered in this thesis.
- In Chapter 3, we outline different instances of agreement problems and relations between certain instances. We also define valency and contraction rates which are important tools for arguing about the existence of algorithms.
- In Chapter 4, an overview of related work on both dynamic networks and agreement in DDNs is presented. The publications are sorted according to certain subtopics for which they are most relevant.
- After this prelude we study oblivious message adversaries in Chapter 5. This type of message adversary chooses the communication pattern for each round from a set of predefined graphs. We present results that precisely define the properties that allow to solve two major instances of agreement problems and relate these problems to each other.
- In Chapter 6, we will take the step to non-oblivious message adversaries and we will argue why this is reasonable via the simple yet important example of a two process DDN. Under this type of message adversary we first focus on so called *eventually stable message adversaries*. These MAs distinguish themselves from those presented in the previous section in that some properties emerge at some unknown point in time during a run of the system. The emerging properties always include guaranteed connections from a subset of the processes to the whole system for a certain duration during a run, so-called stable periods/intervals. We will start with non-optimal simple and memory efficient consensus algorithms for MAs that guarantee relatively long periods of stability and then proceed to more complex but optimal solutions for MAs with short stability intervals.

  Moreover, we will show how to generalize adversaries and algorithmic techniques to solve relaxed instances of agreement problems, namely k-set agreement.

- In Chapter 7, we explore how message adversaries in synchronous systems are connected to failure detectors in asynchronous systems, and explore the concept of MA simulations as a similar concept to related different MAs to each other. This relations will also yield a class of strongest message adversaries equal to the established notion of weakest failure detectors.

CHAPTER 2

Directed dynamic networks

In this chapter, we will informally and formally introduce directed dynamic networks. We give a short motivation why research in this area is necessary and present a comprehensive framework to categorize different dynamic network models taken from [32]. The specific DDN model introduced subsequently is based on [22, 57, 98, 102, 108]. Major parts of it will be included in Kyrill Winklers thesis [107] as well, as both our results are founded on the same model.

Dynamic networks such as wireless sensor networks, mobile ad-hoc networks and vehicle area networks, are becoming ubiquitous nowadays. The primary properties of such networks are sets of participants (called processes in the sequel) that are a priori unknown and potentially changing, time-varying connectivity between processes, and the absence of a central control. Dynamic networks is an important and very active area of research [71].

Accurately modelling dynamic networks is challenging, for several reasons: First, process mobility, process crashes/recoveries, deliberate joins/leaves, and peculiarities in the low-level system design like duty-cycling (used to save energy in wireless sensor networks) make static communication topologies, as typically used in classic network models, inadequate for dynamic networks. Certain instances of dynamic networks, in particular, peer-to-peer networks [73] and inter-vehicle area networks [52], even suffer from significant churn, i.e., a large number of processes that can appear/disappear over time, possibly in the presence of faulty processes [10], and hence consist of a potentially unbounded total number of participants over time. More classic applications like *mobile ad-hoc networks* (MANETS) [67], wireless sensor networks [5, 109] and disaster relief applications [74] typically consist of a *bounded* (but typically unknown) total number of processes.

Second, communication in many dynamic networks, in particular, in wireless networks like MANETS, is inherently broadcast: When a process transmits, then every other process within its transmission range will observe this transmission — either by legitimately receiving the message or as some form of interference. This creates quite irregular communication behavior, such as capture effects and near-far problems [106], where certain (nearby) transmitters may "lock" a receiver and thus prohibit the reception of messages from other senders. Consequently, the "health" of a wireless link between two processes may vary heavily over time [34]. For low-bandwidth wireless transceivers, an acceptable link quality usually even requires communication scheduling [92] (e.g., time-slotted communication) for reducing the mutual interference. Overall, this results in a frequently changing spatial distribution of pairs of nodes that can communicate at a given point in time.

As a consequence, many dynamic networks, in particular, wireless ones [33], are not adequately modelled by means of bidirectional links. Fading and interference phenomenons [60,97], including capture effects and near-far problems, are *local* effects that affect only the receiver of a wireless link: Given that the sender, which is also the receiver of the reverse link, resides at a different location, the two receivers are likely to experience very different levels of fading and interference [58]. This effect is even more pronounced in the case of time-slotted communication, where forward and backward links are used at different times. Consequently, the existence of asymmetric communication links cannot be ruled out in practice: According to [82], 80% of the links in a typical wireless network are sometimes asymmetric.

Despite these facts, most of the dynamic network research we are aware of assumes bidirectional links [69, 72]. The obvious advantage of this abstraction is simplicity of the algorithm design, as strong communication guarantees obviously make this task easier. Moreover, it allows the re-use of existing techniques for wireline networks, which naturally support bidirectional communication. However, there are also major disadvantages of this convenient abstraction:

- First, for dynamic networks that operate in environments with unfavourable communication conditions, like in disaster relief applications or, more generally, in settings with various interferers and obstacles that severely inhibit communication, bidirectional links may simply not be achievable. Implementing distributed services in such settings thus require algorithms that do not need bidirectional links right from the outset.
- Second, the entire system needs to be engineered in such a way that bidirectional single-hop communication can be provided within bounded time. This typically requires relatively dense networks and/or processes that are equipped with powerful communication interfaces, which incur significant cost when compared to sparser networks or/and cheaper or more energy-saving communication devices.
- And last but not least, if directed single-hop communication was already sufficient to reach some desired goal (say, reaching some destination process) via multi-hop messages, waiting for guaranteed single-hop bidirectional communication would incur a potentially significant, unnecessary delay. Obviously, in such settings, algorithmic solutions that do not need bidirectional single-hop communication could be significantly faster.

2.1 Hierarchy and classification

The extensive research in the field of dynamic networks lead to a lot of interesting concepts which have been defined by different researchers in different ways even though they are the same.

This inspired the authors of [32] to introduce a categorization of the different models and concepts used in distributed systems. In the following paragraphs, which were already used in my Masters thesis [100], we will cite large parts of their work to give an overview of the different categories they identified, and later on make a connection to our model.

First, [32] distinguishes 3 main categories of existing networks, which are abstracted by different models in distributed systems.

(a) The study of communication in highly dynamic networks, e.g., broadcasting and routing in delay-tolerant networks. Such networks are characterized by highly dynamic communication links between a fixed set of participants.

(b) The exploitation of passive mobility, e.g., the opportunistic use of transportation networks. Contrary to (a), in such systems, the participants are the dynamic element.

(c) The analysis of complex real-world networks, ranging from neuroscience or biology to transportation systems or social networks, e.g., the characterization of the interaction patterns emerging in a social network. This category is characterized by special emerging properties during the execution of a system.

(a) Delay-tolerant networks

Such networks are highly dynamic and have more or less no infrastructure. Their key characteristic is the absence of guaranteed communications routes between two participants at any time instant. Examples are satellites, pedestrian and vehicular networks. These networks are often called disruption-tolerant, challenged, or opportunistic networks. Even though connectivity assumptions do not hold in general, different mechanisms are available, for example, to broadcast information. Indeed a lot of research has been invested to find new techniques to solve more advanced problems in such networks. This includes pro-active knowledge on the network schedule, delay-based optimization, encounter-based choices, or even analytical and probabilistic strategies. In all those models time is of crucial importance, thus most common graph concepts were extend by a temporal vision.

(b) Opportunistic-mobility networks

Such networks exploit the delay-tolerant network created by mobile carriers equipped with short-ranged-radio transmitters. These carriers are used for performing tasks possibly external and extraneous to the carriers. Entities like code or information can move on the carrier network, using the mobility of the carriers (sometimes called ferries). Examples are, as mentioned in [32], Cabernet, deployed in taxis in the Boston area, which delivers messages and files to users in cars, or UMas DieselNet deployed to 40 buses via WIFI nodes in Amherst. In this context, ferries following deterministic periodic trajectories are of utmost interest. This includes public transport, low earth orbiting satellites or security guard tours. Even though routing is the central aspect here, some research has been done on algorithmic network exploration, i.e, for creating a broadcast infrastructure. Again the time parameter plays an important role in the concepts and the appertaining solutions.

(c) Real world complex networks

As stated in [32], the main problem is to define and formulate mathematical models that properly abstract properties of real dynamic networks. A fundamental idea is to endow the edges with some kind of temporal information. Thus, graph properties can be studied when nodes and edges have temporal constraints. There are many different research papers listed in [32], which used such a concept and established results in different areas of computer science. The authors of [32] conclude that their investigations indicate that temporal concerns are an integral part of recent research efforts in complex system and that the emerging concepts are near identical to those in the field of communication networks.

Time-Varying Graphs

We will use the framework of [32] to formalize a basic, abstract model for real networks. It is general enough to cover most existing models, which emerged independently in various areas of computer science. The **time-varying graph** $TVG = (V, E, T, \rho, \zeta)$ is the core element of this framework and is defined by the following parameters:

- V is the set of all available nodes in the system
- E is the set of all available unidirectional edges in the system
- $T = \mathbb{R}$, which adds a time dimension to the system
- ρ is an indicator function defined on the cross product of E and T
- ζ is a measurement of the delay that different edges may induce

The basic graph, which is also called **underlying graph**, is defined by $G = (V, E)$. G can be seen as a footprint of the TVG. G ignores the time dimension, thus T, ρ and ζ are not relevant for G. Later we will provide a more precise and formal definition. Important to notice is that neither G nor TVG has to be connected; moreover, even if G is connected, it does not imply any connectivity properties of TVG.

Naturally one can define a subgraph TVG', which can be a subset of any part of the original TVG, be it a graph containing only a node subset V' or even the set of all edges labeled by a time subset T', for example $T' = \{1, 5, 7\}$.

Point of View

Until now we introduced a general model that can abstract a broad class of network problems. Depending on the problem it can be important to analyze the evolution from different angles. [32] defines 3 major points of view;

- view of a given relation (edges)
- view of an entity (nodes)
- the view of the global system (entire graph)

If one starts from an edge standpoint, the evolution is represented by the availability and latency of each edge between two different entities. The availability is defined by $I(e)$, which is the union of all dates at which the edge is available, more precisely $I(e) = \{t \in T : \rho(e, t) = 1\}$.
The node point of view can be seen as a changing neighborhood per node. Each node thus identifies the network by neighborhood relations or maybe even transitive neighborhood relations over time.

The graph-centric evolution or the so called characteristic dates of TVG ([32]) is a sequence of snapshots of the dynamic network. It primarily corresponds to the appearance or disappearance of edges in the system. Thus the evolution of TVG can be described by a sequence S_{TVG} of graphs $G_i = (V, E_i)$ where E_i is the set of edges with $\rho(E, t_i) = 1$. Note that one can also give a formal definition of G based on G_i namely, $G = \bigcup G_i$.

Journey and Distance

A sequence J of couples $\{(e_1, t_1), (e_2, t_2)(e_3, t_3)...(e_k, t_k)\}$ such that the edges $e_1, e_2, e_3....e_k$ form a path in G is called a journey in TVG if $\rho(e_i, t_i) = 1$ and $t_{i+1} \geqslant t_i + \zeta(e_i, t_i)$ for all $i \leqslant k$. Let us denote the starting and arrival date of J by $departure(J)$ and $arrival(J)$. The temporal length can by defined by $arrival(J) - departure(J)$. J^*_{TVG} is the set of all journeys in TVG and $J^*_{u,v}$ are all journeys starting in u and ending in v. If at least one journey in $J^*_{u,v}$ exists then we say that u can reach v, which is represented by the notation $u \rightsquigarrow v$.
The distance between nodes can be measured in two ways. Either by

- the already established temporal distance $arrival(J) - departure(J)$
- or the topological distance, which simply counts the hops from u to v in G

TVG **classes**

In this section, which is copied from [32] for the purpose of having a very precise definition of the different classes, we discuss the impact of properties of TVGs on the feasibility and complexity of distributed computing problems, reviewing and unifying existing work from the literature. In particular, we identify a hierarchy of classes of TVGs based on temporal properties that are formalized using the concepts presented in the previous section. These class-defining properties, organized in an ascending partial order of assumptions (see Figure 2.1), from more specific to more general, are important in that they imply necessary conditions and impossibility results for distributed computations. Let us start with the simplest Class 1, i.e., the one with the weakest assumption on the TVG.

Class 1 $\exists u \in V : \forall v \in V, u \rightsquigarrow v$
That is, at least one node can reach all the others. This condition is necessary, for example, for broadcast to be feasible from at least one node.

Class 2 $\exists u \in V : \forall v \in V, v \rightsquigarrow u$
That is, at least one node can be reached by all the others. This condition is necessary to be able to compute a function whose input is spread over all the nodes, with at least one node capable of generating the output. Any algorithm for which a terminal state may depend on all the nodes initial states also falls in this category, such as leader election or counting the number of nodes.

Class 3 (Connectivity over time): $\forall u, v \in V, u \rightsquigarrow v$
That is, every node can reach all the others; in other words, the TVG is connected in T. By the same discussions as for Class 1 and Class 2, this condition is necessary to enable broadcast from any node, to compute a function whose output is known by all the nodes, or to ensure that every node has a chance to be elected.

These three basic classes were used e.g. in [29] to investigate how relations between TVG properties and feasibility of algorithms could be formally established, based on a combination of evolving graphs [54] and graph relabellings [75]. Variants of these classes can be found in recent literature, e.g. in [61], where the assumption that connectivity over time eventually takes place among a stable subset of the nodes is used to implement failure detectors in dynamic networks.

Class 4 (Round connectivity): $\forall u, v \in V, \exists J_1 \in J^*_{(u,v)}, \exists J_2 \in J^*_{(v,u)} : arrival(J_1) \leqslant$

$departure(J_2)$
That is, every node can reach all the others and be reached back afterwards. Such a condition may be required e.g. for adding explicit termination to broadcast, election, or counting algorithms.

The classes defined so far are in general relevant in the case that the lifetime is finite and a limited number of topologically changes are considered. When the lifetime is infinite, connectivity over time is generally assumed on a regular basis, and more elaborate assumptions can be considered.

Class 5 (Recurrent connectivity): $\forall u, v \in V, \forall t \in T, \exists J \in J^*_{(u,v)} : departure(J) > t$
That is, at any point t in time, the temporal subgraph $TVG_{[t,+\infty)}$ remains connected over time. This class is implicitly considered in most works on *delay-tolerant networks*. It indeed represents those DTNs where routing can always be achieved over time. This class was referred to as eventually connected networks by Awerbuch and Even in [13], although the terminology was also used with different meaning in the recent literature (which we mention in another definition below). As discussed in Section 4.1, the fact that the underlying graph $G = (V, E)$ is connected does not imply that the TVG is connected in T, but how the snapshots G_i and thus S_{TVG} is formed matters.

Such a condition on the connectivity of the TVG is, however, necessary to allow connectivity during T and thus to perform any type of global computation. Therefore, the following three classes explicitly assume that the underlying graph G is connected.

Class 6 (Recurrence of edges): $\forall e \in E, \forall t \in T, \exists t' > t : \rho(e, t') = 1$ **and** G **is connected**
That is, if an edge appears once, it appears infinitely often. Since the underlying graph G is connected, Class 6 is a subclass of Class 5. Indeed, if all the edges of a connected graph appear infinitely often, then there must exist, by transitivity, a journey between any pairs of nodes infinitely often. In a context where connectivity is recurrently achieved, it becomes interesting to look at problems where more specific properties of the journeys are involved, e.g. the possibility to broadcast a piece of information in a shortest, foremost, or fastest manner . Interestingly, these three declinations of the same problem have different requirements in terms of TVG properties. It is, for example, possible to broadcast in a foremost fashion in Class 6, whereas shortest and fastest broadcasts are not possible (for a explanation of the different broadcast settings see [31]). Shortest broadcast becomes, however, possible if the recurrence of edges is bounded in time, and the bound is known to the nodes, a property characterizing the next class:

Class 7 (Time-bounded recurrence of edges): $\forall e \in E, \forall t \in T, \exists t' \in [t, t + \Delta), \rho(e, t') = 1$**, for some** $\Delta \in T$ **and** G **is connected**
Some implications of this class include a temporal diameter that is bounded by $\Delta Diam(G)$, as well as the possibility for the nodes to wait a period of Δ to discover all their neighbors (if Δ is known). The feasibility of shortest broadcast follows naturally by using a Δ-rounded breadth-first strategy that minimizes the topological length of journeys.
A particular important type of bounded recurrence is the periodic case:

Class 8 (Periodicity of edges): $\forall e \in E, \forall t \in T, \forall k \in \mathbb{N}, \rho(e, t) = \rho(e, t + k_p)$**, for some** $p \in \mathbb{T}$ **and** G **is connected**
The periodicity assumption holds in practice in many cases, including networks whose entities are mobile with periodic movements (satellites, guards tour, subways, or buses). The periodic assumption within a delay-tolerant network has been considered, among others, in the contexts of network exploration and routing (see [32] for additional references). Periodicity enables also the

construction of foremost broadcast trees that can be re-used (modulo p in time) for subsequent broadcasts (whereas the more general classes of recurrence requires the use of a different tree for every foremost broadcast). More generally, the point in exploiting TVG properties is to rely on invariants that are generated by the dynamics (e.g. recurrent existence of journeys, periodic optimality of a broadcast tree, etc.). In some works, particular assumptions on the network dynamics are made to obtain invariants of a more classical nature. Below are some examples of classes, formulated using the graph-centric point of view of (discrete-time) evolving graphs, i.e., where $TVG = (G, S_{TVG}, \mathbb{N})$.

Class 9 (Constant connectivity): $\forall G_i \in S_{TVG}, G_i$ **is connected**
Here, the dynamics of the network is not constrained as long as it remains connected in every time step. Such a class was used, for example, in [83] to enable progression hypotheses on the broadcast problem. This class was also considered in [70] for the problem of consensus.
Indeed, if the network is always connected, then at every time step there must exist an edge between an informed node and a non-informed node, which allows to bound broadcast time by $n = |V|$ time steps (worst case scenario).

Class 10 (T-interval connectivity): $\forall i \in \mathbb{N}, \exists T \in \mathbb{N}, \exists G' \subseteq G : V_{G'} = V_G, G'$ **is connected, and** $\forall j \in [i, i + T - 1), G' \subseteq G_j$
This class is a particular case of constant connectivity in which the same spanning connected subgraph of the underlying graph G is available for any period of T consecutive time steps. It was introduced in [68] to study problems such as counting, token dissemination, and computation of functions whose input is spread over all the nodes (considering an adversarial edge schedule). The authors show that computation could be sped up by a factor of T compared to the 1-interval connected graphs, that is, graphs of Class 9.
Other classes of TVGs can be found in [93] , based on intermediate properties between constant connectivity and connectivity over time. They include Class 11 and Class 12 below.

Class 11 (Eventual instant-connectivity): $\forall i \in \mathbb{N}, \exists j \in \mathbb{N} : j \geqslant i, G_j$ **is connected. In other words, there is always a time from which on the network is connected**
This class was simply referred to as eventual connectivity in [62], but since the meaning is different than that of Awerbuch and Even (connectivity over time), we renamed it to avoid ambiguities.

Class 12 (Eventual instant-routability): $\forall u, v \in V, \forall i \in N, \exists j \in N : j \geqslant i$ **and a path from** u **to** v **exists in** G_j
That is, for any two nodes, there is always a future time step in which a path exists between them. The difference to Class 11 is that these paths may occur at different times for different pairs of nodes. Classes 11 and 12 were used in [93] to represent networks where routing protocols for (connected) mobile ad-hoc networks eventually succeed if they tolerate transient faults. Most of the works listed above strove to characterize the impact of various temporal properties on problems or algorithms. A reverse approach was considered by Angluin et al. in the field of population protocols [8, 9], where for a given assumption (that any pair of node interacts infinitely often), they characterized all the problems that could be solved in this context. The corresponding class is generally referred to as that of (complete) graph of interaction.

Class 13 (Complete graph of interaction): The underlying graph $G = (V, E)$ **is complete, and** $\forall e \in E, \forall t \in T, \exists t' > t : \rho(e, t') = 1$
From a time-varying graph perspective, this class is the specific subset of Class 6, in which the underlying graph G is complete. Various types of schedulers and assumptions have been

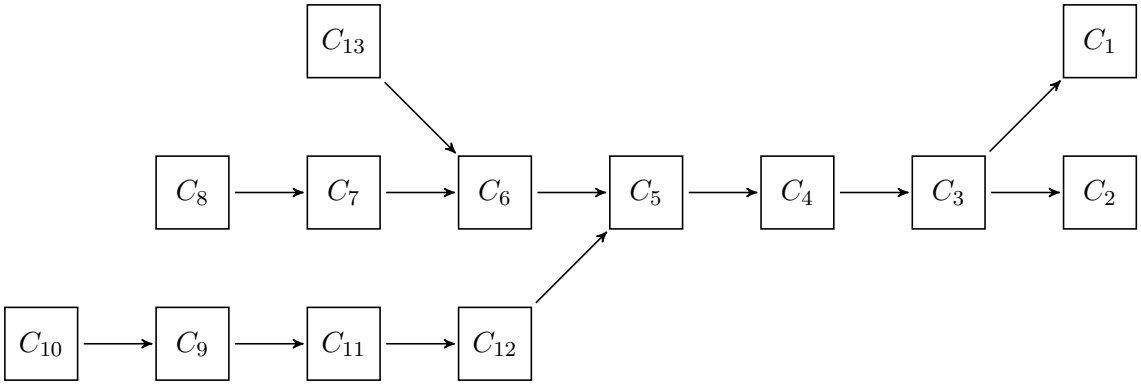


Figure 2.1: Relations of inclusion between classes (from specific to general). Arrows represent "stronger as": e.g., C_{10} is stronger as C_9, i.e., C_{10} includes C_9

subsequently considered in the field of population protocols, adding further constraints to Class 13 (e.g. weak fairness, strong fairness, bounded, or k-bounded schedulers) as well as interaction graphs which might not be complete.

An interesting aspect of unifying these properties within the same formalism is the possibility to see how they relate to one another, and to compare the associated solutions or algorithms. An overview can be gained by looking at the classification shown in Figure 2.1, where basic relations of inclusion between the above classes are reported. All inclusions represented by arrows are strict: for each relation, the parent class (start point of an edge) contains some time-varying graphs that are not in the child class (end point of an edge). Clearly, one should try to solve a problem in the most general context possible. The right-most classes are so general that they offer little properties to be exploited by an algorithm, but some intermediate classes, such as Class 5, appear quite central in the hierarchy. This class indeed contains all the classes where significant work was done. A problem solved in this class would therefore apply to virtually all the contexts considered heretofore in the literature.

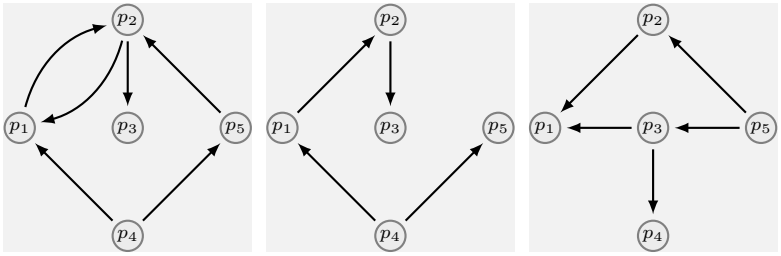
Such a classification may also be used to categorize problems themselves. As mentioned above, shortest broadcast is not generally achievable in Class 6, whereas foremost broadcast is. Similarly, it was shown in [30] that fastest broadcast is not feasible in Class 7, whereas shortest broadcast can be achieved with some knowledge. Since $Class7 \subset Class6$,

$foremostBcast \preceq shortestBcast \preceq fastestBcast$,

defines a partial order on these problems according to topological requirements, where A $\preceq$ B means that A is a less demanding problem than B.

2.2 Model

We consider a set $\Pi = [n] = \{1, \ldots, n\}$ of $n \geqslant 2$ processes with unique identifiers. We use $p_i \in \Pi$ to actually denote the process with UIDs. For every $p_i \in \Pi$, $\mathtt{s}_i$ denotes its local state, taken from a potentially infinite state space, which includes an output variable y_i and an input variable x_i. x_i holds some fixed initial v_i at the beginning of an execution. For some problems x_i is initially assigned to y_i. In these cases x_i is omitted from the problem statement and the algorithm for simplicity and we assume that y_i in round 0 equals x_i. We assume a distributed, round-based computational model in the spirit of the Heard-Of model [40]. Computation proceeds



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

Figure 2.2

in *communicate-closed rounds*: In every round, each process sends its state to its outgoing neighbors, receives messages from its incoming neighbors, and finally updates its state according to a deterministic local algorithm, i.e., a transition function that maps the collection of incoming messages to a new state. Rounds are communication closed in the sense that no process receives messages in round r that are sent in a round different from r. Note that processes do not know, without receiving explicit feedback in later rounds, which processes received their round r broadcast. $\mathtt{s}_i^r$, $r \geqslant 1$, denotes the state of p_i at the end of round r, $\mathtt{s}_i^0$ denotes the initial state.

Failure assumption and adversaries

As our failure assumption we only consider message loss. Communication in a round is modeled by a *directed graph* $\mathcal{G} = \langle V, E \rangle$ with a node for each process. Since a process can obviously communicate with itself instantaneously, every communication graph implicitly contains a self-loop at each node. In the following, we use the *product* of two communication graphs G and H, denoted $G \circ H$, which is the directed graph with an edge from p_i to p_j if there exists p_k such that (p_i, p_k) and (p_k, p_j) are two edges in G and H, respectively. To fully model dynamic networks, in which the topology may change continually and unpredictably, the communication graph at each round is chosen arbitrarily by the so called *adversary*. The communication graph at round r is denoted by G^r, and $\mathcal{N}_j^t = \mathrm{In}_i(r) = \mathrm{In}_i(G^r)$ and $\mathrm{Out}_i(r) = \mathrm{Out}_i(G^r)$ are the sets of incoming and outgoing neighbors (in-neighbors and out-neighbors for short) of process p_i in G^r. Note that we will sloppily write $(p_i \to p_j) \in \mathcal{G}^r$ to denote $(p_i \to p_j) \in E^r$, as well as $p_i \in \mathcal{G}^r$ to denote $p_i \in V = \Pi$.

Figure 2.2 shows a sequence of communication graphs for a network of 5 processes, for rounds 1 to 3. Since every $\mathcal{G}^r$ can range arbitrarily from n isolated nodes to a fully connected graph, there is no hope to solve any non-trivial agreement problem without restricting the power of the adversary to drop messages[1] to some extent. This is further complicated by the fact that processes do not necessarily know n.

Let us fix an algorithm $\mathcal{A}$; a *configuration* is a collection of n process states, one per process. We assume that all processes have the same sets of initial states. Since processes are deterministic, given some configuration C and some communication graph G, the algorithm $\mathcal{A}$ uniquely determines a new configuration which we simply denote $G.C$ if no confusion can arise. Then the *execution of* $\mathcal{A}$ from the initial configuration C^0 and with the communication pattern $\left(G^r\right)_{r \geqslant 1}$

[1]Even though the adversary can only affect communication in our model, it is also possible to model classic send and/or receive omission process failures [87] (and thereby also crash failures): A process that is send/receive omission faulty in round r has no outgoing/incoming edges to/from some other processes in $\mathcal{G}^r$.

is the sequence $C^0, G^1, \ldots, C^{r-1}, G^r, C^r, \ldots$ of alternating configurations and communication graphs such that for each round r, $C^r = G^r.C^{r-1}$.

Formally an adversary is an entity that generates a set of communication graph sequences based on a network model. This set of infinite sequences is called *communication patterns/sequences* in the network model. A particular sequence is *admissible* for an adversary if the sequence is in the communication patterns. If we combine the communication patterns with an initial configuration we get a set of executions denoted $\mathcal{E}_{\mathcal{A}}^{\mathcal{N}}$. Finally, any configuration that occurs in some execution with a communication pattern in some network model is said to be *reachable from* C^0 *by* $\mathcal{A}$ *in* the network model.

A *message adversary* is the combination of an adversary with a given network model.

Definition 1. *Oblivious message adversary: For the oblivious message adversary the network model* $\mathcal{N}$ *is defined by a non-empty set of communication graphs. The message adversary may pick any on those graphs for each round.*

The set of executions with communication patterns in $\mathcal{N}$ with the distance $\text{dist}(E, E') = 1/2^\theta$, where θ is the first index at which E and E' differ, is a compact metric space (e.g., see [76]).

Definition 2. *Non-Oblivious message adversary: For the non-oblivious message adversary, the network model* $\mathcal{M}$ *is defined by a (possible infinitely large) subset of all possible graph sequences. The subset has to have finite description, as the algorithms usually know the message adversary a priori.*

Obviously a network model $\mathcal{N}$ for an oblivious MA can also be defined as an non-oblivious MA. This does not hold for the other direction however, as non-oblivious message adversaries allow e.g. eventual properties like the one that some graph G must eventually occur,which can not be defined as a simple set of graphs.

For convenience, we will sometimes just say that the communication sequences are generated by message adversary $mesAdv$, meaning that the adversary generates graph sequences based on model $\mathcal{N}/\mathcal{M}$ and $mesAdv$ is a simple label for the particular combination. In most cases, this label will contain parameters related to the underlying network model. Furthermore, instead of saying that some sequence is part of the communication patterns generated by the adversary based on some network model, we will simply say the sequence is admissible under message adversary $mesAdv$.

Complementing the traditional approach of partially ordering system models or unreliable failure detectors [35] via their problem solving power (task implementability), the restricted nature of our message adversaries allows us to employ a much simpler and direct way of relating MAs: For a fixed system Π of n processes, we say that A is stronger than B if and only if $A \supseteq B$, i.e., if A can generate at least the communication graph sequences that can be generated by B. As a consequence, an algorithm that works correctly under message adversary A will also work under $B \subseteq A$. If A contains sequences not in B and B contains sequences not in A, then A and B are incomparable. An example for two incomparable adversaries is the adversary that allows only chains for each $\mathcal{G}^r$ and the adversary that allows only circles for each $\mathcal{G}^r$.

We say that a problem is *impossible* under some adversary if there is no deterministic algorithm that solves the problem for every admissible communication graph sequence. For example, every problem that requires at least some communication among the processes is impossible under the unrestricted message adversary, which may generate all possible graph sequences: The sequence $(\mathcal{G}^r)_{r=1}^{\infty}$ where no $\mathcal{G}^r$ contains even a single edge is also admissible here.

Note that every message adversary presented in this thesis introduces a DDN that falls in between class 1 and 3 in Figure 2.1, as we never demand an all-to-all communication during an execution but have some additional restrictions regarding the one-to-all communication property.

Finally, we will establish what it means for a process p_i to *influence* some process p_j, which is central to later parts of this thesis. Note carefully that such an influence is always paired with time: In the spirit of [15, 72], for a given sequence $(\mathcal{G}^r)_{r>0}$ of communication graphs, we say that process p_i at the end of round r influences p_j in round s, denoted as $\mathtt{s}_i^r \rightsquigarrow \mathtt{s}_j^s$, if the state of process p_i at the end of round r could have affected the state of process p_j at the end of round s. Clearly, in our system model, this requires process p_i to send a message in round $r+1$ or later that (directly or indirectly, via some message chain) reaches p_j at the latest in round s, so that it could affect its state $\mathtt{s}_j^s$ reached at the end of round s.

Formally, this is defined via the influence relation given in Definition 3.

Definition 3 (Influence relation)**.** *For a given run with sequence of communication graphs* $(\mathcal{G}^r)_{r>0}$*, the* influence relation *is the smallest relation that satisfies the following conditions for processes* $p_i, p_j, p_k \in \Pi$ *and rounds* $r, r', r'' > 0$*:*

LOCALITY: $\mathtt{s}_i^r \rightsquigarrow \mathtt{s}_i^{r+1}$

NEIGHBOURHOOD: $(p_i \to p_j) \in \mathcal{G}^{r+1} \Rightarrow \mathtt{s}_i^r \rightsquigarrow \mathtt{s}_j^{r+1}$

TRANSITIVITY: $\mathtt{s}_i^r \rightsquigarrow \mathtt{s}_j^{r'}$ *and* $\mathtt{s}_j^{r'} \rightsquigarrow \mathtt{s}_k^{r''} \Rightarrow \mathtt{s}_i^r \rightsquigarrow \mathtt{s}_k^{r''}$

CHAPTER 3

Agreement

A natural approach to build robust services despite the dynamic nature of dynamic networks is to use distributed agreement to agree system-wide on (fundamental) parameters like schedules, frequencies, operating modes etc. In particular, processes that actually need to communicate directly with each other in some application shall agree on those parameters. In order to guarantee this property, agreement algorithms are needed that are always safe. In this chapter we list specific agreement problems for which we will present solutions in different models and define the *v*alency of a configuration which is used to argue about the impossibility of solving certain problems in certain models. The chapter is based on the different problems stated in [22, 57, 102, 108]. Part of this section has been taken from the proposal of the FWF project ADynNet [98], which supports this thesis.

In larger-scale dynamic networks, implementing agreement is challenging (and sometimes impossible), for several reasons:

(a) Solving deterministic (always safe) agreement requires communication graphs that are (eventually) well-connected system-wide. Network partitioning into multiple (partially connected) components cannot be ruled out in dynamic networks, however.

(b) Processes in dynamic networks typically know their "communication-active" neighborhood only. Consequently, they cannot be assumed to have a priori global information, like the number of processes in the system. It is usually even impossible to acquire complete and accurate local knowledge of the entire system at run-time, due to link/node unavailability, network partitioning, insufficient local memory, etc.

(c) Termination times can be large and cannot necessarily be bounded a priori, which makes it difficult for applications to (repeatedly) use consensus for (repeated) decision making: At some given time, different processes could rely on decisions from very different instances of repeated consensus.

Any instance of the collection of agreement problems considered in this thesis is solved in some network model $\mathcal{N}/\mathcal{M}$ if, in all executions admissible in $\mathcal{N}/\mathcal{M}$, three properties *Agreement, Validity and Termination* hold. The exact definition of the three properties depends on the specific instance of problem. In the rest of this chapter we will formalize them and provide some tools used later on in our analysis.

3.1 Problem definitions

3.1.1 Asymptotic consensus

We assume that the local state of process p_i includes a variable y_i that assumes values in an Euclidean d-space $\mathbb{R}^d$, and we let $y^r_{i,E} \in \mathbb{R}^d$ denote the value of y_i at the end of round r in some execution E and define $y^r_E = \left(y^r_{1,E}, \ldots, y^r_{n,E}\right)$. $y^0_{i,E}$ is initialized to x_i. We write $\operatorname{diam}(A) = \sup_{x,y \in A} \|x - y\|$ for the diameter of a set $A \subseteq \mathbb{R}^d$ and $\Delta(y^r_E) = \operatorname{diam}\{y^r_{1,E}, \ldots, y^r_{n,E}\}$ for the diameter of the set of values in round r of $_E$.

Definition 4 (Asymptotic consensus)**.** *We say an algorithm* solves the asymptotic consensus problem *in a network model $\mathcal{N}$ if the following holds for every execution E with a communication pattern in $\mathcal{N}$:*

- Convergence. *Each sequence $\left(y^r_{i,E}\right)_{r \geqslant 0}$ converges.*
- Agreement. *If $y^r_{i,E}$ and $y^r_{j,E}$ converge, then they have a common limit.*
- Validity. *If $y^r_{i,E}$ converges, then its limit is in the convex hull of the initial values $y^0_{1,E}, \ldots, y^0_{n,E}$.*

Observe that the *consensus function* defined by $y^* : E \in (\mathcal{E}, \operatorname{dist}) \mapsto y^*_E \in (\mathbb{R}^d, \|.\|)$, where y^*_E denotes the common limit of the n sequences $\left(y^r_{i,E}\right)_{r \geqslant 0}$, is a priori not continuous. And indeed, there exist asymptotic consensus algorithms whose consensus functions are not continuous.

3.1.2 Approximate consensus

Alternatively, one may also consider the *approximate consensus* problem, in which convergence is replaced by a decision in a finite number of rounds and where agreement should be achieved with an arbitrarily small error tolerance (see, e.g., [77]). Formally, the local state of p_i is augmented with a variable d^i initialized to $\perp$. Process p_i is allowed to set d^i to some value $v \neq \perp$ only once, in which case we say that p_i *decides* v. In addition to the initial values y^0_i, processes initially receive the error tolerance ε and an upper bound Δ on the maximum distance of initial values.

Definition 5 (Approximate consensus)**.** *An algorithm* solves approximate consensus *in $\mathcal{N}$, if for a given $\varepsilon > 0$ and Δ, each execution E with a communication pattern in $\mathcal{N}$ with initial diameter at most Δ satisfies:*

- Termination. *Each process eventually decides.*
- ε-Agreement. *If processes p_i and p_j decide v and v', then $\|v - v'\| \leqslant \varepsilon$.*
- Validity. *If process p_i decides v, then v is in the convex hull of the initial values $y^0_{1,E}, \ldots, y^0_{n,E}$.*

Asymptotic and approximate consensus are clearly closely related.

3.1.3 Terminating exact consensus

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

Definition 6 (Consensus)**.** *An algorithm* solves exact consensus in network model $\mathcal{N}/\mathcal{M}$ *if in all its executions with communication graphs in* $\mathcal{N}$*, the following properties hold in every execution* E*:*

- Termination. *Each process eventually decides.*
- Agreement. *If processes* p_i *and* p_j *decide* v *and* v'*, then* $v = v'$*.*
- Validity. *If process* p_i *decides* v*, then* v *is among the initial values* $x_1, \ldots, x_n$*.*

3.1.4 k-set agreement

For the *k-set agreement problem* [42], which is a relaxation of exact consensus, we assume that both $|\mathcal{V}| > k$ and $n > k$ to rule out trivial solutions:

Definition 7 (k-set agreement)**.** *Algorithm* $\mathcal{A}$ *solves* k-set agreement*, if the following properties hold in every execution of* $\mathcal{A}$*:*

(Termination) Every process must eventually decide.

(k-Agreement) At most k different decision values are obtained system-wide in any execution.

(Validity) If process p_i *decides on* v*, then* v *is some* p_j*'s initial value* x_j*.*

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

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

3.2 Valency

In this section, we define, based on [57], the notion of valency for a consensus algorithm. The basic idea is that the valency of some configuration C is defined by the set of reachable limits of y_i from every admissible sequence going through C. We fix a consensus algorithm $\mathcal{A}$ that solves d-dimensional consensus in a certain network model $\mathcal{N}$ with $n \geqslant 2$ processes. Let C be a configuration reachable by $\mathcal{A}$ in $\mathcal{N}$. Then we define the *valency* of C by

$$Y^*_{\mathcal{N}}(C) = \{y^*_E \in \mathbb{R}^d \ : \ C \text{ occurs in } E \in \mathcal{E}^{\mathcal{N}}_{\mathcal{A}}\} \ .$$

If the algorithm $\mathcal{A}$ is clear from the context, we drop the index.

Let $\delta_{\mathcal{N}}(C) = \operatorname{diam}(Y^*_{\mathcal{N}}(C))$ be the diameter of the set of reachable limits starting from configuration C. It is $\delta_{\mathcal{N}}(C^r) \to 0$ in any execution $E = G^0, C^1, G^1, C^2, \dots$ by Convergence and Agreement. To study the speed of convergence, we introduce the *contraction rate* of algorithm $\mathcal{A}$ in network model $\mathcal{N}$ as

$$\sup_{E \in \mathcal{E}^{\mathcal{N}}_{\mathcal{A}}} \limsup_{r \to \infty} \sqrt[r]{\delta_{\mathcal{N}}(C^r)} \tag{3.1}$$

where $E = C^0, G^1, C^1, G^2, \dots$. In particular, any algorithm that guarantees $\delta_{\mathcal{N}}(C^r) \leqslant \alpha^t \delta_{\mathcal{N}}(C^0)$ for all $r \geqslant 0$ has a contraction rate of at most α.

We obtain the following properties for subsets of network models:

Lemma 8. *Let $\mathcal{N}, \mathcal{N}'$ be two network models with $\mathcal{N}' \subseteq \mathcal{N}$. If $\mathcal{A}$ is an algorithm that solves consensus in $\mathcal{N}$, then (i) it also solves consensus in $\mathcal{N}'$, (ii) for every configuration C reachable by $\mathcal{A}$ in $\mathcal{N}'$, we have $Y^*_{\mathcal{N}'}(C) \subseteq Y^*_{\mathcal{N}}(C)$, (iii) $\delta_{\mathcal{N}'}(C) \leqslant \delta_{\mathcal{N}}(C)$, and (iv) the contraction rate in $\mathcal{N}'$ is less or equal to the contraction rate in $\mathcal{N}$.*

Proof. Statements (i), (ii), and (iii) immediately follow from the definition of valency. It remains to show statement (iv). From $\mathcal{E}^{\mathcal{N}'}_{\mathcal{A}} \subseteq \mathcal{E}^{\mathcal{N}}_{\mathcal{A}}$ and (iii), we deduce

$$\sup_{E \in \mathcal{E}^{\mathcal{N}'}_{\mathcal{A}}} \limsup_{r \to \infty} \sqrt[r]{\delta_{\mathcal{N}'}(C^r)} \leqslant \sup_{E \in \mathcal{E}^{\mathcal{N}}_{\mathcal{A}}} \limsup_{r \to \infty} \sqrt[r]{\delta_{\mathcal{N}}(C^r)} \ ,$$

which concludes the proof. □

We establish two branching properties of valency of configurations in execution trees.

Lemma 9. *Let C be a configuration reachable by algorithm $\mathcal{A}$ in network model $\mathcal{N}$. Then*

$$Y^*_{\mathcal{N}}(C) = \bigcup_{G \in \mathcal{N}} Y^*_{\mathcal{N}}(G.C) \ . \tag{3.2}$$

Proof. First let $y^* \in Y^*_{\mathcal{N}}(C)$. By definition of $Y^*_{\mathcal{N}}(C)$, there exists an execution $E = C^0, G^1, C^1, G^2, \dots$ in $\mathcal{E}^{\mathcal{N}}_{\mathcal{A}}$ and a $t \geqslant 0$ such that $y^* = y^*_E$ and $C = C^r$. Set $G = G^{r+1}$. Hence we have $C^{r+1} = G.C$. But this shows that $y^* \in Y^*_{\mathcal{N}}(G.C)$ since $G.C$ occurs in execution E whose limit is y^*. This shows inclusion of the left-hand side in the right-hand side.

Now let $G \in \mathcal{N}$ and $y^* \in Y^*_{\mathcal{N}}(G.C)$. Then there is an execution $E = C^0, G^1, C^1, G^2, \dots$ in $\mathcal{E}^{\mathcal{N}}_{\mathcal{A}}$ and a $r \geqslant 0$ such that $y^* = y^*_E$ and $G.C = C^r$. Since C is a reachable configuration, there exists an execution $E' = C'^0, G'^1, C'^1, G'^2, \dots$ in $\mathcal{E}^{\mathcal{N}}_{\mathcal{A}}$ and an $s \geqslant 0$ such that $C'^s = C$. Then the sequence

$$E'' = C'^0, G'^1, \dots, C'^s, G, C^r, G^{r+1}, \dots$$

is an execution in $\mathcal{E}^{\mathcal{N}}_{\mathcal{A}}$ with $y^*_{E''} = y^*_E = y^*$. Hence $y^* \in Y^*_{\mathcal{N}}(C)$ because C occurs in E''. This shows inclusion of the right-hand side in the left-hand side and concludes the proof. □

Lemma 10. *Let C be a configuration reachable by algorithm $\mathcal{A}$ in network model $\mathcal{N}$. Then there exist $G, H \in \mathcal{N}$ such that*

$$\operatorname{diam}\left(Y^*_{\mathcal{N}}(C)\right) = \operatorname{diam}\left(Y^*_{\mathcal{N}}(G.C) \cup Y^*_{\mathcal{N}}(H.C)\right) \ . \tag{3.3}$$

Proof. Set $Y = Y^*_{\mathcal{N}}(C)$, and $Y_G = Y^*_{\mathcal{N}}(G.C)$ for $G \in \mathcal{N}$. By Lemma 9 it is $Y = \bigcup_{G\in\mathcal{N}} Y_G$, which means that every sequence of pairs of points in Y whose distances converge to $\operatorname{diam}(Y)$ includes an infinite subsequence in some product $Y_G \times Y_H$ because there are only finitely many. Thus $\operatorname{diam}(Y) \leqslant \operatorname{diam}(Y_G \cup Y_H)$. The other inequality follows from $Y_G \cup Y_H \subseteq Y$. □

Two configurations C and C' are called *indistinguishable for process i*, denoted $C \sim_i C'$, if p_i is in the same state in C as in C'.

As an immediate consequence of the above definition Adm our system model, we obtain:

Lemma 11. *Let C and C' be two reachable configurations, and let G and G' be communication graphs from $\mathcal{N}$. If some process p_i has the same in-neighbors in G and G' and if $C \sim_j C'$ for each of p_i's in-neighbors p_j, then $G.C \sim_i G'.C'$.*

A process p_i is said to be *deaf in a communication graph G* if p_i has a unique in-neighbor in G, namely p_i itself. We are now in position to relate valencies of successor configurations.

Lemma 12. *If the process p_i has the same in-neighbors in two communication graphs G and G' in $\mathcal{N}$, and if there exists a communication graph in $\mathcal{N}$ in which i is deaf, then $Y^*_{\mathcal{N}}(G.C) \cap Y^*_{\mathcal{N}}(G'.C) \neq \emptyset$.*

Proof. From Lemma 11, we have $G.C \sim_i G'.C$.

Let D_i be a communication graph in $\mathcal{N}$ in which the process p_i is deaf. Then we consider an execution E in which C occurs at the end of some round $r_0 - 1$, G is the communication graph at round r_0, and from there on all communication graphs are equal to D_i. Analogously, let E' be an execution identical to E except that the communication graph at round r_0 is G' instead of G. By inductive application of Lemma 11, we show that for all $r \geqslant r_0$, we have $C^r \sim_i C'^r$. In particular, we obtain $y^r_{i,E} = y^r_{i,E'}$. Thus $y^*_E = y^*_{E'}$, which shows that $Y^*_{\mathcal{N}}(G.C)$ and $Y^*_{\mathcal{N}}(G'.C)$ intersect. □

From Lemma 12 we determined the valency of any initial configuration when the network model contains certain communication graphs. If every process is deaf in some communication graph of the network model $\mathcal{N}$, then the next lemma shows that the diameter of the valency of any initial configuration is equal to the diameter of the set of its initial values.

Lemma 13. *If for every process p_i, there is a communication graph in $\mathcal{N}$ in which p_i is deaf, then each initial configuration C^0 satisfies $\delta_{\mathcal{N}}(C^0) = \Delta(y^0)$. In particular, there is an initial configuration for which $\delta_{\mathcal{N}}(C^0) > 0$.*

Proof. Since $Y^*_{\mathcal{N}}(C^0)$ is a subset of the convex hull of $\{y^0_1, \ldots, y^0_n\}$ by the Validity property of consensus and since the diameter of the convex hull of $\{y^0_1, \ldots, y^0_n\}$ is equal to $\Delta(y^0)$, we have the inequality $\delta_{\mathcal{N}}(C^0) \leqslant \Delta(y^0)$.

To show the converse inequality, let p_i and p_j be two processes such that $\|y^0_i - y^0_j\| = \Delta(y^0)$. Let E be the execution with initial configuration C^0 and a constant communication graph in which process p_i is deaf. Now consider $C^0_{(i)}$, an initial configuration such that all initial values are set to y^0_i, and the execution $E_{(i)}$ from $C^0_{(i)}$ with the same communication pattern as in E.

By a repeated application of Lemma 11, we see that at each round r, we have $C^r \sim_i C^r_{(i)}$. Hence, $y^*_E = y^*_{E_{(i)}}$.

From the Validity condition, we deduce that $y^*(E_{(i)}) = y_i^0$. It then follows that $y_i^0 \in Y_{\mathcal{N}}^*(C^0)$. By a similar argument, we see $y_j^0 \in Y_{\mathcal{N}}^*(C^0)$. Hence

$$\delta_{\mathcal{N}}(C^0) \geqslant \|y_i^0 - y_j^0\| = \Delta(y^0) \ ,$$

which concludes the proof. □

3.3 Valency for $0/1$-exact consensus

To show unsolvability of exact consensus in certain network models $\mathcal{N}/\mathcal{M}$, it is common to assume that processes start with one dimensional input values 0 or 1. If every reachable decision from some configuration is 0 the configuration is 0-valent. The analogue holds for 1. If a 0 and a 1 decision is reachable from a configuration it is called bi valent. If we are not interested if it is 0- or 1-valent, we simple call the valency uni valent.

Note that the previously defined general valency collapses to this specialized definition for exact consensus if we demand that y_E^* is either 0 or 1.

CHAPTER 4

Related work

In this chapter we build on and extend the related work discussed in [22, 57, 98, 98, 100, 102, 108] part of which will also be used in Kyrill Winklers thesis [107]. Note that [98] is the proposal of the FWF project ADynNet, which has supported this thesis.

Dynamic networks have been studied intensively in research (see the overview by Kuhn and Oshman [71] and the references therein). Early publications on this topic include [3], [14] or [62] which also mention for the first time modeling a dynamic graph as a sequence of static graphs. One aspect of dynamic networks that can be used to categorize research in this area is whether the set of processes is assumed to be fixed or subject for change. Since it would even be difficult to give a meaningful definition for some agreement problems i.e. k-set agreement in dynamic networks with churn [63] and node mobility [104], we will only consider static (but of possibly unknown number) processes in our evaluation of related work.

Besides work on peer-to-peer networks like [73], where the dynamicity of nodes is the primary concern, different approaches for modeling dynamic connectivity have been proposed, both in the networking context and in the context of classic distributed computing.

4.1 General dynamic network models

Two prominent models in the field of dynamic networks are the so-called LOCAL and CONGEST model presented in [86]. The LOCAL model focuses on locality of executions and hence on one important aspect which is the neighborhood of each process. Each process starts out only knowing its neighborhood and only gradually learns of the rest of the system. The model abstracts away other factors like limited message size or limited local computation space and power. It furthermore assume synchronous communication. The CONGEST model places the same restrictions on the communication graphs but also adds constrains on the bandwidth per link. It hence is allows to explore time and message complexity bottlenecks in the system.

In [3], Afek, Gafni and Rosen introduce a model, where the nodes and communication links of a network form a communication graph. Each edge represents an undirected link, which in turn consists of a directed pair of links. Moreover edges have a bounded capacity and the information sent on each link can be arbitrarily delayed. As it allows unbounded transmission delays, this model can accurately abstract a dynamic network with a fixed set of processes. Based on this model, [3] proposes algorithms to solve the end-to-end communication problem for

different criteria. Assuming an upper bound on time a message is being held in the network, they propose the **slide** protocol for solving the end-to-end communication problem. Furthermore they extend the basic slide protocol with a known data dispersal component to optimize the message complexity to O(n). Finally, they combine an algorithm designed for static network structures (fixed processes and fixed communication channels) with the slide protocol to obtain a protocol, that unlike most algorithms for dynamic networks, adequately improves to performance when being run in a static network.

Kuhn, Moses and Oshman in [70] and Kuhn, Lynch and Oshman in [68] use network models where communication graphs are organized in lock-step rounds. Furthermore both models demand that each per round communication graph is connected. In the later case the connected subgraph has to be the same for an interval of T rounds. As consensus is trivial in both scenarios the authors focus on solving more complex agreement problems. In particular, they study, coordinated consensus where the first and the last process must decide within a fixed number of rounds apart. They also study problems like token dissemination and its performance. Especially interesting for this thesis is a property called *eventual connectedness*. It means that two nodes are connected at some point during the run (not necessarily direct, but via other nodes). This assumption is also used in different ways in most recent publications about dynamic networks; we will also use it in this thesis to guarantee termination of our algorithms.

The authors of [14] use a similar model. Again, the set of processes is fixed and the links can dynamically crash or recover. This is modeled via a finite but unbound travel time per link in conjunction with a synchronous model. Moreover, each node knows the state of all directly connected links. The focus of this paper is the exploration of possible combinations of static approaches with dynamic algorithms. Different approaches are assessed with respect to several complexity measures like communication, space and time complexity. The main result is a poly-log time, space and communication simulation of synchronous, static protocols on dynamic asynchronous networks. To accomplish this, the authors combine the concept of sparse covers to obtain an improved synchronizer protocol for static networks and the concept of local resets to implement a local rollback technique. The result is a protocol that can simulate a static system on a dynamic network.

The main points to gain from these papers, besides the already mentioned results, are that static solutions can be seen as special cases of dynamic ones. Dynamic algorithms often lack runtime, space or message complexity improvements when employed in networks with near-static behavior, however. Moreover, one can conclude that assuming synchrony in a dynamic network is not far fetched, as it can be achieved by some synchronizer algorithm in an asynchronous network.

4.2 Round-by-round models

Rather than considering the possibility of unbounded/infinite delay on the transmission link, Santoro and Widmayer in [96] focus on explicit communication failures and the resulting consensus possibility/impossibility borders. The basic model is a synchronous system with dynamic ("moving") transmission failures. The message sending process is realized via a broadcast from one process to all other processes in the system via point to point communication over a fully connected network. The authors distinguish 3 types of failures: omission (message is never delivered), corruption (message is delivered with different content), and addition (a message is

delivered although no message was sent). The goal is to achieve agreement among a majority of the processes (i.e., the system consists of n processes with initial value $\in \{0, 1\}$ and at least $k > \frac{n}{2}$ processes shall decide on the same value). They establish conditions on the number of transmission failures that render this problem solvable/unsolvable. As we are using a similar model with only omission failures, the result showing that consensus is impossible under the assumption of $n - 1$ dynamic omission transmission failures is very important.

Gafnis work on "round-by-round fault detectors" in [59] has some similarity to the "heard-of" model [40], which we will discuss subsequently. The main difference is that it focuses on process failures instead of transmission failures. An oracle (a round-by-round fault detector) is used by each process to obtain a set of processes from which it will not be able to receive any data in the current round. Thus, the oracle abstracts away the actual reason for not receiving messages. Moreover, the communication means are also abstracted away, in the sense that it does not matter whether message passing or shared memory is used. Obviously, abstracting away the failure source locally at the receiver via a set of faulty processes produced by the oracle has the benefit that the processes do not have to worry about the reason for and the actual source of the failure. On the down side, dedicated implementations of the oracle have to be provided for every different network model, which may or may not be possible.

The "heard-of" model (abbreviated HO-model) published in [40] by Charron-Bost and Schiper is a generalization of the model introduced in [19] and hence the model used in this thesis. The processes are fixed and assumed to be fault-free and their execution progresses as a sequence of lock-step synchronous rounds. It uses a slightly different approach with respect to failures than [59], which also abstracts away the reason why messages are missing (sender crash, receiver failure, etc.). The only information describing communication in each round is a set $H(p, r)$, which consists of all processes that have successfully sent a message to p in round r. Rounds are communication-closed, so messages sent in some round r can only be received in the same round r, and are disregarded and hence not included in any $H(p, r)$ if received late. Thus, a run is described by a set $H(p, r)$ for each node, for every round $r \geqslant 1$, which is determined by the adversary. To make different agreement problems solvable in the HO-model, different predicates restricting the freedom of the adversary are usually required [40]. Note that there is also a Byzantine extension of the HO-model [16]. Obviously, the HO-model is a suitable abstraction for wireless communication, because every participant simply broadcasts its messages but only has knowledge about received information. It is also a reasonable way to hide the different sources of failures in the communication model and to handle different communication failures via the HO-sets.

A model that is similar to the HO-model is the perception-based model by Biely, Schmid and Weiss [23]. A sequence of perception matrices, the rows of which are the sets of messages received by some processes, is used to express failures of processes and links. A *hybrid failure model*, which supports different types of process an link failures, is used to restrict the possible perception matrices in a round-by-round fashion. To circumvent the impossibility of transmission failures established in [96], for example, a restriction to the number of transmission failures per process for outgoing and incoming links can be applied. Since these failures are tied to each process on a per round basis, agreement can be achieved in the presence of $O(n^2)$ transmission failures per round. In [23] the authors discuss a suite of consensus algorithms under this model, ranging from the oral messages (EIG) algorithm to the *Phase Queen* and the *Phase King* algorithms. The lower bound results established in [23] make it clear that there is no hope to solve consensus without quite strong connectivity guarantees. In particular, without some eventual stability assumptions, the number of link failures per round has to be bounded for every process.

4.3 Message adversaries

A central element in this thesis, namely, message adversaries, is most prominently featured in Afek and Gafni [2], where they used them for relating problems solvable in wait-free read-write shared memory systems to those solvable in message-passing systems, and in Raynal and Stainer [94] where message adversaries are used to explore the relationship between round-based models and failure detectors. In the latter, simulations between a message adversary and a failure detector are used to prove the same computational power of the two models. Moreover, based on this task equivalence, a hierarchical structure of message adversaries and failure detectors has been developed.

An equivalent concept has already been used in [18] and [19], which study consensus and k-set agreement under certain communication graph sequences: In [18], the goal was to solve k-set agreement for graph sequences containing a stable skeleton graph as a subgraph per round. In [19], the pivotal concept of vertex-stable strongly connected components has been introduced. The goal is to solve consensus under the assumption that at some point during the execution the information of a set of strongly connected processes can reach every process in the systems albeit processes do not know when this will happen. These vertex-stable components form the basis for most of the message adversaries studied in Section 6.

4.4 Asymptotic consensus

The problem of asymptotic consensus in dynamic networks has been extensively studied, see e.g. [7, 24, 27, 43, 80]. The question of guaranteed convergence rates and decision times of the corresponding approximate consensus problems, naturally arise in this context. Algorithms with convergence times exponential in the number of processes have been proposed, e.g., in [27].

Olshevsky and Tsitsiklis [85], proposed an algorithm with polynomial convergence time in bidirectional networks with certain stability assumptions on the occurring communication graphs. The bounds on convergence times were later on refined in [81]. Chazelle [43] proposed an averaging algorithm with polynomial convergence time, which works in any bidirectional connected network model.

To speed up decision times, algorithms where processes set their output based on values also received in previous round rather than just in the current round, have also been considered in literature: Olshevsky [84] proposed a linear convergence time algorithm that uses messages from two rounds, however, being restricted to fixed bidirectional communication graphs. In [110], a linear decision time algorithm for a possibly non-bidirectional fixed topology was proposed. It requires storing all received values. In previous work [38], Charron-Bost et al. proposed the midpoint algorithm, which has constant decision time in non-split network models, and the amortized midpoint algorithm with linear decision time in rooted network models. Non-split means that each round of communication guarantees for every pair of process to receive a message from a common process.

To the best of our knowledge, the only lower bound on convergence rate in dynamic networks has been shown in [28]: the authors proved that the convergence rate of a specific averaging algorithm in a non-split network model with n processes is at least $1 - \frac{1}{n}$.

In the context of classical distributed computing failure scenarios, Dolev et al. [48] studied the related approximate agreement problem: they considered fully-connected synchronous distributed

systems with up to f Byzantine processes, and its asynchronous variant. The two presented algorithms require $n \geqslant 3f + 1$ for the synchronous and $n \geqslant 5f + 1$ for the asynchronous distributed system, the first of which is optimal in terms of resilience [55]. The latter result was later on improved to $n \geqslant 3f + 1$ in [1]. Both papers also address the question of optimal contraction rate in such systems. Since, however, in synchronous systems with $n \geqslant 3f + 1$, exact consensus is solvable, which results in to a contraction rate of 0, the authors consider bounds for round-by-round contraction rates. In [48], they show that the achieved round-by-round contraction rate of $\frac{1}{2}$ is actually tight for a certain class of algorithms that repeatedly set their output to the image of a so-called "cautious function" applied to the multiset of received values. Lower bounds for arbitrary algorithms, however, remain an open problem. In higher dimensions, Mendes et al. [79] proposed algorithms with a convergence time of $d \cdot \lceil \log_2 \frac{\sqrt{d}\Delta}{\varepsilon} \rceil$ under the optimal resiliency condition $n \geqslant f \cdot \max\{3, d + 1\} + 1$.

Fekete [53] also studied round-by-round contraction rates for several failure scenarios, again, all in which exact consensus is solvable. He proved asymptotically tight lower bounds for synchronous distributed systems in presence of crashes, omission, and Byzantine processes. The bounds hold for approximate agreement algorithms that potentially take into account information from all previous rounds.

4.5 Exact consensus

Early work [87] proposed a model with send and receive omission faults where sending or receiving processes may lose messages. The authors provide a fast, e.g. early stopping, and efficient algorithm which solves the "generals problem", a problem closely related to consensus.

In [96], Santoro and Widmayer introduced the moving omission failure model and proved that $n - 1$ lost messages per round already make consensus unsolvable. [99] extended the moving omission failure model by allowing up to n^2 faults per round, albeit for restricted failure patterns: If the number of send omission faults and receive omission faults per process and round remains suitably bounded consensus is still solvable. Both papers also deal with both message omissions and corruption.

Consensus under a model with round-by-round communication graphs has been studied by Biely, Robinsion and Schmid in [19]. The set of processes is assumed to be fixed and communication is modeled by a sequence of round graphs G_r, exactly as in the setting of message adversaries. Instead of a stable skeleton, however, it is assumed that each round graph is at least weakly connected and has at most one root component,i.e. a strongly connected component without incoming edges. Moreover, eventually, the root component needs to be vertex-stable, i.e. consist of the same process, for a certain number of consecutive rounds to guarantee termination. The model can be placed between class 1 and 3 of the framework established in [32] (note that it is strictly weaker than 3 and stronger than 1).

An algorithm for solving consensus in this model has been proposed which can detect vertex-stable root components and determine their stability intervals. The idea is to optimistically "lock" on $D + 1$-stable root components (D is an upper bound on the information propagation in a strongly connected component) and to promote the locked on information among all processes in the weakly connected component. If the vertex-stability of a root component holds for further D rounds, the algorithm can safely decide, because it is guaranteed that the locked on information has reached every process in the associated weakly connected component. Interestingly, for a single process to terminate, a much smaller stable root interval is needed than for guaranteeing

global termination. Actually the global termination time interval is twice as long $(4D+1)$. Thus, it is possible to have some decision in the network without the majority of the processes knowing. We will see that we have a very similar problem for the k-set case, where some processes that already made decisions but are "hidden" in different components could try to promote their own values later, which would violate the agreement part.

Furthermore, [19] shows, via an impossibility result, that in a model with such weak assumptions, consensus is impossible if no root is vertex stable for at least D rounds. Thus, building on D stable intervals, as we are doing in this thesis, seems a reasonable starting point to solve different agreement problems.

Coulouma et al. show in [46] that for oblivious message adversaries the exact consensus problem and the broadcast problem are closely related. Furthermore they provide a formal framework for oblivious message adversaries which allows a precise characterization of the solvability/impossibility border. They show that each set of graphs can be sorted into subsets called β-classes. This sorting is based on indistinguishability of graphs for subsets of processes S where S is a so called source component of a graph in the β-class. Consensus can be solved in a β-class if at least one process can reach every other process in each graph part of the β-class, which then is called broadcastable. Hence, they provide an algorithm which solves exact consensus if all β-classes are broadcastable on the one hand and on the other hand present an impossibility result if at least one β-class is not broadcastable.

4.6 k-set agreement

The k-set agreement problem is a well known problem in distributed computing. Introduced in [42], it has been studied under many different system models and failure assumptions [6]. Most of these assumptions imply static models, typically involving a foreknown maximum number f of faulty processes and bidirectional reliable communication links between processes.

For synchronous systems, where one assumes full connectivity, algorithms for this problem are rather simple and mostly based on some upper bound on the required number of rounds, like $\lfloor \frac{f}{k} + 1 \rfloor$. These upper bounds, which usually depend on the number of faulty processes in the network, ensure sufficient information flow between correct processes.

Since k-set agreement is a relaxation of consensus, it has primarily been used to explore the possibility/impossibility border of computability in asynchronous systems. In particular, [95] or [26] establish impossibility results in the *f-crash resilient asynchronous* model via arguments from algebraic topology. They prove that, for $f \geqslant k$, where f is the maximum number of crashed processes, k-set agreement is impossible.

In [20], partially synchronous systems with weak synchrony requirements were provided which allow a solution for k-set agreement. This is done via the so called generalized $(n-k)$-loneliness failure detector. Furthermore they show that this failure detector is also a weakest failure detector for anonymous systems. In an anonymous system unique identifiers are not available to the process.

A model that comes close to dynamic networks as defined in this thesis is proposed in [91], where general omission failures are assumed. Such failures include process crashes, send omission and receive omission faults. Nevertheless the faults are bound to the processes and not to the links between processes. Moreover, they do not use the standard distinction between correct and faulty processes but use good and bad processes to define if k-set agreement is solvable. A good process neither crashes nor commits receive omission failures.

As already established in the introduction, static assumption are not suitable for modeling k-set agreement in dynamic systems. A first step towards a more dynamic setting has been made in [18], though. It considers k-set agreement in a model similar to the HO-model, where the communication predicate guarantees a stable skeleton graph: Each round is represented by a communication graph G_r, which consists of nodes (processes) and directed edges (p, q) modeling round r communication between process p and g, i.e., when $p \in H(q, r)$. The assumption of a stable skeleton graph guarantees that $\bigcap_{r \geqslant 1} G^r$ is non- empty, i.e., that there is a non-empty set of edges, which are present in every round.

Moreover, [18] assumes that in each possible set of $k + 1$ nodes of the stable skeleton graph, two nodes share the same parent node in every round. This "two-source" assumption guarantees that the stable skeleton consists of at most k root components i.e., strongly connected component with no incoming edges. An algorithm was proposed that can detect root components via approximating the stable skeleton; since the approximation eventually stabilizes, so do the roots. To solve k-set agreement, it is sufficient to guarantee a common decision among all members of a root. This is possible, since all members of a root can exchange information (as the root is strongly connected) and can reach all nodes in the associated weakly connected component. In [17], the authors establish a framework for k-set agreement *easy impossibility proofs*. It can be used to show impossibility results in different models via the same generic theorem, which reduces k-set agreement to consensus in certain executions. Basically, it uses the concept of *restricted algorithms* for relating executions of the whole system and executions of disconnected components. The relation is based on indistinguishable runs for some processes. Essentially, the executions considered assume k disconnected components. Since disconnected components never hear from each other during the whole run, independent decisions can be forced in every component. Obviously, to guarantee the k-agreement property, i.e., no more than k different decisions, this requires to solve consensus within every component. The impossibility of k-set agreement follows from showing that the restricted algorithm cannot solve consensus in at least one component.

Regarding k-set agreement in dynamic networks, we are not aware of any previous work except [103], where bidirectional links are assumed, and [18], where the existence of an underlying *static* skeleton graph (a non-empty common intersection of the communication graphs of all rounds) with at most k *static* source components was assumed. Note that this essentially implies a directed dynamic network with a static core. By contrast, the algorithms presented in this thesis and published in [22] allow the directed communication graphs to be fully dynamic.

4.7 Degrading consensus problems

Graceful degradation means that the algorithm tries to adapt to the current situation of the system. In the case of k-set agreement, for example, one strives for as few different decisions as possible, in the optimal case, just one. We are not aware of related work exploring gracefully degrading consensus. However, there have been several attempts to weaken the semantics of consensus, in order to cope with partitionable systems and excessive faults. Vaidya and Pradhan introduced the notion of *degradable* agreement [105], where processes are allowed to also decide on a (fixed) default value in case of excessive faults. The *almost everywhere agreement* problem introduced by [51] allows a small linear fraction of processes to remain undecided. Aguilera et. al. [4] considered quiescent consensus in partitionable systems, which requires processes outside the majority partition not to terminate. None of these approaches is comparable to gracefully degrading k-set agreement, however: On the one hand, we allow more different

decisions, on the other hand, all correct processes are required to decide and every decision must be the initial value of some process.

Ingram et. al. [65] presented an asynchronous leader election algorithm for dynamic systems, where every component is guaranteed to elect a leader of its own. Whereas this behavior clearly matches our definition of graceful degradation, contrary to decisions, leader assignments are revocable and the algorithm of [65] is guaranteed to successfully elect a leader only once the topology eventually stabilizes.

4.8 Other agreement problems

Agreement problems in dynamic networks with undirected communication graphs have been studied in [12, 45, 72]. [12] provides fast randomized distributed algorithms that guarantee stable almost-everywhere agreement with high probability. In [72] the authors focus on eventual, simultaneous, and δ-coordinated consensus, as well as their relationship to other distributed problems. In [45], an asymptotically optimal algorithm in respect to time complexity which solves the regional consecutive leader election (RCLE) problem is presented.

In terms of the classes of [32], the model of [68] is in one of the strongest classes (Class 10) in which every process is always reachable by every other process. They study problems like token dissemination and show that calculating the size of the system is bound by $O(n^2)$ in their model.

The leader election problem in dynamic networks has been studied in [44,45], where the adversary controls the mobility of nodes in a wireless ad-hoc network. This induces dynamic changes of the (undirected) network graph in every round and requires any leader election algorithm to take $\Omega(Dn)$ rounds in the worst case, where D is a bound on information propagation.

CHAPTER 5

Oblivious message adversaries

The following section is based on [57] and explores oblivious message adversaries that are too strong to allow a solution of exact consensus but weak enough for asymptotic consensus. We will give tight bounds for the contraction rate for some of those message adversaries and furthermore connect Coulouma et al.'s [46] framework used for exploring the impossibility/solvability border for exact consensus in oblivious message adversaries to derive a general bound on contraction rates.

In a previous paper [37], Charron-Bost et al. proved the following characterization of network models in which asymptotic consensus is solvable which is based on the notion of a directed spanning tree. A spanning tree is a cycle free graph where exactly one process has a path to every other process in the graph.

Theorem 14 ([37, Theorem 5])**.** *In any dimension d, the asymptotic consensus problem is solvable in a network model $\mathcal{N}$ if and only if each graph in $\mathcal{N}$ has a rooted spanning tree.*

For the proof of the sufficient condition, Charron-Bost et al. focused on *convex combination algorithms,* where each process p_i updates its variable y_i to a value within the convex hull of values y_j^{r-1} it has just received. For the proof of the necessary condition, it is easy to show that if even one graph is disconnected, by applying this graph forever, consensus is unsolvable.

If every graph in $\mathcal{N}$ has a rooted spanning tree, we call a network model rooted.

Furthermore, they showed in [37] that convex combination algorithms where processes update their y_i via a weighted average of the received values, with weights that only depend on the currently received values, also solve asymptotic consensus in rooted network models. Such algorithms are memoryless, require little computational overhead and, more importantly, have the benefit of working in anonymous networks. Interestingly, their consensus function y^* is continuous.

Theorem 15. *The consensus function of every convex combination algorithm that solves asymptotic consensus is continuous on the set of its executions.*

Proof. Let $(E_s)_{s\geqslant 0}$ be a sequence of executions that converges to E. By definition of the distance $\operatorname{dist}(E, E')$ on the execution space in Section 2, this in particular means that

$$\forall r \geqslant 0\ \exists s^r\ \forall s \geqslant s^r\colon \quad y_s^0 = y^0,\ y_s^1 = y^1,\ \ldots,\ y_s^r = y^r \tag{5.1}$$

where y_s^r and y^r denote $y_{E_s}^r$ and y_E^r, respectively.

Let $\varepsilon > 0$. By the definition of the limit y^* of execution E, there exists some r such that

$$\forall p_i \in [n]\colon \quad \|y_i^r - y^*\| \leqslant \varepsilon/3 \ .$$

By (5.1), there is an s_r such that

$$\forall s \geqslant s^r \ \forall p_i \in [n]\colon \quad \|y_s^i(t) - y^*\| \leqslant \varepsilon/3 \ .$$

By the triangle inequality, this means

$$\forall s \geqslant s_t \ \forall i, j \in [n]\colon \quad \|y_s^i(t) - y_s^j(t)\| \leqslant 2\varepsilon/3 \ .$$

Because the algorithm is a convex combination algorithm, the limit y_s^* lies in the convex hull of the points $y_s^1(t), \ldots, y_s^n(t)$. That is,

$$\forall s \geqslant s_t \ \forall i \in [n]\colon \quad \|y_s^i(t) - y_s^*\| \leqslant 2\varepsilon/3 \ .$$

Combining these inequalities gives

$$\forall s \geqslant s_t\colon \quad \|y_s^* - y^*\| \leqslant \|y_s^* - y_s^i(t)\| + \|y_s^i(t) - y^*\| \leqslant \frac{2\varepsilon}{3} + \frac{\varepsilon}{3} = \varepsilon$$

where i is any process. This proves $\lim_{s\to\infty} y_s^* = y^*$ as required. □

Observe that in the case $\mathcal{A}$ is a convex combination algorithm, the valency of a configuration C is a compact set in $\mathbb{R}^d$ since the consensus function is continuous and the set of executions in which C occurs is a compact set. The execution space is compact and thus the set of executions containing C is closed. It follows that the latter is also compact.

In [46], Coulouma et al. characterized the network models in which exact consensus is solvable. In [37], Charron-Bost et al. showed that asymptotic consensus is solvable in a significantly broader class: it is solvable if and only if a network model is rooted. In the rest of this chapter we aim to shed light on the deeper relation between these two problems by studying valencies and convergence rates. The main results are a characterization of the topological structure of valencies with respect to solvability of exact consensus (Theorem 25) and nontrivial lower bounds on the contraction rates whenever exact consensus is not solvable (Theorem 27 and Corollary 28).

We start with recalling some definitions from Coulouma et al. [46]. In the following, we denote by $\mathcal{R}(G)$ the set of roots of a communication graph G, i.e., the set of processes that have a directed path to all other a directed path to all other processes in G. For a set $S \subseteq [n]$, let $\mathrm{In}_S(G) = \bigcup_{j\in S} \mathrm{In}_j(G)$. The set $\mathrm{Out}_S(G)$ is defined analogously.

Definition 16 ([46, Definition 4.7])**.** *Let $\mathcal{N}$ be a network model. Given $G, H, K \in \mathcal{N}$, we define $G\alpha_{\mathcal{N},K}H$ if $\mathrm{In}_{\mathcal{R}(K)}(G) = \mathrm{In}_{\mathcal{R}(K)}(H)$. The relation $\alpha_{\mathcal{N}}^*$ is the transitive closure of the union of relation $\alpha_{\mathcal{N},K}$ where K varies in $\mathcal{N}$.*

Definition 17 ([46, Definition 4.8])**.** *Let $\mathcal{N}$ be a network model. We define the relation $\beta_{\mathcal{N}}$ to be the coarsest equivalence relation included in $\alpha_{\mathcal{N}}^*$ such that for all G, H holds:*
(Closure Property) *If $G\beta_{\mathcal{N}}H$, then there exists a non negative integer q and communication graphs $H_0, \ldots, H_q \in \mathcal{N}$ and $K_1, \ldots, K_q \in \mathcal{N}$ such that*

- *(i) $G = H_0$ and $H = H_q$*
- *(ii) $\forall r \in [q]\colon \ H_r\beta_{\mathcal{N}}G$ and $K_r\beta_{\mathcal{N}}G$*
- *(iii) $\forall r \in [q]\colon \ H_{r-1}\alpha_{\mathcal{N},K_r}H_r$*

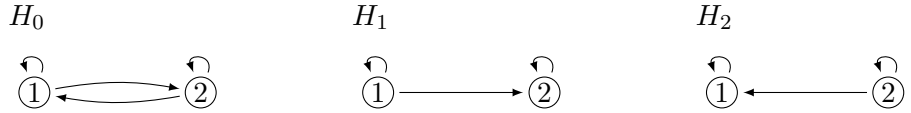


Figure 5.1: The rooted communication graphs H_0, H_1, and H_2 for $n = 2$

5.1 Tight bounds for asymptotic consensus

Lower Bound for $n = 2$

Next, be prove a lower bound of 1/3 on the contraction rate of algorithms that solve asymptotic consensus in the network model of all rooted (and here also non-split) communication graphs with two processes. Combined with Algorithm 1 which achieves this lower bound [38], we have indeed identified a tight bound on the contraction rate for $n = 2$. Moreover, the algorithm also shows that the lower bound is achieved by a simple convex combination algorithm.

Algorithm 1 Algorithm with contraction rate 1/3 for $n = 2$

Require:
1: $y_i \in \mathbb{R}$
Ensure:
2: send y_i to other process and receive y_j if $p_j \in \mathrm{In}_i^t$
3: **if** y_j was received **then**
4: $\quad y_i \leftarrow y_i/3 + 2y_j/3$
5: **end if**

A straightforward analysis of Algorithm 1 shows that its contraction rate is equal to 1/3.

Note that for $n = 2$, there are 3 possible rooted communication graphs that may occur, all of which are non-split; see Figure 5.1: (i) H_0 in which all messages are received, (ii) H_1 in which process 2 receives process 1's message but not vice versa, and (iii) H_2 in which process 1 receives process 2's message but not vice versa.

Theorem 18. *The contraction rate of any asymptotic consensus algorithm for $n = 2$ processes in a network model that includes the three graphs H_0, H_1, and H_2 is greater or equal to* 1/3.

Proof. We show the stronger statement that for every initial configuration C^0 there is an execution $E = C^0, G^1, C^1, G^2, \ldots$ starting from C^0 such that

$$\delta_{\mathcal{N}}(C^r) \geqslant \frac{1}{3^r} \cdot \delta_{\mathcal{N}}(C^0) \tag{5.2}$$

for all $r \geqslant 0$. This, applied to an initial configuration with $\delta_{\mathcal{N}}(C^0) > 0$, which exists by Lemma 13, then shows the theorem.

Note that it suffices to show (5.2) for the specific network model $\mathcal{N}' = \{H_0, H_1, H_2\}$ because $\delta_{\mathcal{N}}(C^r) \geqslant \delta_{\mathcal{N}'}(C^r)$ by Lemma 8 and $\delta_{\mathcal{N}'}(C^0) = \delta_{\mathcal{N}}(C^0)$ by Lemma 13 whenever $\mathcal{N} \supseteq \mathcal{N}'$. We hence suppose $\mathcal{N} = \mathcal{N}'$ in the rest of the proof.

The proof is by inductive construction of an execution $E = C^0, G^1, C^1, G^2, \ldots$ whose configurations C^r satisfy (5.2). Equation (5.2) is trivial for $r = 0$.

Now assume $r \geqslant 0$ and that Equation (5.2) holds for r. There are three possible successor configurations of C^r, one for each of the communication graphs H_0, H_1, and H_2 in $\mathcal{N}'$. Set $C_k^{r+1} = H_k.C^r$. Further let $Y = Y^*_{\mathcal{N}'}(C^r)$, and $Y_k = Y^*_{\mathcal{N}'}(C_k^{r+1})$.

We will show that there is some $\hat{k} \in \{0,1,2\}$ with $\operatorname{diam}(Y_{\hat{k}}) \geqslant \operatorname{diam}(Y)/3$. We then define $G^{r+1} = H_{\hat{k}}$ and $C^{r+1} = C^{r+1}_{\hat{k}}$. By the induction hypothesis, we then have

$$\delta_{\mathcal{N}'}(C^{r+1}) \geqslant \delta_{\mathcal{N}'}(C^r)/3 \geqslant \delta_{\mathcal{N}'}(C^0)/3^{r+1} \ , \tag{5.3}$$

i.e., Equation (5.2) holds for $r+1$.

Assume by contradiction that $\operatorname{diam}(Y_k) < \operatorname{diam}(Y)/3$ for all $k \in \{0,1,2\}$. From Lemma 9 we have $Y = Y_0 \cup Y_1 \cup Y_2$. Noting that process 1 is deaf in H_1 and process 2 has the same incoming edges as in H_0, and that process 2 is deaf in H_2 and process 1 has the same incoming edges as in H_0, we obtain from Lemma 12 that

$$Y_0 \cap Y_1 \neq \emptyset \quad \text{and} \quad Y_0 \cap Y_2 \neq \emptyset \ . \tag{5.4}$$

The sets Y_0 and Y_1 intersecting means

$$\operatorname{diam}(Y_0 \cup Y_1) \leqslant \operatorname{diam}(Y_0) + \operatorname{diam}(Y_1) < \frac{2}{3}\operatorname{diam}(Y) \ . \tag{5.5}$$

Further, the sets $Y_0 \cup Y_1$ and Y_2 intersecting means

$$\operatorname{diam}(Y) = \operatorname{diam}(Y_0 \cup Y_1 \cup Y_2) \leqslant \operatorname{diam}(Y_0 \cup Y_1) + \operatorname{diam}(Y_2) < \operatorname{diam}(Y) \ , \tag{5.6}$$

a contradiction. This concludes the proof. □

Lower Bound for $n \geqslant 3$

In this subsection, we prove a lower bound of 1/2 on the contraction rate of asymptotic consensus algorithms for $n \geqslant 3$ processes, in a network model that includes graphs derived from a communication graph G where processes are made deaf in the derived graphs. As a special case this includes the network model of all non-split communication graphs. Charron-Bost et al. [38] presented the midpoint algorithm (given in Algorithm 2) for dimension $d = 1$, which has a contraction rate 1/2 for non-split communication graphs. Together this shows tightness of our lower bound in dimension one.

Algorithm 2 Midpoint algorithm

Require:
1: $y_i \in \mathbb{R}$

Ensure:
2: send y_i to all processes in Out_i^r and receive y_j from all processes p_j in In_i^r
3: $m_i \leftarrow \min\left\{y^j \mid j \in \mathrm{In}_i^r\right\}$
4: $M_i \leftarrow \max\left\{y_j \mid j \in \mathrm{In}_i^r\right\}$
5: $y_i \leftarrow (m_i + M_i)/2$

One can apply the algorithm component wise in dimension $d = 2$ to show tightness of our lower bound also there. Unfortunately, component wise application in dimension $d \geqslant 3$ does not yield an asymptotic consensus algorithm [39].

We start with a lower bound proof for the network models that include certain deaf graphs. Let G be an arbitrary communication graph. Consider a system with $n \geqslant 3$ processes, and the n communication graphs $F_1, \ldots, F_n$ where F_i is obtained by making p_i deaf in G, i.e., by removing all the edges towards p_i except the self-loop (p_i, p_i): let $\operatorname{deaf}(G) = \{F_1, \ldots, F_n\}$ with $F_i = G \setminus \{(p_j, p_i) \ : \ p_j \in [n] \setminus \{p_i\}\}$.

Theorem 19. *The contraction rate of any asymptotic consensus algorithm for $n \geqslant 3$ processes in a network model that includes* $\mathrm{deaf}(G)$ *is greater or equal to* $1/2$.

Proof. We show the stronger statement that for every initial configuration C^0 there is an execution $E = C^0, G^1, C^1, G^2, \dots$ starting at C^0 such that

$$\delta_{\mathcal{N}}(C^r) \geqslant \frac{1}{2^r}\delta_{\mathcal{N}}(C^0) \tag{5.7}$$

for all $r \geqslant 0$. It suffices to show (5.7) for the specific network model $\mathcal{N}' = \mathrm{deaf}(G)$ because $\delta_{\mathcal{N}}(C^r) \geqslant \delta_{\mathcal{N}'}(C^r)$ by Lemma 8 and $\delta_{\mathcal{N}'}(C^0) = \delta_{\mathcal{N}}(C^0)$ by Lemma 13 whenever $\mathcal{N} \supseteq \mathcal{N}'$. We hence suppose $\mathcal{N} = \mathcal{N}'$ in the rest of the proof. The proof is by inductive construction of an execution $E = C^0, G^1, C^1, G^2, \dots$ whose configurations C^r satisfy (5.7). This, applied to an initial configuration with $\delta_{\mathcal{N}}(C^0) > 0$, which exists by Lemma 13, then shows the theorem.

For $r = 0$ the inequality holds trivially.

Now let r be any positive integer and assume that Equation (5.7) holds for r. There are n possible successor configurations based on the applicable communication graphs $F_1, \dots, F_n$. We denote $C_k^{r+1} = F_k.C^r$, for any process k. Further let $Y = Y^*_{\mathcal{N}'}(C^r)$, and $Y_k = Y^*_{\mathcal{N}'}(C_k^{r+1})$.

We will show that there exists some process $\hat{k} \in [n]$ such that

$$\mathrm{diam}(Y_{\hat{k}}) \geqslant \mathrm{diam}(Y)/2 \ . \tag{5.8}$$

We then define $G^{r+1} = F_{\hat{k}}$ and $C^{r+1} = C_{\hat{k}}^{r+1}$. By (5.8) and the induction hypothesis, we have

$$\delta_{\mathcal{N}'}(C^{r+1}) \geqslant \frac{\delta_{\mathcal{N}'}(C^r)}{2} \geqslant \frac{1}{2^{r+1}}\delta_{\mathcal{N}'}(C^0) \ , \tag{5.9}$$

i.e., Equation (5.7) holds for $r + 1$.

Assume by contradiction that $\mathrm{diam}(Y_k) < \mathrm{diam}(Y)/2$ for all $k \in [n]$. Recall that process p_i is deaf in F_i and has the same in-neighbors in all the communication graphs F_j with $p_j \neq p_i$. Since $n \geqslant 3$, for any pair of processes p_i, p_j we may select a process p_ℓ different from p_i and p_j such that p_ℓ has the same in-neighbors in F_i as in F_j. Lemma 12 with the assumption that F_ℓ is in $\mathcal{N}$ shows that for any pair of processes p_i, p_j, we have

$$Y_i \cap Y_j \neq \emptyset \ . \tag{5.10}$$

By Lemma 10, there exist $k, k' \in [n]$ such that $\mathrm{diam}(Y_k \cup Y_{k'}) = \mathrm{diam}(Y)$. In particular, we can choose $p_i = p_k$ and $p_j = p_{k'}$, which implies that

$$\mathrm{diam}(Y) = \mathrm{diam}(Y_k \cup Y_{k'}) \leqslant \mathrm{diam}(Y_k) + \mathrm{diam}(Y_{k'}) < \mathrm{diam}(Y) \tag{5.11}$$

which is a contradiction and concludes the proof. □

Note that the network model $\mathrm{deaf}(K_n)$ is a subset of the network model that contains all non-split communication graphs. Hence the lower bound holds and since Algorithm 2 is applicable the claim of a tight bound follows. In fact it would even be sufficient to reduce $\mathrm{deaf}(G)$ to the graphs F_i, F_j, F_l with $p_i, p_j, p_\ell \in [n]$.

5.2 Relation between asymptotic consensus and exact consensus and generalized bounds

We next show properties of subsets of the network model $\mathcal{N}$ that form $\beta_{\mathcal{N}}$-classes.

Lemma 20. *Let $\mathcal{N}$ be a network model and let $\mathcal{N}' \subseteq \mathcal{N}$ be a $\beta_{\mathcal{N}}$-class. Then $G\alpha^*_{\mathcal{N}'}H$ and $G\beta_{\mathcal{N}'}H$ for all $G, H \in \mathcal{N}'$.*

Proof. Let $G, H \in \mathcal{N}'$. Since $G\beta_{\mathcal{N}}H$, there is a q and $H_0, \dots, H_q \in \mathcal{N}$ and $K_1, \dots, K_q \in \mathcal{N}$ such that (i) $G = H_0$ and $H = H_q$ (ii) $H_r\beta_{\mathcal{N}}G$ and $K_r\beta_{\mathcal{N}}G$ for all $r \in [q]$, and (iii) $H_{r-1}\alpha_{\mathcal{N},K_r}H_r$ for all $r \in [q]$. Condition (ii) implies $H_0, \dots, H_q \in \mathcal{N}'$ and $K_1, \dots, K_q \in \mathcal{N}'$ since they belong to the same $\beta_{\mathcal{N}}$-class as G, i.e., $\mathcal{N}'$. Since all H_r are in $\mathcal{N}'$, condition (iii) can be strengthened to $H_{r-1}\alpha_{\mathcal{N}',K_r}H_r$ for all $r \in [q]$.

But this means that the pair (G, H) is in the transitive closure of the union of the relations $\alpha_{\mathcal{N}',K_1}$, …, $\alpha_{\mathcal{N}',K_q}$, and thus in $\alpha^*_{\mathcal{N}'}$. Hence $\alpha^*_{\mathcal{N}'} = \mathcal{N}' \times \mathcal{N}'$, i.e., the first part of the lemma.

To show the second part, define relation $\tilde{\beta} = \mathcal{N}' \times \mathcal{N}'$, which, as we just proved, is included in $\alpha^*_{\mathcal{N}'}$. But it also satisfies the closure property in $\mathcal{N}'$. Since $\tilde{\beta}$ is the coarsest equivalence relation on $\mathcal{N}'$, we thus have $\beta_{\mathcal{N}'} = \tilde{\beta} = \mathcal{N}' \times \mathcal{N}'$, i.e., the second part of the lemma. □

Definition 21 ([46, Definition 4.5])**.** *A network model $\mathcal{N}$ is called* source-incompatible *if*

$$\bigcap_{G \in \mathcal{N}} \mathcal{R}(G) = \emptyset \ .$$

The proof of Coulouma et al. [46] actually shows a stronger version of their theorem (they focus on binary consensus), stated below:

Theorem 22 (Generalization of Theorem 4.10 in [46])**.** *Let $\mathcal{N}$ be a network model. Exact consensus is solvable in $\mathcal{N}$ if and only if each $\beta_{\mathcal{N}}$-class is not source-incompatible.*

We start with showing a generalization of Lemma 12, which allows us to induce non-empty intersection of valencies.

Lemma 23. *Let C be a configuration of an asymptotic consensus algorithm $\mathcal{A}$ for $\mathcal{N}$. For all configurations C in an execution of $\mathcal{A}$ in $\mathcal{N}$, and for all $G, H, K \in \mathcal{N}$, if $G\alpha_{\mathcal{N},K}H$ then $Y^*_{\mathcal{N}}(G.C) \cap Y^*_{\mathcal{N}}(H.C) \neq \emptyset$.*

Proof. By the definition of $G\alpha_{\mathcal{N},K}H$ it is $\mathrm{In}_{\mathcal{R}(K)}(G) = \mathrm{In}_{\mathcal{R}(K)}(H)$. Hence, together with Lemma 11, it follows that $G.C \sim_i H.C$ for all nodes i in $\mathcal{R}(K)$. We consider an execution E in which C occurs at some $t_0 - 1$, G is the communication graph at t_0 and all following graphs are equal to K. Analogously, let E' be an execution identical to E except that the communication graph at round t_0 is H instead of G. By inductive application of Lemma 11, we show that for all $t \geqslant r_0$, we have $C^r \sim_i C'^r$. In particular, we obtain $y^r_{i,E} = y^r_{i,E'}$. Thus $y^*_E = y^*_{E'}$, which shows that $Y^*_{\mathcal{N}}(G.C)$ and $Y^*_{\mathcal{N}}(H.C)$ intersect. □

We next establish that for network models in which exact consensus is not solvable, asymptotic consensus algorithms must have initial configurations that can be extended to executions with different limit outputs.

Lemma 24. *Let $\mathcal{N}$ be a network model in which exact consensus is not solvable. Then, for all asymptotic consensus algorithms $\mathcal{A}$, there exists an initial configuration C^0 such that $Y^*_{\mathcal{N}}(C^0)$ is not a singleton.*

More precisely, for every $\Delta > 0$, there exists an initial configuration C^0 such that $\Delta\big(y^0\big) \leqslant \Delta$ and $\delta_{\mathcal{N}}(C^0) \geqslant \Delta/n$.

Proof. We assume without loss of generality that $d = 1$. If not, we embed the initial values in any 1-dimensional affine subspace.

Let $\mathcal{N}' \subseteq \mathcal{N}$ be any source-incompatible $\beta_{\mathcal{N}}$-class, which exists by Theorem 22. Consider the $n+1$ initial configurations $C^0_{(k)}$ where $0 \leqslant k \leqslant n$ with initial values

$$y^0_{i,(k)} = \begin{cases} \Delta & \text{if } i \leqslant k \\ 0 & \text{if } i > k \ . \end{cases}$$

For all these initial configurations, we have $\Delta\big(y^0_{(k)}\big) \leqslant \Delta$. Define $a(k) = \inf Y^*_{\mathcal{N}'}\big(C^0_{(k)}\big)$ and $b(k) = \sup Y^*_{\mathcal{N}'}\big(C^0_{(k)}\big)$. By Validity, $Y^*_{\mathcal{N}'}\big(C^0_{(0)}\big) = \{0\}$ and $Y^*_{\mathcal{N}'}\big(C^0_{(n)}\big) = \{\Delta\}$, which means $a(0) = b(0) = 0$ and $a(n) = b(n) = \Delta$. There exists some k with $1 \leqslant k \leqslant n$ such that $b(k-1) \leqslant b(k) - \Delta/n$ since otherwise $0 = b(0) > b(n) - \Delta = 0$. Because $\mathcal{N}'$ is source-incompatible, for every process k, there exists a communication graph $G_{(k)} \in \mathcal{N}'$ such that $k \notin S\big(G_{(k)}\big)$. Since $C^0_{(k-1)} \sim_i C^0_{(k)}$ for all $i \in S\big(G_{(k)}\big)$, choosing two executions with all communication graphs equal to $G_{(k)}$ shows that $Y^*_{\mathcal{N}'}\big(C^0_{(k-1)}\big) \cap Y^*_{\mathcal{N}'}\big(C^0_{(k)}\big) \neq \emptyset$, which implies $a(k) \leqslant b(k-1)$. Combining both inequalities gives $a(k) \leqslant b(k) - \Delta/n$ and shows that $\delta_{\mathcal{N}'}\big(C^0_{(k)}\big) = b(k) - a(k) \geqslant \Delta/n$. We hence choose the initial configuration $C^0 = C^0_{(k)}$.

This shows $\delta_{\mathcal{N}}(C^0) \geqslant \delta_{\mathcal{N}'}(C^0) \geqslant \Delta/n$ by Lemma 8 and concludes the proof. □

This finally allows us to derive one of our main results of this section: a characterization of network models in which exact consensus is solvable by the topological structure of valencies of asymptotic consensus algorithms.

Theorem 25. *Let $\mathcal{N}$ be a network model. Exact consensus is solvable in $\mathcal{N}$ if and only if there exists an asymptotic consensus algorithm $\mathcal{A}$ for $\mathcal{N}$ such that $Y^*_{\mathcal{N}',\mathcal{A}}(C^0)$ is either a singleton or disconnected for all network models $\mathcal{N}' \subseteq \mathcal{N}$ and all initial configurations C^0 of $\mathcal{A}$.*

Proof. ($\Rightarrow$): Assume that exact consensus is solvable in $\mathcal{N}$, an let $\mathcal{A}'$ be an algorithm that solves exact consensus in $\mathcal{N}$. Let $\mathcal{A}$ be the algorithm derived from $\mathcal{A}'$ in that deciding is replaced by setting its output variable to the decision value of $\mathcal{A}'$ and not changing it anymore. Before the decision of algorithm $\mathcal{A}'$, algorithm $\mathcal{A}$ outputs its initial value. Then $\mathcal{A}$ is an asymptotic consensus algorithm in $\mathcal{N}$. Further, from Validity of exact consensus, for any initial configuration C^0, the valency $Y^*_{\mathcal{N},\mathcal{A}}(C^0)$ is a subset of the set of initial values in C^0. As the set of initial values of C^0 is finite, so is $Y^*_{\mathcal{N},\mathcal{A}}(C^0)$ and, by Lemma 8, also $Y^*_{\mathcal{N}',\mathcal{A}}(C^0)$ for all $\mathcal{N}' \subseteq \mathcal{N}$. Since any finite set is either a singleton or disconnected, the claim follows.

($\Leftarrow$): We assume without loss of generality that $d = 1$. If not, we embed the initial values in any 1-dimensional affine subspace.

We proceed by means of contradiction. Assume that exact consensus is unsolvable in $\mathcal{N}$. We will show that for all asymptotic consensus algorithms $\mathcal{A}$ for $\mathcal{N}$, there exists an initial configurations C_0 and a network model $\mathcal{N}' \subseteq \mathcal{N}$ such that $Y^*_{\mathcal{N}',\mathcal{A}}(C^0)$ is a nontrivial interval.

By Theorem 22, there is a source-incompatible $\beta_{\mathcal{N}}$-class. Choose $\mathcal{N}'$ to be equal to such a class. We choose C^0 via Lemma 24 such that $Y^*_{\mathcal{N}'}(C^0)$ is not a singleton.

To show that $Y^*_{\mathcal{N}'}(C^0)$ is connected, we assume to the contrary that it is not and derive a contradiction. The set $Y^*_{\mathcal{N}'}(C^0)$ not being connected means the existence of some $z \notin Y^*_{\mathcal{N}'}(C^0)$ such that

$$\exists z_1, z_2 \in Y^*_{\mathcal{N}'}(C^0)\colon\ z_1 < z < z_2\ . \tag{5.12}$$

We will inductively construct an execution $E = C^0, G^1, C^1, G^2, \dots$ such that

$$\exists z_1, z_2 \in Y^*_{\mathcal{N}'}(C^r)\colon\ z_1 < z < z_2 \tag{5.13}$$

for all $r \geqslant 0$. Setting $m(r) = \inf Y^*_{\mathcal{N}'}(C^r)$ and $M(r) = \sup Y^*_{\mathcal{N}'}(C^r)$, we then have $m(r) \leqslant z \leqslant M(r)$ by (5.13) and $M(r) - m(r) = \delta_{\mathcal{N}'}(C^r) \to 0$ by Convergence and Agreement. Hence $\lim\limits_{r\to\infty} m(r) = \lim\limits_{r\to\infty} M(r) = z$, which means

$$\lim_{r\to\infty} Y^*_{\mathcal{N}'}(C^r) = \bigcap_{r\geqslant 0} Y^*_{\mathcal{N}'}(C^r) = \{z\}\ ,$$

where the first equality follows from Lemma 9. In particular $z \in Y^*_{\mathcal{N}'}(C^0)$, which gives the desired contradiction.

It thus suffices to construct execution E satisfying (5.13). Assume that (5.13) holds for a given $r \geqslant 0$ and let $z_1^{(r)}, z_2^{(r)} \in Y^*_{\mathcal{N}'}(C^r)$ with $z_1^{(r)} < z < z_2^{(r)}$. By Lemma 9, it follows that there are communication graphs $G, H \in \mathcal{N}'$ with $z_1^{(r)} \in Y^*_{\mathcal{N}'}(G.C)$ and $z_2^{(r)} \in Y^*_{\mathcal{N}'}(H.C)$. By Lemma 20, we have $G\alpha^*_{\mathcal{N}'}H$. Thus there exists a chain $G = H_0, H_1, \dots, H_q = H \in \mathcal{N}'$ and communication graphs $K_1, \dots, K_q \in \mathcal{N}'$ such that $H_{s-1}\alpha_{\mathcal{N}',K_s}H_s$ for all $s \in [q]$. From Lemma 23 we thus know that

$$Y^*_{\mathcal{N}'}(H_{s-1}.C) \cap Y^*_{\mathcal{N}'}(H_s.C) \neq \emptyset \tag{5.14}$$

for all $s \in [q]$. Set $f(s) = \inf Y^*_{\mathcal{N}'}(H_s.C)$ and $g(s) = \sup Y^*_{\mathcal{N}'}(H_s.C)$ for $s \in \{0, \dots, q\}$, and

$$\hat{s} = \min\left\{s \in \{0, \dots, q\} \mid g(s) > z\right\}\ .$$

Then $f(0) \leqslant z_1^{(r)} \leqslant g(0)$ and $f(q) \leqslant z_2^{(r)} \leqslant g(q)$. The quantity $\hat{s}$ is well defined since $g(q) \geqslant z_2^{(r)} > z$. We show $f(\hat{s}) < z$ by distinguishing two cases:

1. $\hat{s} = 0$: Then $f(\hat{s}) = f(0) \leqslant z_1^{(r)} < z$.
2. $\hat{s} \geqslant 1$: Then, by (5.14) and the definition of $\hat{s}$, we have $f(\hat{s}) \leqslant g(\hat{s}-1) < z$.

In both cases, we showed $f(\hat{s}) < z < g(\hat{s})$. Choosing $G_{r+1} = H_{\hat{s}}$ and $C^{r+1} = G^{r+1}.C^r$, we hence proved (5.13) for $r+1$. This concludes the proof. □

We next introduce the *α-diameter* of a network model $\mathcal{N}$, which we will then (cf. Theorem 27 and Corollary 28) show to be directly linked to a nontrivial lower bound on the contraction rate in $\mathcal{N}$ if exact consensus is not solvable in $\mathcal{N}$. Note, that in the case where exact consensus is solvable in $\mathcal{N}$, the optimal contraction rate always is 0, obtained by a reduction argument to exact consensus.

Definition 26. *Let $\mathcal{N}$ be a network model. The* α-diameter *of $\mathcal{N}$ is the smallest $D \geqslant 1$ such that for all $G, H \in \mathcal{N}$ there exist communication graphs $H_0, \dots, H_q \in \mathcal{N}$ and $K_1, \dots, K_q \in \mathcal{N}$ with $q \leqslant D$ such that $G = H_0$, $H = H_q$, and $H_{s-1}\alpha_{\mathcal{N},K_s}H_s$ for all $s \in [q]$. In case it does not exists we set $D = \infty$.*

Observe that, for the network model $\{H_0, H_1, H_2\}$ from Theorem 18, we obtain $D = 2$. Further, for network model deaf(G), where G is an arbitrary communication graph G, we have $D = 1$. The following theorem and corollary thus generalize Theorems 18 and 19 to arbitrary network models in which exact consensus is not solvable.

Theorem 27. *Let $\mathcal{N}$ be a network model in which exact consensus is not solvable. The contraction rate of any asymptotic consensus algorithm in $\mathcal{N}$ is greater or equal to $1/(D+1)$ where D is the α-diameter of $\mathcal{N}$.*

Proof. We show the stronger statement that for every initial configuration C^0 there is an execution $E = C^0, G^1, C^1, G^2, \dots$ starting at C^0 such that

$$\delta_{\mathcal{N}}(C^r) \geqslant \frac{1}{(D+1)^r}\delta_{\mathcal{N}}(C^0) \tag{5.15}$$

for all $r \geqslant 0$. This, applied to an initial configuration with $\delta_{\mathcal{N}}(C^0) > 0$, which exists by Lemma 24, then shows the theorem.

For the case $D = \infty$, the above statement follows trivially. We hence suppose $D < \infty$. The proof is by inductive construction of an execution $E = C^0, G^1, C^1, G^2, \dots$ whose configurations C^r satisfy (5.15).

For $r = 0$ the inequality trivially holds.

Now let t be any non negative integer and assume that Equation (5.15) holds for t. By Lemma 10, there exist $G, H \in \mathcal{N}$ such that $\operatorname{diam}\big(Y^*_{\mathcal{N}}(C^r)\big) = \operatorname{diam}\big(Y^*_{\mathcal{N}}(G.C^r) \cup Y^*_{\mathcal{N}}(H.C^r)\big)$. Because the α-diameter of $\mathcal{N}$ is equal to $D < \infty$, there exist communication graphs $H_0, \dots, H_q \in \mathcal{N}$ and $K_1, \dots, K_q \in \mathcal{N}$ with $q \leqslant D$ such that $G = H_0$, $H = H_q$, and $H_{s-1}\alpha_{\mathcal{N},K_s}H_s$ for all $s \in [q]$.

Define $Y = Y^*_{\mathcal{N}}(C^r)$ and $Y_s = Y^*_{\mathcal{N}}(H_s.C^r)$. We have $\operatorname{diam}(Y) = \operatorname{diam}(Y_0 \cup Y_q)$ by choice of $G = H_0$ and $H = H_q$. We show that there exists some $s \in \{0, \dots, q\}$ such that $\operatorname{diam}(Y_s) \geqslant \operatorname{diam}(Y)/(q+1)$ and then set $G^{r+1} = H_s$ and $C^{r+1} = H_s.C^r$. Then, by the induction hypothesis, we have

$$\delta_{\mathcal{N}}(C^{r+1}) \geqslant \frac{\delta_{\mathcal{N}}(C^r)}{q+1} \geqslant \frac{\delta_{\mathcal{N}}(C^r)}{D+1} \geqslant \frac{1}{(D+1)^{r+1}}\delta_{\mathcal{N}}(C^0) \ , \tag{5.16}$$

i.e., Equation (5.15) holds for $r+1$.

Assume by contradiction that $\operatorname{diam}(Y_s) < \operatorname{diam}(Y)/(q+1)$ for all $s \in \{0, \dots, q\}$. By Lemma 23, we have $Y_{s-1} \cap Y_r \neq \emptyset$ for all $s \in [q]$. Inductively, we can easily prove

$$\operatorname{diam}\big(\bigcup_{u=0}^{s} Y_u\big) < \frac{s+1}{q+1} \cdot \operatorname{diam}(Y) \tag{5.17}$$

for all $s \in \{0, \dots, q\}$. In particular for $s = q$, which leads to $\operatorname{diam}(Y) \leqslant \operatorname{diam}(Y_0 \cup Y_q) < \operatorname{diam}(Y)$, which is a contradiction and concludes the proof. □

Direct application of Theorem 27 to a network model $\mathcal{N}$ in which exact consensus is not solvable may yield a trivial bound of 0 in case its α-diameter is ∞. We can, however, use Lemma 8 to derive a strictly positive bound for any $\mathcal{N}$ in which exact consensus is not solvable: By Theorem 22 and Lemma 20, any such network model $\mathcal{N}$ contains a source-incompatible $\beta_{\mathcal{N}}$-class, which has a finite α-diameter.

Corollary 28. *Let $\mathcal{N}$ be a network model in which exact consensus is not solvable. The contraction rate of any asymptotic consensus algorithm in $\mathcal{N}$ is greater or equal to $1/(D+1)$ where D is the smallest α-diameter of $\mathcal{N}' \subseteq \mathcal{N}$ in which exact consensus is not solvable.*

Proof. Set $\mathcal{N}' \subseteq \mathcal{N}$ equal to the network model with the smallest α-diameter in which exact consensus is not solvable. Applying Theorem 27 to $\mathcal{N}'$, and Lemma 8 (iv) to $\mathcal{N}'$ and $\mathcal{N}$ yields the corollary. □

5.3 Tight bounds for approximate consensus

In this section, we extend our lower bounds on the contraction rate of asymptotic consensus to lower bounds on the decision time of approximate consensus. In particular, we show optimality of the decision times of the algorithms of Charron-Bost et al. [38] of $\lceil \log_3 \frac{\Delta}{\varepsilon} \rceil$ for $n = 2$ and $\lceil \log_2 \frac{\Delta}{\varepsilon} \rceil$ for $n \geqslant 3$.

We start with the case of two processes in Theorem 29. The proof is by reducing asymptotic consensus to approximate consensus, arriving at a contradiction with Theorem 18 for too fast approximate consensus algorithms.

Theorem 29. *Let $\Delta > 0$ and $\varepsilon > 0$. In a network model of $n = 2$ processes that includes the three communication graphs H_0, H_1, and H_2, all approximate consensus algorithms have an execution with initial diameter $\Delta(y^0) \leqslant \Delta$ and decision time greater or equal to $\log_3 \frac{\Delta}{\varepsilon}$.*

Proof. Assume to the contrary that algorithm $\mathcal{A}$ solves approximate consensus in some network model $\mathcal{N} \supseteq \{H_0, H_1, H_2\}$ that decides in $t < \log_3 \frac{\Delta}{\varepsilon}$ rounds for all vectors of initial values y^0 with $\Delta(y^0) \leqslant \Delta$ and some $\varepsilon > 0$.

Choose any y^0 with $\Delta(y^0) = \Delta$. Define algorithm $\tilde{A}$ by running algorithm $\mathcal{A}$, updating y to the processes' decision values in round t, and then running Algorithm 1 with the initial values $y_i^t = d_i$ from round $t+1$ on. Because Algorithm 1 is an asymptotic consensus algorithm and the decision values y^t of $\mathcal{A}$ satisfy the Validity condition of approximate consensus, algorithm $\tilde{A}$ is an asymptotic consensus algorithm.

Let C^0 be an initial configuration of $\tilde{A}$ with initial values y^0. By the proof of Theorem 18, namely (5.2), there is an execution $E = C^0, G^1, C^1, G^2, \ldots$ starting from C^0 such that

$$\delta_{\mathcal{N}}(C^t) \geqslant \frac{1}{3^t} \cdot \delta_{\mathcal{N}}(C^0) \ . \tag{5.18}$$

We obtain $\delta_{\mathcal{N}}(C^0) = \Delta(y^0) = \Delta$ by Lemma 13 and $\delta_{\mathcal{N}}(C^t) \leqslant \Delta(y^t) \leqslant \varepsilon$ by Validity of Algorithm 1 and ε-Agreement of algorithm $\mathcal{A}$. But this means $t \geqslant \log_3 \frac{\Delta}{\varepsilon}$, a contradiction. □

With a similar proof, but using (5.7) instead of (5.2), we also get the lower bound for approximate consensus with $n \geqslant 3$ processes:

Theorem 30. *Let $\Delta > 0$ and $\varepsilon > 0$. In a network model of $n \geqslant 3$ processes that includes the communication graphs* $\mathrm{deaf}(G)$*, all approximate consensus algorithms have an execution with initial diameter $\Delta(y(0)) \leqslant \Delta$ and decision time greater or equal to $\log_2 \frac{\Delta}{\varepsilon}$.*

In case the network model does not include the graphs $\mathrm{deaf}(G)$, we obtain the following general bound on the termination time:

agents	dimension	network model in which exact consensus is unsolvable		
		non-split with deaf graphs	$\subseteq$ non-split $\subseteq$	rooted
$n = 2$	$d \geqslant 1$	$\frac{1}{3}^*$	$\frac{1}{3}^*$	$\frac{1}{3}^*$
$n \geqslant 3$	$d \in \{1, 2\}$	$\frac{1}{2}^*$	$\left[\frac{1}{D+1}^*, \frac{1}{2}\right]$	$\left[\frac{1}{D+1}^*, \sqrt[n-1]{\frac{1}{2}}\right]$
	$d \geqslant 3$	$\left[\frac{1}{2}^*, \frac{d}{d+1}\right]$	$\left[\frac{1}{D+1}^*, \frac{d}{d+1}\right]$	$\left[\frac{1}{D+1}^*, \sqrt[n-1]{\frac{d}{d+1}}\right]$

Table 5.1: Summary of lower and upper bounds on contraction rates if consensus is not solvable. New lower bounds proved in this work are marked with a *. The three right columns distinguish between the case the network model is (i) non-split and contains deaf(G) for some communication graph G, (ii) is non-split, and (iii) is rooted.

Theorem 31. *Let $\Delta > 0$ and $\varepsilon > 0$. In a network model in which exact consensus is not solvable, all approximate consensus algorithms have an execution with initial diameter $\Delta(y^0) \leqslant \Delta$ and decision time greater or equal to $\log_{D+1} \frac{\Delta}{\varepsilon n}$, where D is the α-diameter of the network model.*

Proof. Assume to the contrary that algorithm $\mathcal{A}$ solves approximate consensus in some network model $\mathcal{N}$ in which exact consensus is not solvable and that decides in $t < \log_3 \frac{\Delta}{\varepsilon}$ rounds for all vectors of initial values y^0 with $\Delta(y^0) \leqslant \Delta$ and some $\varepsilon > 0$.

Define algorithm $\tilde{\mathcal{A}}$ by repeatedly running algorithm $\mathcal{A}$, updating y to the processes' decision values in round kT, and then restarting $\mathcal{A}$ in round $kt + 1$ with the decision values from the previous phase. Then, $\tilde{\mathcal{A}}$ is an asymptotic consensus algorithm.

Let C^0 be an initial configuration of $\tilde{\mathcal{A}}$ with $\Delta\big(y^0\big) \leqslant \Delta$ and $\delta_{\mathcal{N}}(C^0) \geqslant \Delta/n$ By the proof of Theorem 27, namely (5.15), there is an execution $E = C^0, G^1, C^1, G^2, \ldots$ starting from C^0 such that

$$\delta_{\mathcal{N}}(C^t) \geqslant \frac{1}{(D+1)^t} \cdot \delta_{\mathcal{N}}(C^0) \ . \tag{5.19}$$

It is $\delta_{\mathcal{N}}(C^0) \leqslant \Delta(y^0) \leqslant \Delta/n$ and $\delta_{\mathcal{N}}(C^t) \leqslant \Delta(y^t) \leqslant \varepsilon$ by ε-Agreement of algorithm $\mathcal{A}$. But this means $t \geqslant \log_{D+1} \frac{\Delta}{\varepsilon n}$, a contradiction. □

From Theorem 31 and the fact that $\mathcal{N}' \subseteq \mathcal{N}$ implies $\mathcal{E}' \subseteq \mathcal{E}$ for the corresponding sets of executions of algorithm $\mathcal{A}$, we get:

Corollary 32. *Let $\Delta > 0$ and $\varepsilon > 0$. In a network model in which exact consensus is not solvable, all approximate consensus algorithms have an execution with initial diameter $\Delta(y^0) \leqslant \Delta$ and decision time greater or equal to $\log_{D+1} \frac{\Delta}{\varepsilon n}$, where D is the smallest α-diameter of a network model $\mathcal{N}' \subseteq \mathcal{N}$ in which exact consensus is not solvable.*

In conclusion, we summarize the results in this chapter in the Table 5.1.

CHAPTER 6

Non-oblivious message adversaries

In the previous chapter, we used the precise characterization of the consensus solvability under oblivious message adversaries developed by Coulouma et. al [46] for studying asymptotic and approximate consensus. In this chapter, we focus our attention on solving exact consensus under non-oblivious message adversaries. Obviously, the question arises why one would use the relatively unexplored non-oblivious network models in dynamic networks at all.

6.1 From oblivious to non-oblivious message adversaries

We will show by means of three simple message adversaries, two oblivious and the other not, that it makes sense to study non-oblivious message adversaries in combination with exact terminating consensus besides simple research curiosity, for the sake of improving the *assumption coverage*. This term describes how good a given network model captures the behavior of a actual distributed system. A high assumption coverage implies that the restrictions of the network model hold in most (ideally in all) runs of the real system.

The following three message adversaries are based on the simple and well studied 2 process system.

Definition 33 (Impossible message adversary)**.** *Let the network model $\mathcal{N}$ with $n = 2$ contain every possible graph G with the exception of the disconnected graph.*

Definition 34 (Simple message adversary)**.** *Let the network model $\mathcal{N}$ with $n = 2$ contain every possible graph G with the exception of the disconnected and the bidirectionally fully connected graph.*

Definition 35 (Complex message adversary)**.** *Under the network model $\mathcal{M}$ with $n = 2$ every graph sequence is admissible with the exception of the sequence where the processes are bidirectionally connected forever and the sequences that contains a disconnected graph.*

Note that the third adversary is a non-oblivious adversary as it cannot be expressed by a set of graphs. All adversaries contain only sequences without disconnected graphs. We know from [46] that solving exact consensus under the first message adversary is impossible. The second and third adversary, on the other hand, allow the simple Algorithm 3 for solving exact terminating

consensus. In the first case a decision is achieved in round 2, in the second the decision time is unbounded but finite. Note that the algorithm continuously sends messages even after deciding on some value.

Algorithm 3 Simple consensus algorithm for 2 processes

```
   init:
1: y_i ← x_i
   In every round r ⩾ 1:
2: send y_i to all processes in Out_i(r) and receive y^j from all processes p_j in In_i(r)
3: if not yet decided then
4:    if y_i = y_j or In_i(r) = ∅ then
5:       decide y_i
6:    else
7:       y_i ← y_j
8:    end if
9: end if
```

Theorem 36. *Algorithm 3 solves consensus in the simple and the complex message adversary.*

Proof. For both message adversaries it holds that eventually some round r communication graph G^r contains only one edge. In the following round $r+1$ the algorithm will terminate, as both processes have the same value $y_i = v$ and hence the guard in line 4 will be executed. □

A more detailed exploration of the two process model with non-oblivious message adversaries can be found in the Masters thesis of Daniel Pfleger [88].

Notice that the simple message adversary differs from the impossible message adversary in a minimal way, as we remove only one of the three admissible graphs. The same is true for the complex message adversary as we reduce all admissible sequences under the impossible message adversary by exactly one sequence. Hence the message adversaries differ from the impossible message adversary in the minimal way possible in the respective adversary types.

If we compare the number of admissible sequences of the simple and the complex message adversary, we see that for the non-oblivious message adversary it is orders of magnitude larger than for the oblivious message adversary. This implies that the assumption coverage of non-oblivious message adversaries by far exceeds the coverage of oblivious adversaries. And indeed, non-oblivious message adversaries allow for more diverse and complex network model restrictions: Algorithms that work correctly under eventually stabilizing message adversaries are particularly suitable for systems that suffer from uncoordinated boot-up sequences or systems that must recover from massive transient faults. Network connectivity can be expected to improve over time here, e.g., due to improving clock synchronization quality. Since it is usually difficult to determine the time when such a system has reached normal operation mode, algorithms that terminate only after a reasonably stable period has been reached are obviously advantageous. Algorithms that work correctly under short-lived stable periods are particularly interesting, since they have higher coverage and terminate earlier in systems where longer stable periods occur only rarely or even not at all. Note that the occurrence of short-lived stability periods were confirmed experimentally in the case of a prototype wireless sensor network [90].

6.2 Vertex stability

As there are many conceivable approaches possible for defining non-oblivious adversaries, the first step is to pick a set of restrictions and explore them more closely. As a basis we chose the concept of vertex stability in combination with rooted graphs originally introduced in [19]. The former is motivated by the fact that this restriction falls in between categories 1 and 3 outlined in Section Section 2.1, which will be argued later, and that such adversaries are close to the class of strongest adversaries where exact consensus is still solvable. The assumption of rooted graphs makes sense as it is even a necessary condition for the weaker problem of asymptotic consensus and has already been employed for solving asynchronous consensus in systems with initially dead processes in the classic FLP paper [56]. The algorithms rely on a suitable constructed clique, which is just a special case of a rooted component.

We will now define the cornerstones of the message adversaries used in the remaining thesis. This part is mainly based on [22, 102, 108], and will also be used in Kyrill Winklers thesis [107]. Message adversaries based on different network models such as VSSC(d) (Definition 55) and VSSC(k, d) (Definition 89) will be defined via the properties of the sequences of admissible communication graphs. Informally, most of those will rest on the pivotal concept of *source components*, which are strongly connected components in $\mathcal{G}^r$ without *incoming* edges from processes outside the component. The graphs generated by our message adversaries will be required to eventually guarantee source components that are vertex-stable, i.e., consist of the same *set* of nodes (with possibly varying interconnect) during a sufficiently large number of consecutive rounds. It will turn out that vertex-stability guarantees that eventually all members receive information from each other.

Definition 37 (Source Component)**.** *A non-empty set of nodes $S \subseteq V$ is called a round r source component of $\mathcal{G}^r$, if it is the set of vertices of a strongly connected component $\mathcal{S}$ of $\mathcal{G}^r$ and $\forall p_i \in \mathcal{G}^r, p_j \in S : (p_i \to p_j) \in \mathcal{G}^r \Rightarrow p_i \in S$. We denote by $\texttt{sources}(\mathcal{G}^r)$ the set of all source components of $\mathcal{G}^r$, resp. the single source component of $\mathcal{G}^r$, and by $|S|$ the number of nodes in S.*

By contracting strongly connected components (SCCs), it is easy to see that every weakly connected directed simple graph $\mathcal{G}$ has at least one source component, see Lemma 38. Hence, if $\mathcal{G}$ has k source components, it has at most k weakly connected components. Furthermore, if $\mathcal{G}^r$ contains a single source only, contraction leads to a tree, so $\mathcal{G}^r$ must be weakly connected in this case.

Lemma 38. *Any directed graph $\mathcal{G}$ contains at least one and at most n source components (isolated processes), which are all disjoint. If $\mathcal{G}$ contains a single source component S, then $\mathcal{G}$ is weakly connected, and there is a directed (out-going) path from every $p_i \in S$ to every $p_j \in \mathcal{G}$.*

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

To prove the second statement, we use the observation that there is a directed path from u to v in $\mathcal{G}$ if and only if there is a directed path from the vertex C_u (containing u) to the vertex C_v (containing v) in the contracted graph $\mathcal{G}'$. If there is only one source component in $\mathcal{G}$, the above

observations imply that there is exactly one vertex S in the contracted graph $\mathcal{G}'$ that has no incoming edges. Since $\mathcal{G}'$ is connected, S has a directed path to every other vertex in $\mathcal{G}'$, which implies that every process $p_i \in S$ has a directed path to every vertex p_j, as required. □

We now introduce *vertex-stable source components*, abbreviated *stable source* components, as source components that consist of the same set of nodes in every communication graph of a sequence $(\mathcal{G}^r)_{r\in I}$. Note carefully that the interconnect topology of the nodes in S, i.e., the source component $\mathcal{S}$ taken as a subgraph of $\mathcal{G}^r$, as well as the outgoing edges to the remaining nodes $\Pi \setminus S$ in $\mathcal{G}^r$, may be different in every round r in the sequence. The index set I of rounds in $(\mathcal{G}^r)_{r\in I}$ is usually an interval $I = [a, b]$ of $|I| = b - a + 1$ consecutive rounds[1] (we will call $(\mathcal{G}^r)_{r\in I}$ a *consecutive* graph sequence in this case), but can also be an arbitrary index set that is ordered according to increasing round numbers. If a consecutive graph sequence is maximal wrt. S being a stable source, we call S a maximal stable source.

Definition 39 (Stable source)**.** *We say that a sequence $(\mathcal{G}^r)_{r\in I}$ has a stable source S, iff there exists a source S (with possibly different interconnect topology) such that $S \in \texttt{sources}(\mathcal{G}^r)$ for all $r \in I$. If $I = [a, b]$ with $|I| = b-a+1$ is an interval of consecutive rounds $a, a+1, \ldots, b$, $(\mathcal{G}^r)_{r\in I}$ is called a consecutive graph sequence. We call S a maximal stable source of a consecutive graph sequence $(\mathcal{G}^r)^b_{r=a}$, iff S is a stable source of $(\mathcal{G}^r)^b_{r=a}$ but neither of $(\mathcal{G}^r)^b_{r=a-1}$ nor $(\mathcal{G}^r)^{b+1}_{r=a}$.*

We abbreviate I-vertex-stable source component as I-VSSC, and write $|I|$-VSSC if only the length of I matters. Note carefully that we assume $|I| = b - a + 1$ here, since $I = [a, b]$ ranges from the *beginning* of round a to the *end* of round b; hence, $I = [r, r]$ is not empty but rather represents round r.

If $\mathcal{G}^r$ contains only a unique source component we call it *root component* or shortly *root*.

Definition 40 (Root Component)**.** *A source component is called a root (component) R of graph $\mathcal{G}$, if $|\,\texttt{sources}(\mathcal{G})| = 1$. Furthermore a graph is called rooted if it contains a root component.*

Finally, a graph sequence that contains a root with the same set of nodes contains a stable root.

Definition 41 (Stable root)**.** *We say that a sequence $(\mathcal{G}^r)_{r\in I}$ has a stable rood R or is stable rooted, R-stable rooted or shortly R-rooted sequence, if there exists a root component R s.t. $\forall i, j \in I : \texttt{sources}(\mathcal{G}^i) = \texttt{sources}(\mathcal{G}^j) = \{R\}$. We call R a maximal stable root of a consecutive graph sequence $(\mathcal{G}^r)_{r\in I}$ with $I = [a, b]$, iff R is a root of $(\mathcal{G}^r)^b_{r=a}$ but neither of $(\mathcal{G}^r)^b_{r=a-1}$ nor $(\mathcal{G}^r)^{b+1}_{r=a}$.*

We should like to clarify that while "rooted" describes a graph property, "R-rooted" describes a property of a sequence of graphs. It is similar to I-VSSC but the focus here is on the sequence and not on the source component and will make more sense in later chapters where the different terms are used in their respective environments. Given two graphs $\mathcal{G} = \langle V, E\rangle$, $\mathcal{G}' = \langle V, E'\rangle$ with the same vertex-set V, let the *compound graph* $\mathcal{G} \circ \mathcal{G}' := \langle V, E''\rangle$ where $(p_i, p_j) \in E''$ if and only if for some $p_k \in V : (p_i, p_k) \in E$ and $(p_k, p_j) \in E'$.

In order to extend the concept of information propagation in the network from the simple influence relation in Definition 3, we use a notion of *causal past*: Intuitively, a process p_j is in p_i's causal past, denoted $p_j \in \mathrm{CP}^r_i(r')$ if, before its round r computation starts, p_i received

[1] In [19, 21], the term *I-vertex-stable source component* (I-VSSC, or alternatively d-VSSC) has been coined for S being a stable source in $(\mathcal{G}^r)_{r\in I}$ with $I = [a, a + d - 1]$.

information (either directly or transitively, via intermediate messages) that p_j sent immediately after it finished its round r' computation (that is, during the communication at the beginning of round $r'+1$ or later).

Definition 42 (Causal past)**.** *Given a sequence σ of communication graphs that contains rounds a and b, the causal past of process p_i from (the end of) round b down to (the end of) round a is $\mathrm{CP}_i^b(a) = \emptyset$ if $a \geqslant b$ and $\mathrm{CP}_i^b(a) = \mathrm{In}_i(\mathcal{G}^{a+1} \circ \cdots \circ \mathcal{G}^b)$ if $a < b$, where $\mathrm{In}_p(\mathcal{G})$ is the set of nodes that have an outgoing edge to p in G.*

A useful fact about the causal past is that in full-information protocols, where processes exchange their entire state history in every round, we have $p_j \in \mathrm{CP}_i^r(s)$ if and only if, in its round r computation (and hence thereafter), p_i knows already the round s state of p_j.

From the monotonic growth of $\mathrm{CP}_i^b(a)$ (recall the self-loops in every $\mathcal{G}^r$), we can deduce the following corollary:

Corollary 43. *$\mathbf{s}_i^a \rightsquigarrow \mathbf{s}_j^b$ implies $\mathbf{s}_i^a \rightsquigarrow \mathbf{s}_j^{b'}$ for all $b' \geqslant b$. Analogously, $\mathbf{s}_i^a \rightsquigarrow \mathbf{s}_j^b$ implies that $\mathbf{s}_i^{a'} \rightsquigarrow \mathbf{s}_j^b$ for all $a' \leqslant a$.*

To familiarize the reader with our notation, we conclude this section with some technical lemmas. The first one describes the information propagation in a graph sequence containing an ordered set $G = \{\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_n}\}$, $i \neq j \Rightarrow r_i \neq r_j$, and $i > j \Rightarrow r_i > r_j$, of n distinct communication graphs on the same vertex-set Π, where any $\mathcal{G}, \mathcal{G}' \in G$ are rooted but $R(\mathcal{G})$ is not necessarily the same as $R(\mathcal{G}')$. As we have mentioned earlier, every $\mathcal{G} \in G$ contains a rooted spanning tree and is therefore weakly connected. In essence, the lemma shows that, by the end of round r_n, each process p (except some root process of round r_n) transitively received a message from some process q that was sent after q was member of a root component of some graph of G.

Lemma 44. *Let $G = \{\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_n}\}$ be an ordered set of rooted communication graphs on the same vertex-set Π where $|\Pi| = n > 1$. Pick any mapping $f\colon [1,n] \mapsto \Pi$ s.t. $f(i) \in R(\mathcal{G}^{r_i})$. Then $\forall p_i \in \Pi \setminus \{f(n)\}$, $\exists i \in [1, n-1]\colon\ f(i) \in \mathrm{CP}_i^{r_n}(r_i)$.*

Proof. Let $S(i) = \{p_i \in \Pi \mid \exists j \in [1,i]\colon p_i = f(j) \vee f(j) \in \mathrm{CP}_i^{r_i}(r_j)\}$ be the set of nodes that, by round r_i, received the round r_j-state, for some $j \leqslant i$, of $f(j)$ or are equal to $f(j)$. We show by induction that $|S(n)| \geqslant n$.

The base $|S(1)| \geqslant 1$, follows because $f(1) \in S(1)$.

For the step from i to $i+1$, we have the hypothesis $|S(i)| \geqslant i$. Since $S(i) \subseteq S(i+1)$, we only need to consider the case $|S(i)| = i < n$. Let $p_i = f(i+1)$. If $p_i \notin S(i)$, the claim is immediate, so assume $p_i \in S(i)$. As $p_i \in R(\mathcal{G}^{r_{i+1}})$, there is a path from p_i to every $p_j \in \Pi$ in $\mathcal{G}^{r_{i+1}}$. Because we assumed $|S(i)| = i < n$, there is some edge (p_k, p_ℓ) on this path such that $up_k \in S(i)$ and $p_\ell \in \Pi \setminus S(i)$. By construction of $S(i)$ and Definition 42, $p_\ell \in S(i+1)$.

It remains to be shown that $|S(n)| \geqslant n$ implies the lemma. By construction of $S(i)$, $S(n) \setminus \{f(n)\}$ contains only processes p_i for which the claim holds directly or which satisfy $p_i = f(j)$ for some $j \in [1, n-1]$. In case of the latter, since we assume self-loops in every communication graph, $p_i \in \mathrm{CP}_i^{r_n}(r_j)$ also holds. □

The following lemma highlights the most important property of a I-VSSC. It guarantees to spread information among its vertices if the interval I is large enough, as expressed in Corollary 47 below. To prove this, we need a few basic observations and lemmas. Our first observation is a direct consequence of the definition of a strongly connected component.

Observation 45. *Let C denote the set of processes of a strongly connected component of some graph $\mathcal{G}$, and C' be any proper subset of C. Then, there exists a process $p_i \in C'$ s.t. $(p_i \to p_j) \in \mathcal{G}$ for some $p_j \in C \setminus C'$.*

Based on influence and strongly connected components, we can show that a certain amount of information propagation is guaranteed in any strongly connected component C that is vertex-stable, i.e., whose vertex set remains the same, for a given number of rounds. The following Lemma 46 shows that if the number of rounds of the interval of vertex stability $|[a, b]| = b - a + 1$ matches the size of the component minus 1, then for all $p_i \in C$, $\mathtt{s}_i^{a-1}$ reaches every process of C in round b at latest.

Lemma 46. *Let $C \subseteq \Pi$ with $|C| > 1$, let $a \in \mathbb{N}$ and let C form a SCC of $\mathcal{G}^r$ for all $r \in [a + 1, a + |C| - 1]$. Then, $\forall p_i, p_j \in C$, it holds that $\mathtt{s}_i^a \rightsquigarrow \mathtt{s}_j^{a+|C|-1}$.*

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

For the induction start $y = a + 1$, as C with $|C| > 1$ is the vertex-set of a strongly connected component in round $a + 1$, Observation 45 implies that p_i has at least one neighbor such that $\mathtt{s}_i^a \rightsquigarrow \mathtt{s}_j^{a+1}$. By **LOCALITY**, we also have $\mathtt{s}_i^a \rightsquigarrow \mathtt{s}_i^{a+1}$, hence $|P_i^{a+1}| \geqslant 2 = \min\{2, |C|\}$ as required. For the induction step, assume $|P_i^y| \geqslant \min\{y - a + 1, |C|\}$, and consider two cases: (i) If $|P_i^y| < |C|$, then the induction hypothesis implies $y - a + 1 < |C|$, i.e., $y + 1 \in I$. By Observation 45, there is some process in P_i^y that has at least one neighbor $p_j' \notin P_i^y$ in round $y + 1$, which, by **NEIGHBOURHOOD** and **TRANSITIVITY**, results in $|P_i^{y+1}| \geqslant \min\{y+1-a+1, |C|\}$ as required. (ii) If already $|P_i^y| = |C|$, then by **LOCALITY** $|P_i^{y+1}| \geqslant |P_i^y|$, so $|P_i^{y+1}| = |C| \geqslant \min\{y + 1 - a + 1, |C|\}$ holds trivially. □

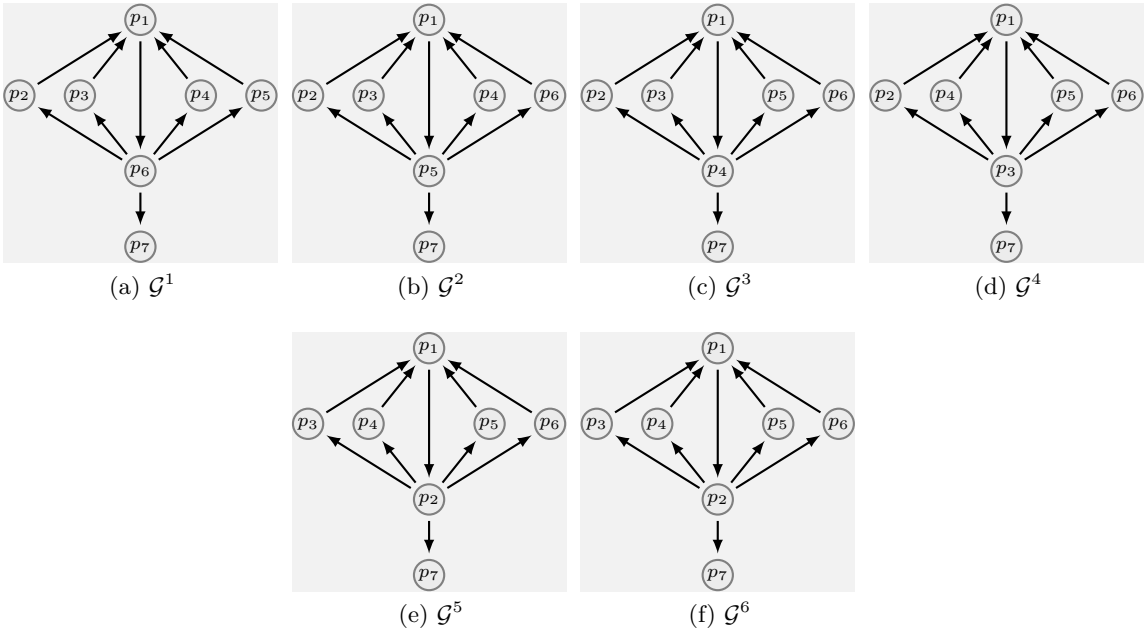
Corollary 47 follows immediately from Lemma 46 and the fact that, by definition, VSSCs are strongly connected components.

Corollary 47. *For every I-vertex-stable source component S with $|S| > 1$ and $I = [a, b]$, it holds that $\forall p_i, p_j \in S$, $\forall x, y \in I$: $y \geqslant x + |S| - 2 \Rightarrow \mathtt{s}_i^{x-1} \rightsquigarrow \mathtt{s}_j^y$.*

Another important information propagation property relates to stable rooted sequences, where the following lemma guarantees an upper bound of $n - 1$ rounds from the root to the rest of the system:

Lemma 48. *Let σ be a graph sequence containing a sequence $S = (\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_{n-1}})$ of $n - 1$ not necessarily consecutive R-stable rooted communication graphs. Then, for all $p_i \in \Pi : R \subseteq \mathrm{CP}_i^{r_{n-1}}(r_0)$ with $r_1 - 1 = r_0$.*

Proof. Pick an arbitrary process $p \in \Pi$, $q \in R$. We show by induction that, for $\ell \in [2, n - 1]$, $|\mathrm{CP}_p^{r_{n-1}}(r_{n-\ell})| \geqslant \ell$ or $q \in \mathrm{CP}_p^{r_{n-1}}(r_{n-\ell})$. For $\ell = 2$, this follows directly from Definition 42 for $p \neq q$ and $p = q$, respectively. For the induction step, we assume that the claim holds for ℓ and show that it holds for $\ell + 1$ as well. If the claim holds because $q \in \mathrm{CP}_p^{r_{n-1}}(r_{n-\ell})$, by Corollary 43, we have $q \in \mathrm{CP}_p^{r_{n-1}}(r_{n-(\ell+1)})$. Thus, assume that $q \notin \mathrm{CP}_p^{r_{n-1}}(r_{n-\ell})$ and $|\mathrm{CP}_p^{r_{n-1}}(r_{n-\ell})| \geqslant \ell$. If it holds that $|\mathrm{CP}_p^{r_{n-1}}(r_{n-\ell})| > \ell$, we get $|\mathrm{CP}_p^{r_{n-1}}(r_{n-(\ell+1)})| \geqslant \ell + 1$ immediately, so assume that $|\mathrm{CP}_p^{r_{n-1}}(r_{n-\ell})| = \ell$. Since $\mathcal{G}^{r_{n-\ell}}$ is R-rooted, there is a path from q to p in $\mathcal{G}^{r_{n-\ell}}$, according to Lemma 38. Because $q \notin \mathrm{CP}_p^{r_{n-1}}(r_{n-\ell})$, there is some process q' on the



(a) $\mathcal{G}^1$ (b) $\mathcal{G}^2$ (c) $\mathcal{G}^3$ (d) $\mathcal{G}^4$

(e) $\mathcal{G}^5$ (f) $\mathcal{G}^6$

Figure 6.1: Graph sequence with diameter 3, dynamic diameter 5 and network depth 6

path from q to p s.t. $q' \notin \mathrm{CP}_p^{r_{n-1}}(r_{n-\ell})$ but $(q' \to p') \in \mathcal{G}^{r_{n-\ell}}$ for some $p' \in \mathrm{CP}_p^{r_{n-1}}(r_{n-\ell})$. By Definition 42, $\mathrm{CP}_p^{r_{n-1}}(r_{n-(\ell+1)}) \supseteq \mathrm{CP}_p^{r_{n-1}}(r_{n-\ell}) \cup \{q'\}$. By the induction hypothesis, therefore $|\mathrm{CP}_p^{r_{n-1}}(r_{n-(\ell+1)})| \geqslant \ell + 1$. □

In order to specify message adversaries that guarantee faster information propagation than guaranteed by the previous lemmas, we introduce appropriate system parameters. Intuitively, they ensure that the information from all nodes in some source component has reached all nodes in the network or at least the processes in the source component if a stable component occurs.

Definition 49 (D-bounded I-VSSC)**.** *A I-VSSC S is D-bounded with dynamic source diameter D, if $\forall p_i, p_j \in S$, $\forall r, r' \in I$: $r' \geqslant r + D - 1 \Rightarrow \mathtt{s}_i^{r-1} \rightsquigarrow \mathtt{s}_j^{r'}$.*

Corollary 47 revealed that every sufficiently long I-VSSC S guarantees $D \leqslant |S| - 1$; all sufficiently long VSSCs hence necessarily give $D \leqslant n - 1$. Choosing some $D < n - 1$ can be used to force the message adversary to speed-up information propagation accordingly. For example, we show in Section 6.2 that certain expander graph topologies ensure $D = O(\log n)$. Next we expand this notion to the whole system.

Definition 50 (E-influencing I-VSSC)**.** *A I-VSSC S is E-influencing with dynamic network depth E, if $\forall p_i \in S$, $\forall p_j \in \Pi$, $\forall r, r' \in I$: $r' \geqslant r + E - 1 \Rightarrow \mathtt{s}_i^{r-1} \rightsquigarrow \mathtt{s}_j^{r'}$.*

Analogous versions of Lemma 46 and Corollary 47 are easily established.

We will later show that processes need to know some estimate of D or E for solving consensus: Without this knowledge, it is impossible to locally verify a necessary condition for solving consensus, namely, the ability of some process to disseminate its initial value system-wide.

We note that, by definition, for $|I| < D$ and $|I| < E$, an I-VSSC is vacuously D-bounded and E-influencing. While it might be tempting to assume a connection between the graph diameter and the dynamic source diameter, resp. the dynamic network depth, in general, these notions

are independent of each other. To illustrate this, Figure 6.1 depicts an example where the graph diameter is constant even though, due to p_1, the dynamic source diameter and the dynamic network depth are in the order of the number of vertices. It is straightforward to generalize this example to n vertices.

To formalize information propagation from source components to the rest of the network in the general case with more than a single source component per communication graph $\mathcal{G}^r$, one has to account for the fact that a process p_j outside any source component could be reachable from *multiple* source components. Intuitively speaking, this allows modelling dynamic networks that do not "cleanly" partition. Similarly to Lemma 46, the following Lemma 51 shows that there is a guaranteed information propagation from at least one process of the set of VSSCs to every process in the system, provided all occurring source components are I-VSSCs with $|I| \geqslant n-1$.

Lemma 51. *Let $n \geqslant 2$ and $R = \{S_1, S_2 ... S_\ell\}$ be a set of $\ell \geqslant 1$ I-VSSCs with $I = [a+1, a+n-1]$ such that, for any $r \in I$, every source component of $\mathcal{G}^r$ is in R. Then, for all $p_j \in \Pi$, it holds that $\exists S \in R$ such that $\mathtt{s}_i^a \rightsquigarrow \mathtt{s}_j^{a+n-1}$ for some $p_i \in S$.*

Proof. Let $R' = \bigcup_{S \in R} S$ denote the set of processes of all VSSCs of R. First, we show an analogue of Observation 45 for processes outside any source component of R: In every $\mathcal{G}^r$, $r \in I$, at least one process of $\Pi \setminus R'$ has an incoming edge from a process contained in some S of R. Suppose that this is not the case. Then, contracting the strongly connected components of $\mathcal{G}^r$ yields at least one node, contracted entirely from nodes of $\Pi \setminus R'$, with no incoming edges. Hence, some source component of $\mathcal{G}^r$ consists entirely of nodes from $\Pi \setminus R'$ and thus cannot be in R. This contradicts the assumptions made on R.

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

For the induction start $r = a+1$, **LOCALITY** implies that $P_R(a+1)$ contains all processes in R', in addition to at least one process of $\Pi \setminus R'$, secured by our equivalent of Observation 45. Hence, $|P_R(a+1)| \geqslant 2 = \min\{2, n\}$ as required. For the induction step, assume $|P_R(r)| \geqslant \min\{r-a+1, n\}$, and consider two cases: (i) If $|P_R(r)| < n$, then the induction hypothesis implies $r - a + 1 < n$, i.e., $r + 1 \in I$. Since $R' \subseteq P_R(a+1) \subseteq P_R(r)$, there is at least one process $p'_j \notin P_R(r)$ that must be contained in $\Pi \setminus R'$; thus, **NEIGHBORHOOD** and **TRANSITIVITY** in conjunction with our equivalent of Observation 45 secure $|P_R(r+1)| \geqslant \min\{r+1-a+1, n\}$. (ii) If already $|P_R(r)| = n$, then $|P_R(r+1)| \geqslant |P_R(r)|$ by **LOCALITY**, so $|P_R(r+1)| = n \geqslant \min\{r+1-a+1, n\}$ holds trivially. □

Again, we introduce a parameter H that allows a more fine-grained modelling of the information propagation in a dynamic network than just assuming the worst case $n-1$ secured by Lemma 51. For this purpose, Definition 52 generalizes Definition 50 from a single I-VSSC to a set R of I-VSSCs. If $|I| \geqslant H$ it guarantees that every process in the network receives a message from some member of at least one I-VSSC of R within H rounds. Note carefully, though, that this does not necessarily imply that there exists an H-influencing I-VSSC. In the special case where R is a singleton set, however, the sole member of R is obviously a H-influencing VSSC.

Definition 52 (H-influencing set of I-VSSCs)**.** *A set $R = \{S_1, S_2 ... S_\ell\}$ of $\ell \geqslant 1$ I-VSSCs with $I = [a, b]$ is H-influencing with dynamic network depth H if $\forall p_j \in \Pi$ $\exists S \in R$ s.t. $\forall r \in I$: if $r \geqslant a + H - 1$ then $\mathtt{s}_i^{a-1} \rightsquigarrow \mathtt{s}_j^r$ for some $p_i \in S$.*

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

We conclude this section with an example of a network topology that guarantees that all I-VSSCs are E-influencing for some E that is much smaller than $n - 1$, which justifies why we introduced this parameter (as well as D) explicitly in our model.[2]

An undirected graph $\mathcal{G}$ is an *α-vertex expander* if, for all sets $R \subset V(\mathcal{G})$ of size $\leqslant |V(\mathcal{G})|/2$, it holds that $\frac{|\mathcal{N}(R)|}{|R|} \geqslant \alpha$, where $\mathcal{N}(R)$ is the set of neighbors of R in $\mathcal{G}$, i.e., those nodes in $V(\mathcal{G}) \setminus R$ that have a neighbor in R. (Explicit expander constructions can be found in [64].) As we need an expander property for *directed* communication graphs, we consider, for a vertex/process set R and a round r, both the set $\mathcal{N}^r_+(R)$ of nodes outside of R that are reachable from R and the set of nodes $\mathcal{N}^r_-(R)$ that can reach R in r. Definition 53 ensures an expansion property both for subsets R chosen from source components (property (a)) and other processes (properties (b), (c)).

Definition 53 (Directed Expander Topology)**.** *There is a fixed constant α and a fixed set S such that the following conditions hold for all sets $R \subseteq V(\mathcal{G}^r)$:*

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

The following Lemma 54 shows that (1) Definition 53 does not contradict the existence of a single source component and that (2) these expander topologies guarantee that I-VSSCs are both D-bounded with $D = O(\log n)$ and E-influencing with $E = O(\log n)$.

Lemma 54. *There are sequences of graphs $(\mathcal{G}^r)_{r>0}$ with a single source component in every $\mathcal{G}^r$ where Definition 53 holds and where, for any such run, every I-VSSC is D-bounded and E-influencing with $D = O(\log n)$ and $E = O(\log n)$.*

Proof. We will first argue that *directed* graphs with a single source component exist that satisfy Definition 53. Consider the simple *undirected* graph $\bar{\mathcal{U}}$ that is the union of an α-vertex expander on some I-VSSC S with $I = [a, b]$ and member set S, and an α-vertex expander on $V(\mathcal{G}^r)$. We turn $\bar{\mathcal{U}}$ into a directed graph by replacing every edge $(p_i, p_j) \in E(\bar{\mathcal{U}})$ with oriented directed edges $p_i \to p_j$ and $p_j \to p_i$. This guarantees Properties (a)-(c). In order to guarantee the existence of exactly one source component, we drop all directed edges pointing to S from the remaining graph, i.e., we remove all edges $p_i \to p_j$ where $p_i \notin S$ and $p_j \in S$, which leaves Properties (a)-(c) intact and makes the S from Definition 53 the single source component of the graph. We stress that the actual topologies chosen by the adversary might be quite different from this construction, which merely serves to show the existence of such graphs.

We also recall that our message adversaries like the one given in Definition 55 will rely on I-vertex-stable source components, which only require that the set of vertices remains unchanged, whereas the interconnect topology can change arbitrarily. Adding Definition 53 does of course not change this fact.

We will first show that the "per round" expander topology stipulated by Definition 53 is strong enough to guarantee that every sufficiently long VSSC is D-bounded with $D = O(\log n)$.

[2] An expander topology can be maintained in a dynamic network by using the protocol in [11].

Let S be some I-VSSC with $I = [a, b]$ and $|I| = \Omega(\log n)$. For $i \geqslant 1$, let $\mathcal{P}_i \subseteq S$ be the set of processes p_j in S such that $\mathtt{s}_i^{a-1} \rightsquigarrow \mathtt{s}_j^{a+i-1}$, and $\mathcal{P}_0 = \{p_i\}$. The result $D = O(\log n)$ follows immediately from Lemma 46 if $|S| \in O(\log n)$, so assume that $|S| \in \Omega(\log n)$ and consider some process $p_i \in S$. For round a, Property (a) yields $|\mathcal{P}_1| \geqslant |\mathcal{P}_0|(1+\alpha)$. In fact, for all i where $|\mathcal{P}_i| \leqslant |S|/2$, we can apply Property (a) to get $|\mathcal{P}_{i+1}| \geqslant |\mathcal{P}_i|(1+\alpha)$, hence $|\mathcal{P}_i| \geqslant \min\{(1+\alpha)^i, |S|/2\}$. Let ℓ be the smallest value such that $(1+\alpha)^\ell > |S|/2$, which guarantees that $|\mathcal{P}_\ell| > |S|/2$. That is, $\ell = \left\lceil \frac{\log(|S|/2)}{\log(1+\alpha)} \right\rceil \in O(\log n)$. Now consider any $p_j \in S$ and define $\mathcal{Q}_{i-1} \subset S$ as the set of nodes that causally influence the set $\mathcal{Q}_i$ in round $a+i$, for $\mathcal{Q}_{2\ell+1} = \{p_j\}$. Again, by Property (a), we get $|\mathcal{Q}_{i-1}| \geqslant |\mathcal{Q}_i|(1+\alpha)$, so $|\mathcal{Q}_{2k-i}| \geqslant \max\{(1+\alpha)^i, |S|/2\}$. From the definition of ℓ above, we thus have $|\mathcal{Q}_\ell| > |S|/2$. Since $\mathcal{P}_\ell \cap \mathcal{Q}_\ell \neq \emptyset$, it follows that every $p_i \in S$ influences every $p_j \in S$ within $2\ell \in O(\log n)$ rounds. While the above proof has been applied to the starting round $x = a$ only, it is evident that it carries over literally also for any $x < s - 2\ell$, which shows that S is indeed a D-bounded I-VSSC.

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

This confirms that sequences of communication graphs with $D < n-1$ and $E < n-1$ indeed exists and are compatible with message adversaries such as VSSC(d) stated in Definition 55 below.

6.3 A simple rooted message adversary

In this section, which is based on [22] and will also be presented in Kyrill Winklers thesis [107], we will introduce a message adversary $\text{VSSC}_{D,E}(d)$ that allows to solve consensus for $d \geqslant 2D+2E+2$ in our model. First and foremost, it requires that every $\mathcal{G}^r$ is rooted, i.e., contains only a single source component. Moreover, albeit the processes do not need to know n, they need a priori knowledge of the dynamic source diameter D and the dynamic network depth E from Definition 49 and Definition 50. And finally, our message adversary must guarantee that, eventually, a d-VSSC occurs. Interestingly, whereas $\text{VSSC}_{D,E}(d)$ allows to solve consensus for $d \geqslant 2D+2E+2$, it is too strong for solving other standard problems in dynamic networks such as reliable broadcasting [22].

Since consensus is trivially impossible for an unrestricted message adversary, which may just inhibit any communication in the system, it is natural to consider the question whether weakly connected communication graphs $\mathcal{G}^r$ in every round r allow to solve consensus. However, it is not difficult to see that this does not work, even when all $\mathcal{G}^r = \mathcal{G}$ are the same, i.e., in a static topology: Consider the case where $\mathcal{G}$ contains two source components S_1 and S_2; such a graph obviously exists, cf. Lemma 38 below. If all processes in S_1 start with initial value 0 and all processes in S_2 start with initial value 1, they must decide on their own initial value (by validity and termination) and hence violate agreement. After all, no process in, say, S_1 ever has an incoming link from any process not in S_1.

Therefore, we restrict our attention to message adversaries that guarantee a root component in $\mathcal{G}^r$ for any round r. Figure 2.2 showed a sequence of graphs where this is the case.

Obviously, assuming a root component makes consensus solvable if the root component is static (shown in detail in [18]). In this paper, we allow the root component to change throughout the run, i.e., the (single) root component R of $\mathcal{G}^r$ might consist of a different set of processes in every round r. However, results in [22] prove that a sufficiently long interval of vertex-stability is indispensable for solving consensus in this setting. In the sequel, we will consider the message adversary $\mathrm{VSSC}_{D,E}(d)$ stated in Definition 55, which enforces the dynamic source diameter D and the dynamic network depth $E \geqslant D$ and is parametrized by some stability window duration $d > 0$.

Definition 55 (Consensus message adversary $\mathrm{VSSC}_{D,E}(d)$)**.** *For $d > 0$, the message adversary $VSSC_{D,E}(d)$ is the set of all sequences of communication graphs $(\mathcal{G}^r)_{r>0}$, where*

(i) for every round r, $\mathcal{G}^r$ contains a root component,

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

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

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

Lemma 56 (Properties of $\mathrm{VSSC}_{D,E}(d)$)**.** *In every sequence $(\mathcal{G}^r)_{r>0}$ of communication graphs feasible for $VSSC_{D,E}(d)$,*

(i) there is at least one process p_i such that $\forall p_j \in \Pi$: $\mathbf{s}_i^0 \rightsquigarrow \mathbf{s}_j^{n(n-2)+1}$ holds, where $\mathbf{s}_i^0$ represents p_i's initial state.

(ii) Conversely, for $n > 2$, the adversary can choose some sequence $(\mathcal{G}^r)_{r>0}$ where no process p_i is causally influenced by all other processes p_j, i.e., $\nexists p_i \in \Pi$ s.t. $\exists y$ and $\forall p_j \in \Pi$: $\mathbf{s}_j^0 \rightsquigarrow \mathbf{s}_i^y$.

Proof. Definition 55 guarantees that there is (at most) one source component in every $\mathcal{G}^r$, $r > 0$. Since we have infinitely many graphs in $(\mathcal{G}^r)_{r>0}$ but only finitely many processes, there is at least one process p_i in the root component of $\mathcal{G}^r$ for infinitely many r. Let $r_1, r_2, \ldots$ be this sequence of rounds. Moreover, let $\mathcal{P}_0 = \{p_i\}$, and define for each $i > 0$ the set $\mathcal{P}_i = \mathcal{P}_{i-1} \cup \{p_j : \exists p_c \in \mathcal{P}_{i-1} : p_c \in \mathcal{N}_j^{r_i}\}$.

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

Finally, at most $n(n-2)+1$ rounds are needed until all processes p_j have been influenced by p_i, i.e., $r_{n-1} \leqslant n(n-2)+1$: A pigeonhole argument reveals that at least one process p_i must have been in the root component for $n-1$ times after so many rounds. After all, if every p_i

appeared at most $n-2$ times, we could fill up at most $n(n-2)$ rounds. By the above result, this is enough to secure that some p_i influenced every p_j.

The converse statement (ii) follows directly from considering a static star, for example, i.e., a communication graph where there is one central process p_c, and for all r, $\mathcal{G}^r = \langle \Pi, \{(p_c \to p_j) | p_j \in \Pi \setminus \{p_c\}\}\rangle$. Clearly, p_c cannot be causally influenced by any other process, and for $p_j, p_k \neq p_j \in \Pi \setminus \{p_c\}$ and $\forall x, y\ \mathtt{s}_j^x \rightsquigarrow \mathtt{s}_k^y$ does not hold. On the other hand, this topology satisfies Definition 55, which includes the requirement of at most one source component per round. □

In the light of Lemma 56, it is interesting to relate the message adversary in Definition 55 to the classification of [32]: It is apparent that $\mathrm{VSSC}_{D,E}(d)$ belongs to a class that it is stronger than the weakest class that requests one node that eventually reaches all others, but weaker than the second-weakest class that requests one node that is reached by all. By contrast, models like [68, 72] that assume bidirectionally connected graphs $\mathcal{G}^r$ in every round belong to the strongest classes (Class 10) in [32].

In Theorem 57, we will examine the solvability of several broadcast problems [68] under the message adversary $\mathrm{VSSC}_{D,E}(d)$. It will turn out that none of these are implementable under our assumptions—basically, because there is no guarantee of (eventual) bidirectional communication. This is clearly in contrast to the usual strong bond between some of these problems and consensus in traditional settings.

Theorem 57. *The message adversary $VSSC_{D,E}(d)$ given in Definition 55, for any d, belongs to a class that is between the weakest and second-weakest in [32]. Neither* reliable broadcast*,* atomic broadcast*, nor* causal-order broadcast *can be implemented. Moreover, there is no algorithm that solves* counting*,* k-verification*,* k-token dissemination*,* all-to-all token dissemination*, and* k-committee election*.*

Proof. We first consider reliable broadcast, which requires that when a correct process broadcasts m, every correct process eventually delivers m. Suppose that the adversary chooses the communication graphs $\forall r:\ \mathcal{G}^r = \langle \{p_i, p_j, p_\ell\}, \{(p_i \to p_j), (p_j \to p_\ell)\}\rangle$, which matches Definition 55. Clearly, p_j is a correct process in our model. Since p_i never receives a message from p_j, p_i can trivially never deliver a message that p_j broadcasts.

For the token dissemination problems stated in [68], consider the same communication graphs and assume that there is a token that only p_ℓ has. Since no other process ever receives a message from p_ℓ, token dissemination is impossible.

For counting, k-verification, and k-committee election, we return to the static star round graph $\mathcal{G}^r = \langle \Pi, \{(p_c \to p_j) | p_j \in \Pi \setminus \{p_c\}\}\rangle$ with central node p_c considered in the proof of Lemma 56. As the local history of any process is obviously independent of n here, it is impossible to solve any of these problems. □

6.3.1 A Consensus Algorithm for $\mathbf{VSSC}_{D,E}(2D+2E+2)$

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

The underlying idea of our consensus algorithm is to use flooding to propagate the largest input value to everyone. However, as Definition 55 does not guarantee bidirectional communication between every pair of processes according to (ii) of Lemma 56, flooding is not sufficient: The

largest input value could be hidden at a single process p_i that never has outgoing edges. If such a process p_i would never accept smaller values, it is impossible to reach agreement (without potentially violating validity). Thus, we have to find a way to force p_i to accept also a smaller value.

A well-known technique to do so is *locking* a candidate value. Obviously, we do not want any process to lock its value, but rather some process(es) that will be able to impose their locked value, i.e., can successfully flood the system. In addition, we may allow processes that have successfully locked a value to decide only when they are sure that every other process has accepted their value as well. According to Definition 52, both can be guaranteed when these processes have been in a vertex stable root component long enough, which is guaranteed by $\mathrm{VSSC}_{D,E}(2D + 2E + 2)$.

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

Note that the following arguments and proofs are based on detecting source component and not root components, as processes may never know if there exists more than one source component in some graph $\mathcal{G}^r$. Never the less the message adversary guarantees that every graph is rooted hence every source can be assumed to be a root by the process and thus the following theorems prove correctness of the algorithms.

The Local Network Approximation Algorithm

According to our system model, no process p_i has any initial knowledge of the network. In order to learn about VSSCs, for example, it hence needs to *locally* acquire such knowledge. Process p_i achieves this by means of Algorithm 4, which maintains a *network estimate* A_i in a local variable.[3] A_i is a graph that holds the local estimates of every communication graph $\mathcal{G}^r$ that occurred so far, simply by labeling an edge $(p_i \to p_j)$ with the set of round numbers of every $\mathcal{G}^r$ once p_i received evidence that $(p_i \to p_j)$ was present in round r.

Initially, A_i consists of process p_i only. In every round, every process p_i broadcasts its current A_i and fuses it with the network estimates received from its neighbors. In more detail, p_i updates A_i whenever $p_j \in \mathcal{N}_i^r$, by adding $(p_j \overset{\{r\}}{\to} p_i)$ if p_j is p_i's neighbor for the first time, or by updating the label of the edge $(p_j \overset{U}{\to} p_i)$ to $(p_j \overset{U\cup\{r\}}{\to} p_i)$ (Line 5 and 7). Moreover, p_i also receives A_j from p_j and uses this information to update its own knowledge: The loop in Line 11 ensures that p_i has an edge $(p_\ell \overset{T\cup T'}{\to} p_i,)$ for each $(p_\ell \overset{T'}{\to} \pm)$ in A_j, where T is the set of rounds previously known to p_i.

Given A_i, we use $A_i|t$ with[4] $0 < t \leqslant r$ to denote the current estimate of $\mathcal{G}^t$ contained in A_i.

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

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

Formally, $A_i|t$ is the graph induced by the set of edges

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

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

Algorithm 4 finally provides an externally callable function `InStableSource`(I), which will be used by the core consensus consensus algorithm to find out whether the calling process p_i was member in an I-VSSC S and to query the set of all members of S. We will prove in Lemma 59 below that p_i is a member of a I-VSSC if $A_i|t$ is strongly connected and consists of the same non-empty set S of processes for all $t \in I$. Informally, this is due to the fact that the members of an I-VSSC will not be able to acquire knowledge of the topology outside S within I, as they do not have incoming links from outside.

Algorithm 4 *Local Network Approximation (Process p_i)*

Provides externally callable function `InStableSource`(I).

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

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

We start our analysis of Algorithm 4 with Lemma 58, which shows that $A_i|t$ under approximates $\mathcal{G}^t$ in a way that consistently includes neighborhoods. Its proof uses the trivial invariant asserting $A_i|t = \langle \{p_i\}, \emptyset \rangle$ at the end of every round $r < t$.

Lemma 58. *If $A_i|t$ contains $(p_k \to p_\ell)$ at the end of some round r, then (i) $(p_k \to p_\ell) \in \mathcal{G}^t$, i.e., $A_i|t \subseteq \mathcal{G}^t$, and (ii) $A_i|t$ also contains $(p_m \to p_\ell)$ for every $p_m \in \mathcal{N}_\ell^t \subseteq \mathcal{G}^t$.*

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

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

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

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

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

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

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

The following Lemma 59 shows that locally detecting $A_i|t$ to be strongly connected (in Line 17 of Algorithm 4) implies that p_i is in the source component of round t. This result rests on the fact that $A_i|t$ under approximates $\mathcal{G}^t$ (Lemma 58.(i)), but does so in a way that never omits an in-edge at any process $p_j \in A_i|t$ (Lemma 58.(ii)).

Lemma 59. *If the graph $C_i|t$ (Line 17) with $t < r$ is non-empty in round r, then p_i is member of S, the source component of $\mathcal{G}^t$.*

Proof. For a contradiction, assume that $C_i|t$ is non-empty (hence $A_i|t$ is an SCC by Line 17), but $p_i \notin S$. Since p_i is always included in any A_i by construction and $A_i|t$ under approximates $\mathcal{G}^t$ by Lemma 58.(i), this implies that $A_i|t$ cannot be the source component of $\mathcal{G}^t$. Rather, $A_i|t$ must contain some process p_k that has an in-edge $(p_j \to p_k)$ in $\mathcal{G}^t$ that is not present in $A_i|t$. As p_k and hence some edge $(p_j \overset{t}{\to} p_k)$ is contained in $A_i|t$, because it is an SCC, Lemma 58.(ii) reveals that this is impossible. □

From the definition of the function `InStableSource`(I) in Algorithm 4 and Lemma 59, we get the following Corollary 60.

Corollary 60. *If the function* `InStableSource`(I) *evaluates to $S \neq \emptyset$ at process p_i in round r, then $\forall x \in I$ where $x < r$, it holds that p_i is a member of S and S is the source component of $\mathcal{G}^x$.*

The following Lemma 61 proves that, in a sufficiently long $I = [a, b]$ with a I-vertex-stable source component S, every member p_i of S detects an SCC for round a (i.e., $C_i|a \neq \emptyset$) with a latency of at most D rounds (i.e., at the end of round $a + D$). Informally speaking, together with Lemma 59, it asserts that if there is an I-vertex-stable source component S for a sufficiently long interval I, then a process p_i observes $C_i|a \neq \emptyset$ from the end of round $a + D$ on if and only if $p_i \in S$.

Lemma 61. *Consider an interval of rounds $I = [a, b]$, such that there is a D-bounded I-vertex-stable source component S and assume $|I| = b - a + 1 > D$. Then, from the end of round $a + D$ onwards, we have $C_i|a = S$, for every process in $p_i \in S$.*

Proof. Consider any $p_j \in S$. At the beginning of round $a + 1$, p_j has an edge $(p_k \xrightarrow{T} p_j)$ in its approximation graph A_j with $a \in T$ if and only if $p_k \in \mathcal{N}_j^a$. Since processes always merge all graph information from other processes into their own graph approximation, it follows from the definition of a D-bounded I-vertex-stable source component (Definition 49) in conjunction with the fact that $a + 1 \leqslant b - D + 1$ that every $p_i \in S$ has these in-edges of p_j in its graph approximation by the end of round $a + 1 + D - 1$. Since S is a vertex-stable source-component, it is strongly connected without in-edges from processes outside S. Hence $C_i|a = S$ from the end of round $a + D$ on, as asserted. □

This immediately gives us the following Corollary 62, which ensures that in a sufficiently long I-VSSC S, with $I = [a, b]$ and member set S, every $p_i \in S$ detects its membership in the J-VSSC S, $J = [a, b - D] \subseteq I$, with a latency of at most D rounds.

Corollary 62. *Consider an interval of rounds $I = [a, b]$, with $|I| = b - a + 1 > D$, such that there is a D-bounded vertex-stable source component S. Then, from the end of round b on, a call to* `InStableSource`$([a, b - D])$ *returns S at every process in S.*

Together, Corollary 60 and Corollary 62 reveal that `InStableSource`(.) *precisely* characterizes the caller's actual membership in the $[a, b - D]$-VSSC S in the communication graphs from the end of round b on.

Core consensus algorithm for VSSC$_{D,E}(2D + 2E + 2)$

As explained in Section 6.3.1, the core consensus algorithm stated in Algorithm 5 builds upon the network approximation algorithm given as Algorithm 4: Relying on Corollary 60, every process uses `InStableSource` provided by Algorithm 4 to detect whether it has been in the vertex-stable source component of some past round(s). Since Corollary 62 reveals that `InStableSource` has a latency of up to $D \leqslant E$ rounds for reliably detecting that a process is in the vertex-stable source component of some (interval of) rounds, our algorithm (conservatively) looks back D rounds in the past when locking a value.

In more detail, Algorithm 5 proceeds as follows: Initially, no process has locked a value, that is, $locked_i =$ FALSE and $lockRound_i = 0$. Processes try to detect whether they are privileged by evaluating the condition in Line 16. When this condition is true in some round ℓ, they lock the current value (by setting $locked_i =$ TRUE and *lockRound* to the current round), unless $locked_i$ is already TRUE. Note that our locking mechanism does not actually protect the value against being overwritten by a larger value being also locked in ℓ; it locks out only those values that have older locks $l < \ell$.

Algorithm 5 Solving Consensus; code for process p_i

```
1: Simultaneously run Algorithm 4.

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

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

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

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

When the process p_m that had the largest value in the source component of round ℓ detects that it has been in a vertex-stable source component in all rounds ℓ to $\ell + E$ (Line 21), it can decide on its current value. As all other processes in that source component must have had p_m's value imposed on them, they can decide as well. After deciding, a process stops participating in the flooding of locked values, but rather (Line 6) floods the network with $\langle \text{DECIDE}, x \rangle$. At the point when the stability window guaranteed by Definition 55 with $d = 2D + 2E + 2$ is large enough to allow every process to receive this message, all processes will eventually decide.

Before we turn our attention to the correctness proof of Algorithm 5, we need to define how the network approximation algorithm and the core consensus algorithm are combined to form a joint algorithm in our computation model. Let $m\_appr_i^{r-1}$ be the information to be broadcast by the network approximation algorithm and $m\_c_i^{r-1}$ the information to be broadcast by the consensus algorithm in round r. Process p_i actually performs the following steps in round r:

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

(ii) Receive all messages based on $\mathcal{G}^r$.

(iii) At the end of round r,

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

Note carefully that this joint execution scheme implies that when `InStableSource()` is called in the above step (iii.2) of the consensus algorithm, the network approximation algorithm is already in the state $\mathtt{s}_{p_i}^r$ reached at the end of round r, so A_i has already been updated with the information received in round r. Consequently, according to Corollary 60 and Corollary 62, a call to `InStableSource`(I) with $I = [a, b - D]$ by p_i in the computing step at the end of round b (or a later round) returns $S \neq \emptyset$ *precisely* when a I-VSSC S containing p_i existed.

Our correctness proof starts with the validity property of consensus according to Definition 6.

Lemma 63 (Validity)**.** *Every decision value is the input value of some process.*

Proof. Processes decide either in Line 13 or in Line 22. When a process decides via the former case, it has received a $\langle \text{DECIDE}, x_j \rangle$ message, which is sent by p_j if and only if p_j has decided on x_j in an earlier round. In order to prove validity, it is thus sufficient to show that processes can only decide on some process' input value when they decide in Line 22, where they decide on their current estimate x_i. Let the round of this decision be r. The estimate x_i is either p_i's initial value, or was updated in some round $r' \leqslant r$ in Line 15 from a value received by way of one of its neighbors' $\langle lockRound, x \rangle$ message. In order to send such a message, p_j must have had $x_j = x$ at the beginning of round r', which in turn means that x_j was either p_j's initial value, or p_j has updated x_j after receiving a message in some round $r'' < r$. By repeating this argument, we will eventually reach a process that sent its initial value, since no process can have updated its decision estimate prior to the first round. □

The following Lemma 64 states a number of properties maintained by our algorithm when the first process p_i has decided. Essentially, they say that there has been a vertex-stable source component in the interval $I = [\ell - D - 1, \ell + E]$ centered around the lock round ℓ (but not earlier), and asserts that all processes in that source component chose the same lock round ℓ.

Lemma 64. *Suppose that process p_i decides in round r, no decisions occurred before r, and $\ell = lockRound_i^{r-1}$, then*

(i) p_i is in the I-vertex-stable source component S with $I = [\ell - D - 1, \ell + E]$,

(ii) $\ell + E \leqslant r \leqslant \ell + E + D$,

(iii) $S \neq S'$, where S' is the source component of $G^{\ell - D - 2}$, and

(iv) all processes in S executed Line 19 in round ℓ, and no process in $\Pi \setminus S$ can have executed Line 19 in a round $\geqslant \ell$.

Proof. Item (i) follows since Line 16 has been continuously TRUE since round ℓ and from Lemma 59. As for item (ii), $\ell + E \leqslant r$ follows from the requirement of Line 21, while $r \leqslant \ell + E + D$ follows from (i) and the fact that by Lemma 61 the requirement of Line 21 cannot be, for the first time, fulfilled strictly after round $\ell + E + D$. From Lemma 61, it also follows that if $S = S'$, then the condition in Line 16 would return true already in round

$\ell - 1$, thus locking would occur already in round $\ell - 1$. Since p_i did not lock in round $\ell - 1$, (iii) must hold. Finally, from (i), (iii), and Lemma 61, it follows that every other process in S also has $\texttt{InStableSource}([\ell - D - 1, \ell - D]) = \text{TRUE}$ in round ℓ. Moreover, due to (iii), $\texttt{InStableSource}([\ell-1-D-1, \ell-1-D]) = \text{FALSE}$ in round $\ell-1$, which causes all the processes in S (as well as those in $\Pi \setminus S$) to set *locked* to 0. Since $\texttt{InStableSource}([\ell' - D - 1, \ell' - D])$ cannot become true for any $\ell' \geqslant \ell$ at a process $p_j \in \Pi \setminus S$, as $C_j|r = \emptyset$ for any $r \in I$ by Corollary 60, (iv) also holds. □

The following Lemma 65 asserts that if a process decides, then it has successfully imposed its proposal value on all other processes.

Lemma 65 (Agreement)**.** *Suppose that process p_i decides in Line 22 in round r and that no other process has executed Line 22 before r. Then, for all p_j, it holds that $x_j^{r-1} = x_i^{r-1}$.*

Proof. Using items (i) and (iv) in Lemma 64, we can conclude that p_i was in S, the vertex-stable source component of rounds $\ell = lockRound_i^{r-1}$ to $\ell + E$, and that all processes in it S have locked in round ℓ. Therefore, in the interval $[\ell, \ell + E]$, ℓ is the maximal value of *lockRound*. More specifically, all processes p_j in S have $lockRound_j = \ell$, whereas all processes p_k in $\Pi \setminus S$ have $lockRound_k < \ell$ during these rounds by Lemma 64.(iv). Let $p_m \in S$ have the largest proposal value $x_m^\ell = x_{max}$ among all processes in S. Since p_m is in S, there is a causal chain of length at most E from p_m to any $p_j \in \Pi$. Note carefully that guaranteeing this property requires item (ii) of Definition 55, as the first decision (in round r) need not occur in the eventually guaranteed $2D + 2E + 2$-VSSC but already in some earlier "spurious" VSSC.

Since no process executed Line 22 before round r, no process will send DECIDE messages in $[\ell, \ell + E]$. Thus, all processes continue to execute the update rule of Line 15, which implies that x_{max} will propagate along the aforementioned causal path to p_j. □

Theorem 66 (Consensus under $\text{VSSC}_{D,E}(2D + 2E + 2)$)**.** *Let r_{ST} be the beginning of the stability window guaranteed by the message adversary $VSSC_{D,E}(2D + 2E + 2)$ given in Definition 55. Then, Algorithm 5 in conjunction with Algorithm 4 solves consensus by the end of round $r_{ST} + 2D + 2E + 1$.*

Proof. Validity holds by Lemma 63. Considering Lemma 65, we immediately get agreement: Since the first process p_i that decides must do so via Line 22, there are no other proposal values left in the system.

Observe that, so far, we have not used the liveness part of Definition 55. In fact, Algorithm 5 is always safe in the sense that agreement and validity are not violated, even if there is no vertex-stable source component.

We now show the termination property. By Corollary 62, we know that every process in $p_i \in S$ evaluates the predicate $\texttt{InStableSource}([r_{ST}, r_{ST} + 1]) = \text{TRUE}$ in round $\ell = r_{ST} + D + 1$, thus locking in that round. Furthermore, Definition 55 and Corollary 62 imply that at the latest in round $d = \ell + E + D$ every process $p_i \in S$ will evaluate the condition of Line 21 to TRUE and thus decide using Line 22. Thus, every such process p_i will send out a message $m = \langle \text{DECIDE}, x_i \rangle$. By Definition 52 and Definition 55, we know that every $p_j \in \Pi$ will receive a DECIDE message at the latest in round $d + E = \ell + D + 2E = r_{ST} + 2D + 2E + 1$ and decide by the end of this round. □

We conclude our considerations regarding consensus under our eventually stabilizing message adversary $\mathrm{VSSC}_{D,E}(d)$ by pointing out that the upper bound $2D + 2E + 2$ and and the lower bound $E - 1$ of the stability interval d presented in [22] match up to a small constant factor. The algorithms presented and mentioned in the following chapters will eventually close this gap.

6.4 Optimizing stability and rootedness

We will now follow up with an algorithm and a proof of optimality, which reduces the stability from $2D + 2E + 2$ to $2E + 1$. Furthermore, we recognized that rootedness is not mandatory for every graph in the sequence. Hence, we not only could improve the stability interval but also learned that the necessary restriction for oblivious message adversaries (Theorem 12) is not necessary in the non-oblivious case. The results were published in [102]. Note that parts of the network definitions will also be included in Kyrill Winklers thesis [107].

Recall that the purpose of our stabilizing message adversary is to allow an unbounded (but finite) initial period of "chaotic" behavior, where the communication graphs can be arbitrary: Unlike for $\mathrm{VSSC}_{D,E}(d)$, any $\mathcal{G}^r$ may be arbitrarily sparse and could contain several root components here. Clearly, one cannot hope to solve consensus during this initial period in general. Eventually, however, the adversary must start to generate suitably restricted communication graphs, which should allow the design of algorithms that solve consensus. We will develop two instances in this paper, and also relate those to the message adversary introduced in in the previous section (published in [21]).

The simple message adversary $\Diamond\mathtt{STABLE}_{n,E}$ defined in this section uses a straightforward means for closing the initial period, which is well-known from eventual-type models in distributed computing: In partially synchronous systems [50], for example, one assumes that speed and communication delay bounds hold forever from some unknown stabilization time on. Analogously, we assume that there is some unknown round r_{ST}, from which on the adversary must behave "nicely" forever. Albeit the resulting message adversary is restricted in its behavior, it provides easy comparability of the performance (in particular, of the termination times) of different consensus algorithms. Moreover, in Section 6.4.2, we will show how to generalize $\Diamond\mathtt{STABLE}_{n,E}$ to a considerably stronger message adversary $\Diamond\mathtt{STABLE}'(E)$, which does not require such a restrictive "forever after" property.

In order to define what "behaving nicely" actually means in the case of $\Diamond\mathtt{STABLE}_{n,E}$, we start from a necessary condition for solving consensus in $(\mathcal{G}^r)_{r=r_{ST}}^{\infty}$: The arguably most obvious requirement here is information propagation from a non-empty set of processes to all processes in the system. According to Lemma 48, this can be guaranteed when there is a sufficiently long sub-sequence of communication graphs in $(\mathcal{G}^r)_{r=r_{ST}}^{\infty}$ with a stable root. Natural candidate choices for feasible graphs would hence be the very same rooted graph $\mathcal{G}$ in all rounds $r \geqslant r_{ST}$, or the assumption that all $\mathcal{G}^r$ are strongly or even completely connected (and hence also rooted). While simple, these choices would impose severe and unnecessary restrictions on our message adversary, however, which are avoided by the following more general definition (that includes these choices as special instances, and hence results in a stronger message adversary):

Definition 67. *We say that $(\mathcal{G}^r)_{r=1}^{\infty}$ has a (unique)* FAE-stable root *R ("forever after, eventually") starting at round $r_{ST} \geqslant 1$, iff R is (i) a maximal stable source of $(\mathcal{G}^r)_{r=r_{ST}}^{\infty}$ and (ii) a maximal root of $(\mathcal{G}^r)_{r=r'}^{\infty}$, for some round $r' \geqslant r_{ST}$.*

$\Diamond$STABILITY contains those communication graph sequences $(\mathcal{G}^r)_{r=1}^{\infty}$ that have a FAE*-stable root R.*

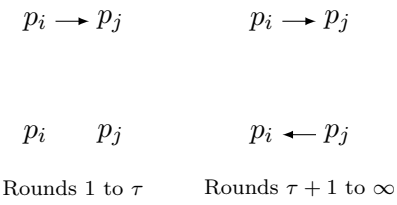


Figure 6.2: Two executions ε_1 (top) and ε_2 (bottom), indistinguishable for p until τ.

Note that the eventual rootedness of $(\mathcal{G}^r)_{r=r_{ST}}^{\infty}$ implied by $\Diamond$STABILITY allows the respective round graphs $\mathcal{G}^r$ to be very sparse: For instance, each $\mathcal{G}^r$ of $(\mathcal{G}^r)_{r=r_{ST}}^{\infty}$ consisting of a chain with the same head but varying body would satisfy the requirement for rootedness.

Whereas the properties guaranteed by $\Diamond$STABILITY will suffice to ensure liveness of the consensus algorithm presented in this chapter, i.e., termination, it is not sufficient for also ensuring safety, i.e., agreement. Consider for instance the top run (execution ε_1) from Figure 6.2, where p_i is connected to q in a chain forever, which is feasible for $\Diamond$STABILITY. In any correct solution algorithm, the head p_i of this chain must eventually decide in some round τ on its initial value x_i. Now consider the execution ε_2, depicted in the bottom of Figure 6.2, where p_i is disconnected until τ and $x_i \neq x_j$. Since ε_2 is indistinguishable for p_i from ε_1 until τ, process p_i will decide x_i at time τ. However, in ε_2, a chain forms with head $p_j \neq p_i$ forever after τ. Since p_j is only aware of its own input value x_j, it can never make a safe decision in this execution.

This is why $\Diamond\mathsf{STABLE}_{n,E}$ needs to combine $\Diamond$STABILITY with another message adversary STICKY(x) that enables our solution algorithm to also ensure safety. The above example illustrates the main problem that we face here: If we allow root components to remain common for too many consecutive rounds in the initial period (before r_{ST}), the members of such a source component (which does not need to be single) cannot distinguish this from the situation where they are belonging to the final FAE-stable root (after r_{ST}). In the previous chapter this problem was void since *all* communication graphs were assumed to be rooted. In the following Definition 68, we require that every source that is common during a sequence of "significant" length $x + 1$ is already the FAE-stable root R. Again, in Section 6.4.2, we will present a significant relaxation of this quite restrictive (but convenient) assumption.

Definition 68. STICKY(x) *contains those communication graph sequences* $\sigma = (\mathcal{G}^r)_{r=1}^{\infty}$*, where every source* S *that is stable for* $> x$ *consecutive rounds in* σ *is the* FAE*-stable root* R *in* σ*.*

What remains is to redefine the network depth such that it is not a property of vertex stable sources but instead a property of rooted graphs.

Definition 69 (Dynamic network depth E)**.** *For all subsequences* $(\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_1+E-1}) \subset \sigma \in \mathsf{DEPTH}_n(E)$ *of non necessarily consecutive* R*-rooted communication graphs, we have* $R \subseteq \mathrm{CP}_i^{r_1+E-1}(r_1 - 1)$ *for every* $p_i \in \Pi$*.*

We are now ready to define our simple eventually stabilizing message adversary $\Diamond\mathsf{STABLE}_{n,E}$, which is the conjunction of the adversaries from Definition 67 and Definition 68, augmented by the additional requirement to always guarantee a dynamic network depth E according to Definition 69:

Definition 70. *The message adversary* $\Diamond\mathsf{STABLE}_{n,E} =$
STICKY$(E) + \Diamond$STABILITY *contains those graph sequences of* STICKY$(E) \cap \Diamond$STABILITY *that are in* $\mathsf{DEPTH}_n(E)$*.*

Note carefully that Definition 67 allows the coexistence of the FAE-stable root R with some other source component $S \neq R$ in communication graphs that occur before R becomes the root (in round r'). However, according to Definition 68, S cannot be stable root for more than E consecutive rounds in this case.

6.4.1 A fast consensus algorithm

We now present our consensus algorithm for the message adversary $\Diamond\mathtt{STABLE}_{n,E}$, which also works correctly under the generalized $\Diamond\mathtt{STABLE}'(E)$ that will be introduced in Section 6.4.2. The algorithm is based on the fact that, from the messages a node receives, it can reconstruct a faithful under-approximation of (the relevant part of) the communication graph of every round, albeit with delay E. Obviously, an algorithm similar to Algorithm 4 can be used for this purpose.

The algorithm stated in Figure 6.3 works as follows: Every process p_i maintains an array $A_i[r]$ that holds the *graph approximation* of $\mathcal{G}^r$, and a matrix $\mathtt{lock}_i[j][r]$ that holds the history of a special value, the *lock-value*, for every known process q and every round r. $A_i^m[r]$ and $\mathtt{lock}_i^m[j][r]$ denote the content of the respective array entry at the end of round m as usual. The first entries of these arrays are initialized to the singleton-graph $A_i^0[0] = (\{p_i\}, \{\})$ resp. to $\mathtt{lock}_i^0[i][0] := x_i$, the input value of p_i, and to $\mathtt{lock}_i^0[j][0] := \perp$ for every $p_j \neq p_i$. Note that $\mathtt{lock}_i[i][m-1]$ can be viewed as p_i's *proposal value* for round m. Every process broadcasts $A_i^{m-1}[r]$ and $\mathtt{lock}_i^{m-1}[j][r]$ in round $m \geqslant 1$, and updates $A_i^m[r]$ and $\mathtt{lock}_i^m[j][r]$, by fusing the information contained in the messages received in round m in a per-round fashion (as detailed below), before executing the round m *core computation* (we will omit the attribute core in the sequel if no ambiguity arises) of the algorithm. Note that the round m core computation for $m \in \{1, \ldots, E\}$ is empty.

In the core computation of some round τ, p_i will eventually decide on the maximum $\mathtt{lock}_i[j][a]$ value for all $p_j \in R$, where R is a stable root of some sequence $(\mathcal{G}^r)_{r=a}^{a+E}$ but not of $(\mathcal{G}^r)_{r=a-1}^{a+E-1}$, as detected locally in $A_i^\tau[*]$. Note carefully that τ may be different for processes other than p_i.

Two mechanisms are central to the algorithm for accomplishing this: First, any process p_i that, in its round m computation, locally detects a root component R in $A_{p_i}^m[m-E]$ will "lock" it, i.e., assign the maximum value of $\mathtt{lock}_i^m[j][m-E]$ for any $p_j \in R$ to $\mathtt{lock}_i^m[i][m]$. Second, if process p_i detects in round τ that a graph sequence had a stable root R' for at least $E+1$ rounds in its graph approximation, starting in round a, p_i will decide, i.e., set y_i to the maximum of $\mathtt{lock}_i^\tau[j][a]$ among all $p_j \in R'$.

Informally, the reason why this algorithm works is the following: From detecting an R-rooted sequence of length $\geqslant E+1$, p_i can infer, by the $\mathtt{STICKY}(E)$ property of our message adversary, that the entire system is about to lock p_i's decision value. Moreover, by exploiting the information propagation guarantee given by Lemma 48, we can be sure that, after p_i's decision in round τ, every other process p_j decides (in some round $\tau' \geqslant \tau$) on the very same value: Under $\Diamond\mathtt{STABLE}_{n,E}$, it decides because the root that triggered the decision of p_i is the FAE-stable root; under $\Diamond\mathtt{STABLE}'(E)$, p_j decides on the same value because it will never assign a value different from $\mathtt{lock}_i[i][\tau]$ to $\mathtt{lock}_j[x][\tau'']$ for any $\tau'' \geqslant \tau'$ and any known process x. Finally, termination is guaranteed since every p_i will eventually find an R-rooted sequence of duration at least $E+1$ because of $\Diamond\mathtt{STABILITY}$ resp. $\Diamond\mathtt{STABILITY}'$.

Graph approximation and lock maintenance

Our algorithm relies on a simple mechanism for maintaining the graph approximation $A_i[r]$ and the array of lock values $\mathtt{lock}_i[j][r]$ at every process p_i already used in Algorithm 4: In every

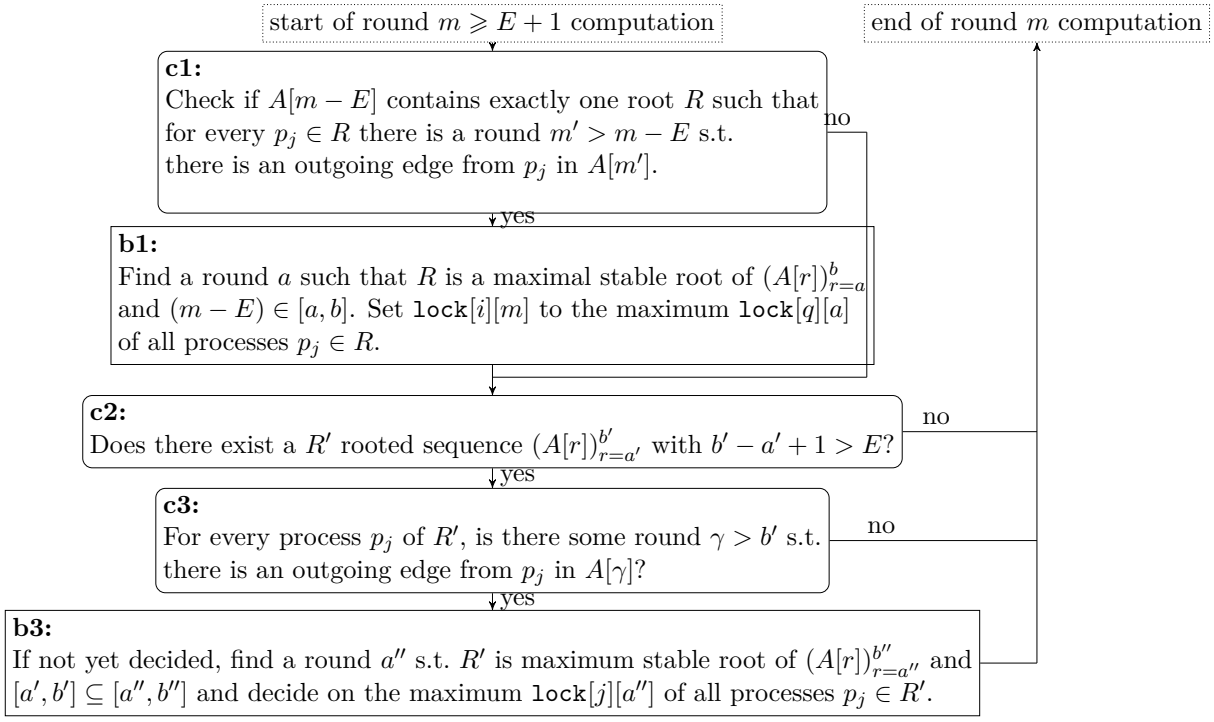


Figure 6.3: Round $m \geqslant E+1$ core computation step of our consensus algorithm for process p_i. $A[r] = A_i^m[r]$ denotes p_i's round m view of $\mathcal{G}^r$ provided by the network approximation algorithm. $\texttt{lock}[j][r]$ denotes $\texttt{lock}_i^m[j][r]$, where $\texttt{lock}[i][m]$ represents p_i's proposal value for the next round $m+1$. Note that $c1$ and $c2$ are only relevant if $|R| = 1$ and $|R'| = 1$, respectively, as the graph approximation may not yet know the outgoing edges at the time the root is detected.

round, each process p_i broadcasts its current $A_i[*]$ and $\texttt{lock}_i[*][*]$ and updates all entries with new information possibly obtained in the received approximations from other processes. In more detail, an edge $(p_j \to p_k)$ will be present in $A_i^m[r]$ at the end of round $m \geqslant r$ if either $p_i = p_k$ and p_i received a message from p_j in round r, or if p_i received $A_\ell^{r''}[r]$ for $m \geqslant r'' \geqslant r$ from some process p_ℓ and $(p_j \to p_k) \in A_\ell^{r''}[r]$. Similarly, $\texttt{lock}_i^m[j][r]$ for $r < m$ is updated to $\texttt{lock}_k[j][r] \neq \perp$ whenever such an entry is received from any process p_k; the entry $\texttt{lock}_i^m[j][m]$ for the current round m is initialized to $\texttt{lock}_i[i][m] := \texttt{lock}_i[i][m-1]$ for $p_j = p_i$ and to $\texttt{lock}_i[j][m] := \perp$ for every $p_j \neq p_i$.

Note carefully that we again assume that the round m computation of the approximation algorithm is executed *before* the round m core computing step at every process. Therefore, the round m approximation $A_i^m[*]$ is already available *before* the core computing step of round m at process p_i is executed. Note that we assume that, instead of accessing information provide by Algorithm 4 via a dedicated function, the algorithm in Figure 6.3 accesses the approximation $A_i^r[*]$ directly. We also drop the index from the approximation in the algorithm of process p_i as it obviously accesses its own approximation. Furthermore, the full-information approach of the above implementation incurs sending and storing a large amount of redundant information. Comments related to a more efficient implementation are provided in Section 6.4.2.

The crucial property guaranteed by our graph approximation is again that processes under-approximate the actual communication graph, i.e., that they do not fabricate edges in their approximation. Using our notion of causal past, it is not difficult to prove the following assertion about edges that are guaranteed to exist in the graph approximation:

Lemma 71. *For Algorithm 4 it holds that:* $\mathtt{s}_j^r \rightsquigarrow \mathtt{s}_i^{r'}$ *holds for* $r' > r \Leftrightarrow$ *there exists a process* p_k *s.t.* $(p_j \to p_k) \in A_i^{r'}[r'']$ *for some* $r'' \in (r, r']$.

Proof. "⇒"-direction: If $p_i = p_j$, the claim trivially holds because $\mathcal{G}^r$ contains the self-loop $(p_i \to p_i)$. For $p_i \neq p_j$, since we assume $\mathtt{s}_j^r \rightsquigarrow \mathtt{s}_i^{r'}$, by Definition 42, there exists a round $r'' > r$, such that $\exists \mathtt{s}_k^{r''} \rightsquigarrow \mathtt{s}_i^{r'}$ with $(p_j \to p_k) \in \mathcal{G}^{r''}$. Therefore, p_i must have received the round r'' state of p_k and hence learned about the edge $(p_j \to p_k)$, by round r' via line 5,line 7 or line 13. In other words, $(p_j \to p_k) \in A_p^{r'}[r'']$, as claimed.

"⇐"-direction: Holds by Lemma 58. □

We now present a more abstract view on this mechanism of approximating the communication graph. First, we answer which state information a process needs in order to reliably detect which roots are present in the actual communication graph.

Lemma 72. *Let* $R \in \mathtt{sources}(\mathcal{G}^r)$ *and let there be some process* p_i *and round* r' *such that* $R \subseteq \mathrm{CP}_{p_i}^{r'}(r)$. *For Algorithm 4 it holds that* $R \in \mathtt{sources}(A_i^{r'}[r])$. *Furthermore, there exists a process* p_k *s.t.* $(p_j \to p_k) \in A_i^{r'}[r'']$ *for some* $r < r'' \leqslant r'$.

Proof. Since $R \subseteq \mathrm{CP}_{p_i}^{r'}(r)$, according to Corollary 43, by the end of round r', p_i has received the round r state $\mathtt{s}_j^r$ of all processes $p_j \in R$. In particular, p_i has received all round r in-edges of every process p_j. Hence, R is a strongly connected component of $A_i^{r'}[r]$ and there are no processes $p_k \in \Pi \setminus R$ s.t. $(p_k \to p_j) \in A_i^{r'}[r]$. But then, $R \in \mathtt{sources}(A_i^{r'}[r])$, as asserted. The presence of $(p_j \to p_k)$ in $A_i^{r'}[r'']$ follows directly from Lemma 71. □

We conclude our considerations regarding the graph approximation by looking at what is sufficient from an algorithmic point of view for a process p_i to faithfully determine the root components in some communication graph. In the case where a root component $R \in \mathtt{sources}(\mathcal{G}^r)$ has size $|R| > 1$, we note that as soon as a process p_i knows, in some round r', at least one in-edge $(p_k \to p_j) \in A_i^{r'}[r]$ for each $p_j \in R$, then p_i knows $\mathtt{s}_j^r$ and hence all in-edges of p_j. Consequently, it can reliably deduce that indeed $R \in \mathtt{sources}(\mathcal{G}^r)$.

In the case where $|R| = |\{p_j\}| = 1$, if p_i has no edge $(p_k \to p_j) \in A_i^{r'}[r]$, this is *not* sufficient for concluding that $\{p_j\} \in \mathtt{sources}(\mathcal{G}^r)$: Process p_i seeing no in-edge to a process p_j in the local graph approximation $A_i^{r'}[r]$ happens naturally if $\mathtt{s}_{p_j}^{r-1} \rightsquigarrow \mathtt{s}_{p_i}^{r'}$ and $\mathtt{s}_{p_j}^{r} \not\rightsquigarrow \mathtt{s}_{p_i}^{r'}$, i.e., when the last message p_i received from p_j was sent at the beginning of round r. In order to overcome this issue, process p_i must somehow ascertain that it already received the state $\mathtt{s}_j^r$ of process p_j in round r. In particular, process p_i can deduce this directly from its graph approximation as soon as it observed some outgoing edge from p_j in a round strictly after r.

Let us state this more formally in the following lemma.

Lemma 73. *Let* $R \in \mathtt{sources}(A_i^{r'}[r])$ *for* $r' > r$, *and let, for all processes* $p_j \in R$, *there be a process* p_k *and a round* $r'' \in (r, r']$, *such that* $(p_j \to p_k) \in A_i^{r'}[r'']$. *Then,* $R \in \mathtt{sources}(\mathcal{G}^r)$, *and* $R \subseteq \mathrm{CP}_i^{r'}(r)$.

Proof. By contradiction. Assume that $R \in \mathtt{sources}(A_i^{r'}[r])$, $\forall p_j \in R\, \exists p_k \in \Pi, r'' \in (r, r']$: $(p_j \to p_k) \in A_i^{r'}[r'']$ and $R \notin \mathtt{sources}(\mathcal{G}^r)$. Because of the latter, there exist some processes $p_j \in R$ and $p_\ell \notin R$ with $(p_\ell \to p_j) \in \mathcal{G}^r$. By the presence of the edge $(p_j \to p_k)$ in $A_i^{r'}[r'']$ and Corollary 71, we have $R \subseteq \mathrm{CP}_i^{r'}(r)$. But then, by the assumption that $(p_\ell \to p_j) \in \mathcal{G}^r$, it must also hold that $(p_\ell \to p_j) \in A_i^{r'}[r]$. This, however, contradicts that $R \in \mathtt{sources}(A_i^{r'}[r])$. □

Finally, the way how the lock arrays are maintained by our algorithm implies the following simple results:

Corollary 74. *If $r' > r$, then $\mathtt{s}_{p_j}^{r} \rightsquigarrow \mathtt{s}_{p_i}^{r'}$ implies that also $\mathtt{lock}_i^{r'}[j][r''] = \mathtt{lock}_j^{r''}[j][r'']$ for all rounds $r'' \leqslant r$.*

Lemma 75. *Let m be a round reached by process p_i in the execution. Then, $\mathtt{lock}_i^{m}[i][r] \neq \perp$ for all $0 \leqslant r \leqslant m$.*

Proof. Since $\mathtt{lock}_i^{0}[i][0] = x_p$, it follows from the update rule $\mathtt{lock}_i[i][m] := \mathtt{lock}_i[i][m-1]$ that $\mathtt{lock}_i[i][m] \neq \perp$ for all reached rounds m, provided that the core algorithm never assigns $\perp$ in b1. Since the latter can only assign the maximum of $\mathtt{lock}_i[j][a]$ for all $p_j \in R$ from some earlier round $a \leqslant m - E < m$, the statement of our lemma follows from a trivial induction based on Corollary 74, provided we can guarantee $\mathtt{s}_j^a \rightsquigarrow \mathtt{s}_i^m$. The latter follows immediately from c1 in conjunction with Lemma 73, however. □

Correctness proof

Before proving the correctness of the algorithm given in Figure 6.3 (Theorem 79 below), we first establish two technical lemmas: Lemma 76 reveals that our algorithm terminates for every message adversary MAT that guarantees certain properties (without guaranteeing agreement, though). The complementary Lemma 78 shows that our algorithm ensures agreement (without guaranteeing termination, though) for every message adversary MAA that guarantees certain other properties. Theorem 79 will then follow from the fact that $\Diamond\mathtt{STABLE}_{n,E} \subseteq \mathtt{MAT} \cap \mathtt{MAA}$.

Lemma 76. *The algorithm terminates by the end of round τ under any message adversary MAT that guarantees $\sigma \in \mathtt{DEPTH}_n(E)$ and if for every $\sigma \in \mathtt{MAT}$, there is an R-rooted sequence $(\mathcal{G}^r)_{r=\alpha}^{\beta} \in \sigma$ with $\beta - \alpha + 1 > E$.*

Proof. We show that if process p_i has not decided before round τ, it will do so in round τ. By round τ, every process $p_i \in \Pi$ received $\mathtt{s}_j^{\beta}$ for all $p_j \in R$ by the assumption that $R \subseteq \mathrm{CP}_i^{\tau}(\beta)$. Hence, by Lemma 72 and Lemma 73, for every $p_i \in \Pi$, it holds that R is the root of $\mathtt{sources}(A_i^{\tau}[\beta])$. Furthermore, by Corollary 43, R is in fact the root of $\mathtt{sources}(A_i^{\tau}[r])$ for any $r \in [\alpha, \beta]$. Therefore, process p will pass the check c2 in round τ.

In addition, by the assumption that $R \subseteq \mathrm{CP}_i^{\tau}(\beta)$ and Lemma 72, for every $p_j \in R$, there exists a round $\beta' \in (\beta, \tau]$, s.t. $(p_j \to p_k) \in A_i^{\tau}[\beta']$ for some process p_k. Therefore, process p will pass the check c3 in round τ and decide. □

Lemma 78 below shows that, under message adversaries that guarantee a $\mathtt{EC}(E+1)$-stable root according to Definition 77, the algorithm from Figure 6.3 satisfies agreement.

Definition 77. *We say that a graph sequence $(\mathcal{G}^r)_{r=\alpha}^{\alpha+d}$ has a* $\mathtt{EC}(x+1)$-stable source *("embedded $x+1$-consecutive stable source") R, if (i) $(\mathcal{G}^r)_{r=\alpha}^{\alpha+d}$ has a stable source R and (ii) $(\mathcal{G}^r)_{r=\alpha'}^{\alpha'+x} \subseteq (\mathcal{G}^r)_{r=\alpha}^{\alpha+d}$ has a root R.*

Lemma 78. *Let MAA be a message adversary that guarantees, for every $\sigma \in \mathtt{MAA}$ that $\sigma \in \mathtt{DEPTH}_n(E)$ and that the first subsequence $(\mathcal{G}^r)_{r=\alpha}^{\beta} \subseteq \sigma$ with a maximum stable source R and $\beta - \alpha + 1 > E$ has a $\mathtt{EC}(E+1)$-stable source. Under MAA, if two or more processes decide in our algorithm, then they decide on the same value $\neq \perp$.*

Proof. Let α' and β', with $\beta' - \alpha' + 1 > E$, delimit the maximal period where R is root, as predicted by Definition 77.

Setting $\lambda = \max_{p_j \in R} \texttt{lock}_j^{\alpha}[j][\alpha]$, we show that if an arbitrary process p decides in round τ, it decides on λ and $\lambda \neq \bot$. Assume that p_i decides in some round τ. It follows from c2 and c3 that p_i detects in round τ that R' is the root of $(A_i^{\tau}[r])_{r=a'}^{b'}$ with $b' - a' + 1 > E$, and that, for every $p_j \in R'$, there is a round $\gamma > b'$ where there is an edge (p_j, p_k) in $A_{p_i}^{\tau}[\gamma]$ for some process $p_k \in \Pi$. By Lemma 73, we have that $R' \in \texttt{sources}(\mathcal{G}^r)$ for all $r \in [a', b']$, and $R' \subseteq \mathrm{CP}_i^{\tau}(b')$. Thus, Corollary 74 in conjunction with Lemma 75 confirm that indeed $\lambda \neq \bot$. We distinguish two cases:

Case 1. $[a', b'] \subseteq [\alpha, \beta]$: From the definition of `MAA`, in combination with the fact that $b' - a' + 1 > E$, it follows that $R' = R$: if this was not the case, then either $(\mathcal{G}^r)_{r=\alpha}^{\beta}$ would not be the first sequence of its kind or $(\mathcal{G}^r)_{r=\alpha'}^{\beta'}$ would not be R-rooted.

By b3, p_i will decide on the maximum of $\texttt{lock}_i[j][a'']$, where a'' is a round such that $(A_i^{\tau}[r])_{r=a''}^{b''}$ has a maximum stable source R, $[a'', b''] \supseteq [a', b']$, and $p_j \in R$. Hence, since $R \subseteq \mathrm{CP}_i^{\tau}(b')$ and $\alpha < b'$, it follows from Corollary 43 that $R \subseteq \mathrm{CP}_p^{\tau}(\alpha)$. Thus, by Lemma 72, we have $a'' = \alpha$. According to Corollary 43 in conjunction with Corollary 74, it follows that p_i indeed decides on λ.

Case 2. $[a', b'] \nsubseteq [\alpha, \beta]$: First, observe that $a' > \beta'$: If $a' \leqslant \beta'$ then, because $(\mathcal{G}^r)_{r=\alpha}^{\beta}$ is the first sequence of its kind, we have that $a' \geqslant \alpha$. Thus, since $\mathcal{G}^{\beta'}$ is R-rooted, $R' = R$, and hence $[a', b'] \nsubseteq [\alpha, \beta]$ is a contradiction to the assumption that R is maximal stable in $(\mathcal{G}^r)_{r=\alpha}^{\beta}$.

It follows from this observation and b3 that p_i decides on the maximum value of $\texttt{lock}_i[j][a'']$ for $p_j \in R'$, where $a'' > \beta'$. Thus, to conclude our proof, it suffices to show that $\texttt{lock}_i^{r}[i][r] = \lambda$ for all rounds $r > \beta'$ and all processes $p_i \in \Pi$.

Since $(\mathcal{G}^r)_{r=\beta'-E}^{\beta'}$ is R-rooted, it follows from Definition 69 and Lemma 72 that in round β' every process p_i sets $\texttt{lock}_i^{\beta'}[i][\beta']$ to λ via b1. Moreover, if a process assigns a value to $\texttt{lock}_i[i][m]$ during some round $m \in (\beta', \beta' + E]$ via b1 later on, it follows from the rootedness of $(\mathcal{G}^r)_{r=\beta'-E}^{\beta'}$ and Lemma 73 that the assigned value is also λ.

For $\ell \geqslant \beta' + E$, we show by induction on ℓ that λ is assigned to $\texttt{lock}_i[i][m]$ (if there is any assignment at all), in round m, for all $m \in [\beta', \ell]$ and all processes p. The induction basis is $\ell = \beta' + E$, for which the claim has been established already. For the induction step, assume that the claim holds for the interval $[\beta', \ell]$ and all p. If no process p_i changes its lock value in b1 during the core round $\ell + 1$ computation, i.e., $\texttt{lock}_i^{\ell}[i][\ell] = \texttt{lock}_i^{\ell+1}[i][\ell + 1]$, then the claim follows immediately from the induction hypothesis. Thus, assume that $\lambda = \texttt{lock}_i^{\ell}[i][\ell] \neq \texttt{lock}_i^{\ell+1}[i][\ell+1]$. This means that p_i has successfully passed c1 and hence, by Lemma 73, that there is a root $R'' \in \texttt{sources}(\mathcal{G}^{\ell+1-E})$ with $R'' \subseteq \mathrm{CP}_i^{\ell+1}(\ell + 1 - E)$. If $R'' = R$ is a maximal stable source of $(\mathcal{G}^r)_{r=\alpha}^{\beta}$, by Corollary 43, it follows from the definition of λ and Corollary 74 that p_i assigns $\texttt{lock}_i^{\ell+1}[i][\ell + 1] := \lambda$. Therefore, assume that this is not the case, i.e., $R'' \neq R$. Still, R'' must be a maximal stable source in $(\mathcal{G}^r)_{r=\alpha''}^{\beta''}$ for some $\alpha'' > \beta'$ with $\alpha'' \leqslant \ell + 1 - E$. By the induction hypothesis, $\texttt{lock}_j^{\ell+1-E}[j][r] = \lambda$ for every process p_j of R'' and round $r \in [\beta', \ell]$ and so, in particular, $\texttt{lock}_j^{\ell+1-E}[j][\alpha''] = \lambda$. It follows from Corollary 74 and $R'' \subseteq \mathrm{CP}_i^{\ell+1}(\ell + 1 - E)$ that for all processes $q \in R''$, we have $\texttt{lock}_i^{\ell+1}[j][\alpha''] = \lambda$. Therefore, since, by b1, p_i chooses its new value for $\texttt{lock}_i^{\ell+1}[i][\ell + 1]$ as the maximum of the entries $\texttt{lock}_i^{\ell+1}[j][\alpha'']$, it assigns $\texttt{lock}_i^{\ell+1}[i][\ell + 1] := \lambda$. □

Theorem 79. *The algorithm from Figure 6.3 solves consensus by round $r' + 2E$ under message adversary $\Diamond\mathtt{STABLE}_{n,E}$.*

Proof. According to b3, a process p can decide only on a value in $\mathtt{lock}_p^m[i][*]$ in some round m. By Lemma 75, this value must be $\neq \bot$. Since $\mathtt{lock}_j[j][0]$ is initialized to x_j for any process p_j, and the only assignments $\neq \bot$ to any $\mathtt{lock}_j$ entry are $\mathtt{lock}_k$ entries of other processes, validity follows.

For agreement, recall that $\mathtt{STICKY}(E)$ guarantees that the first sequence $(\mathcal{G}^r)_{r\in I}$ with a stable source R and $|I| > E$ must be the $\mathtt{FAE}$-stable root. Hence, agreement follows from Lemma 78.

For termination, recall that $\Diamond\mathtt{STABILITY}$ guarantees the existence of some round $r' \geqslant r_{ST}$ such that $(\mathcal{G}^r)_{r=r'}^{\infty}$ is R-rooted. This implies that the sequence $(\mathcal{G}^r)_{r=r'}^{r'+E}$ is R-rooted and, by Definition 69, $R \subseteq \mathrm{CP}_p^{r'+2E}(r' + E)$. Lemma 76 thus implies termination by round $r' + 2E$. □

6.4.2 Generalized stabilizing message adversary

The simple message adversary introduced in at the start of Section 6.4 may be criticized due to the fact that the first source component R that is common in at least $E + 1$ consecutive rounds must already be the $\mathtt{FAE}$-stable root that persists forever after. In this section, we will considerably relax this assumption, which is convenient for analysis and comparison purposes but usually unrealistic in practice [89].

In the following Definition 80, we start with a significantly relaxed variant $\Diamond\mathtt{STABILITY}'(x)$ of $\Diamond\mathtt{STABILITY}$ from Definition 67: Instead of requesting an infinitely stable $\mathtt{FAE}$-stable root R, we only require R to be (i) a $\mathtt{EC}(x + 1)$-stable source that starts at r_{ST} and becomes single at $r' \geqslant r_{ST}$, and (ii) to re-appear as a root in at least E not necessarily consecutive later round graphs $\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_E}$. Note that, according to Definition 69, the latter condition ensures $R \subseteq \mathrm{CP}_p^{r_E}(r' + x)$ for all $p_i \in \Pi$ if $\Diamond\mathtt{STABILITY}'(x) \subseteq \mathtt{DEPTH}_n(E)$.

Definition 80. *Every communication graph sequence $\sigma \in \Diamond\mathtt{STABILITY}'(x)$ contains a subsequence $(\mathcal{G}^r)_{r=\alpha}^{\alpha+d}$, which has a $\mathtt{EC}(x + 1)$-stable source R; let $r_{ST} = \alpha$ be its starting round and $r' = \alpha'$ be the time when it becomes single. Furthermore, there are at least E, not necessarily consecutive, R-rooted round graphs $\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_E}$ with $r' + x < r_1 < \cdots < r_E$ in σ.*

Moreover, we relax the $\mathtt{STICKY}(x)$ condition in Definition 68 accordingly: We only require that the first source component R that is common for at least $x + 1$ consecutive rounds in a graph sequence $\sigma = (\mathcal{G}^r)_{r=1}^{\infty}$ is a $\mathtt{EC}(x + 1)$-stable source:

Definition 81. *For every $\sigma \in \mathtt{STICKY}'(x)$, it holds that the earliest subsequence in σ with a maximal stable source R in at least $x + 1$ consecutive rounds actually has a $\mathtt{EC}(x + 1)$-stable source.*

Combining these two definitions results in the following strong version of our stabilizing message adversary.

Definition 82. *The strong stabilizing message adversary $\Diamond\mathtt{STABLE}'(E) = \mathtt{STICKY}'(E) + \Diamond\mathtt{STABILITY}'(E)$ contains all graph sequences in $\mathtt{STICKY}'(E) \cap \Diamond\mathtt{STABILITY}'(E)$ that guarantee a dynamic network depth of E.*

Note carefully that the very first $\texttt{EC}(E+1)$-stable source R' occurring in $\sigma \in \Diamond\texttt{STABLE}'(E)$ need *not* be the $\texttt{EC}(E+1)$-stable source R guaranteed by Definition 80.

The following Lemma 83 shows that the message adversary $\Diamond\texttt{STABLE}'(E)$ is indeed weaker than $\Diamond\texttt{STABLE}_{n,E}$. This is not only favorable in terms of model coverage, but also ensures that an algorithm designed for $\Diamond\texttt{STABLE}'(E)$ works under $\Diamond\texttt{STABLE}_{n,E}$ as well.

Lemma 83. $\Diamond\texttt{STABLE}_{n,E} \subseteq \Diamond\texttt{STABLE}'(E)$

Proof. Pick any graph sequence $\sigma \in \Diamond\texttt{STABLE}_{n,E}$. Since $\sigma \in \Diamond\texttt{STABILITY}$, there exists a round $r' \geqslant r_{ST}$ such that $(\mathcal{G}^r)_{r=r'}^{\infty}$ is R-rooted. But then $(\mathcal{G}^r)_{r=r'}^{r'+E}$ is also R-rooted and there is a set of E additional communication graphs $S = \left\{\mathcal{G}^{r'+E+1}, \ldots, \mathcal{G}^{r'+2E}\right\}$ such that every $\mathcal{G}^r \in S$ is also R-rooted. Hence, σ satisfies $\Diamond\texttt{STABILITY}'(E)$.

Furthermore, σ satisfies $\texttt{STICKY}(E)$. Thus, for the first sequence $(\mathcal{G}^r)_{r=a}^{a+E}$ with stable source R, R must already be the $\texttt{FAE}$-stable root and hence $(\mathcal{G}^r)_{r=r'}^{\infty}$ is R-rooted for some $r' \geqslant a$. Consequently, R is a $\texttt{EC}(x+1)$-stable source starting at a. Hence, σ satisfies $\texttt{STICKY}'(E)$. □

The following Theorem 84 shows that the algorithm from Figure 6.3 also solves consensus under the stronger message adversary $\Diamond\texttt{STABLE}'(E)$:

Theorem 84. *For a graph sequence $\sigma \in \Diamond\texttt{STABLE}'(E)$, let $\mathcal{G}^{r_1}, \ldots, \mathcal{G}^{r_E}$ with $r_1 > r' + E$ denote the E re-appearances of the $\texttt{EC}(E+1)$-stable source R guaranteed by $\Diamond\texttt{STABILITY}'$ according to Definition 80. Then, the algorithm from Figure 6.3 correctly terminates by the end of round $\tau = r_E$.*

Proof. The proof of validity in Theorem 79 is not affected by changing the message adversary.

For the agreement condition, recall that $\texttt{STICKY}'(E)$ guarantees that the first sequence $(\mathcal{G}^r)_{r\in I}$ with stable source R in $E+1$ consecutive rounds has a $\texttt{EC}(E+1)$-stable source. Hence, we can again apply Lemma 78 to prove that the algorithm satisfies agreement.

For the termination condition, recall that for any sequence $\sigma \in \Diamond\texttt{STABILITY}'(E)$ it is guaranteed that there exists some round r' s.t. $(\mathcal{G}^r)_{r=r'}^{r'+E}$ is R-rooted. Furthermore, σ contains at least E not necessarily subsequent R-rooted communication graphs after $r'+E$. The latter implies, by Definition 69, that $R \subseteq \mathrm{CP}_p^{\tau}(r'+E)$ for every process $p \in \Pi$. Hence, we can again apply Lemma 76, which shows that the algorithm indeed terminates by round τ. □

By contrast, the Algorithm 7 from Section 6.6 does not work under $\Diamond\texttt{STABLE}'(E)$. Under an appropriate adversary, this algorithm ensures graceful degradation from consensus to general k-set agreement. This does not allow the algorithm to adapt to the comparably shorter and weaker stability periods of $\Diamond\texttt{STABLE}'(E)$, however. In more detail, $\texttt{VSRC}(n, 4E)$ requires a four times longer period of consecutive stability than $\Diamond\texttt{STABILITY}'(E)$. The adversarial restriction $\texttt{MAJINF}(k)$ that enables k-agreement under partitions in Section 6.6 for $k > 1$, on the other hand, is very weak and thus requires quite involved algorithmic solutions. Nevertheless, despite its weakness, it is not comparable to $\texttt{STICKY}'(E)$.

6.5 Minimal stability for rooted graphs

In this section, which is based on [108], we consider a slightly different message adversaries than in the previous section. We already stated in Section 6.2 that $E \leqslant n-1$ by an analog proof to Lemma 46. The following liveness property, *eventual stability*, ensures that eventually every graph sequence σ has a R-rooted subsequence $\sigma' \subseteq \sigma$ of length x. Herein, Σ denotes the unrestricted message adversary, i.e., the set of all communication graph sequences. Again, parts of the model will also be presented in Kyrill Winklers thesis [107].

Definition 85. $\Diamond\mathsf{GOOD}_n(x) := \{\sigma \in \Sigma \mid \Pi_\sigma = [1,n] \wedge \exists R \subseteq \Pi_\sigma\colon$ *some* $\sigma' \subseteq \sigma$ *with* $|\sigma'| \geqslant x$ *is* R*-rooted* $\}$.

For finite x, $\Diamond\mathsf{GOOD}_n(x)$ alone is insufficient for solving consensus: Arbitrarily long sequences of graphs that are not rooted before the stability phase occurs can fool any consensus algorithm to make wrong decisions. For this reason, we introduce a safety property in the form of the message adversary that generates only rooted graphs.

Definition 86. $\mathsf{ROOTED}_n := \{\sigma \in \Sigma \mid \Pi_\sigma = [1,n] \wedge$ *every* $\mathcal{G}^r$ *of* σ *is rooted* $\}$.

The short-lived eventually stabilizing message adversary $\Diamond\mathsf{STABLE}_{n,E}(E+1)$ used throughout this section adheres to the dynamic network depth E, guarantees that every $\mathcal{G}^r$ is rooted and that every sequence has a subsequence of at least $x = E+1$ consecutive communication graphs with a stable root component. Since processes are aware under which message adversary they are executing, they have common a priori knowledge of the dynamic network depth E and the duration of the stability phase x. Moreover, depending on the variant actually used, they have some knowledge regarding the system size n.

Definition 87. *We call* $\Diamond\mathsf{STABLE}_{n,E}(x) = \mathsf{ROOTED}_n \cap \Diamond\mathsf{GOOD}_n(x) \cap \mathsf{DEPTH}_n(E)$ *the eventually stabilizing message adversary with stability period* x*. For some fixed* E*, we consider the following generalizations:*

- $\Diamond\mathsf{STABLE}_{<\infty,E}(x) = \bigcup_{n\in\mathbb{N}\setminus\{0,1\}} \Diamond\mathsf{STABLE}_{n,E}(x)$
- $\Diamond\mathsf{STABLE}_{\leqslant N,E}(x) = \bigcup_{n=2}^{N} \Diamond\mathsf{STABLE}_{n,E}(x)$

We observe that $\Diamond\mathsf{GOOD}_n(x) \supseteq \Diamond\mathsf{GOOD}_n(E)$ for any $1 \leqslant x \leqslant E$, hence it follows that $\Diamond\mathsf{STABLE}_{n,E}(x) \supseteq \Diamond\mathsf{STABLE}_{n,E}(E)$.

Impossibility Results and Lower Bounds

Even though processes know the dynamic network depth E, for very short stability periods, this is not enough for solving consensus. Theorem 88 shows that consensus is impossible under $\Diamond\mathsf{STABLE}_{<\infty,E}(2E-1)$. That is, if processes do not have access to an upper bound N on the number of processes, solving consensus is impossible if the period x of eventual stability is shorter than $2E$: Here, processes can never be sure whether a stable root component occurred for at least E rounds, albeit detecting such a stable root component is necessary to satisfy validity.

Theorem 88. *Under* $\Diamond\mathsf{STABLE}_{<\infty,E}(x)$ *consensus is impossible for* $0 < x < 2E$.

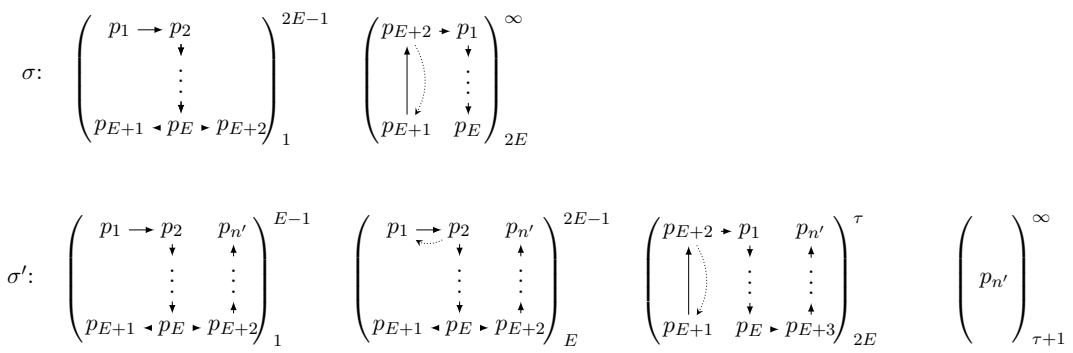


Figure 6.4: Example sequences where process p_{E+1} and p_{E+2} can not distinguish if they are in σ or σ' until round τ and both sequences, by a suitable choice of initial values, eventually have different valencies. A dotted edge represents and edge which is in $\mathcal{G}^i$ if and only if it is not in $\mathcal{G}^{i-1}$. We assume that there are self-loops and there is an edge from every process depicted in the graph to every process not depicted in the graph.

Proof. As we have $\Diamond\mathsf{GOOD}_n(x) \subset \Diamond\mathsf{GOOD}_n(x')$ for $x > x'$, it suffices to show that consensus is impossible under message adversary $\mathsf{MA} = \Diamond\mathsf{STABLE}_{<\infty,E}(2E-1)$.

Pick any $E \in \mathbb{N}$ and suppose some algorithm Alg solves consensus under MA. Let n, resp. n', denote the number of nodes in any communication graph of σ, resp. σ' from Figure 6.4. We provide two admissible executions $\varepsilon, \varepsilon'$ based on σ, resp. σ', of Alg where $\varepsilon \overset{p_{E+1}}{\sim} \varepsilon'$. We show that p_{E+1} decides 0 in ε and hence in ε', whereas process p_n decides 1 in ε'.

Let C^0 be the initial configuration with input values $x_i = 0$ if $i \in [1, E+2]$ and $x_i = 1$ otherwise.

Consider execution $\varepsilon = \langle C^0, \sigma\rangle$ with σ from Figure 6.4, where a dotted edge exists only in every second graph of the sequence, and all processes not depicted have an in-edge from every depicted process. $\sigma \in \mathsf{MA}$, since it guarantees eventual stability for $2E-1$ rounds, adheres to the dynamic network depth E and in every round the communication graph is rooted. By the assumed correctness of Alg, there is a finite round $\hat{\tau}$ by which every process has decided in ε; let $\tau = \max\{\hat{\tau}, 2E\}$. The decision must be 0 because p_{E+1} only ever saw processes that knew of input value 0. This is indistinguishable for p_{E+1} from the execution where all processes did start with input 0, which, according to the validity property of consensus, implies a decision on 0.

Now, consider the execution $\varepsilon' = \langle C^0, \sigma'\rangle$ with σ' from Figure 6.4 and $n' > \tau + E + 3$. Again, $\sigma' \in \mathsf{MA}$, since $(\mathcal{G}^r)_{r=\tau+1}^{\infty}$ is $p_{n'}$-rooted. In every round $r \leqslant \tau$, $p \in \{p_{E+1}, p_{E+2}\}$ have the same view in ε and ε': This is immediately obvious for $1 \leqslant r \leqslant E-1$. For $E \leqslant r \leqslant 2E-1$, the view of p_1 is different, but this difference is not revealed to p by the end of round $2E-1$. Finally, in rounds $2E \leqslant r \leqslant \tau$, the processes $\{p_{E+1}, p_{E+2}\}$ hear only from each other in both executions, hence maintain $\varepsilon \overset{p_{E+1}}{\sim} \varepsilon'$.

Consequently, by round τ, p_{E+1} has decided 0 also in ε'. Yet, by construction of ε', $p_{n'}$ never saw a process that had an input value different from 1. By validity and an analogous argument as above, $p_{n'}$ must hence decide 1 in ε' here, which violates agreement and hence provides the required contradiction. □

This improves the result from [102] and shows that even if every round contains exactly one source component, without knowledge of n the algorithm from the previous section can not be

significantly improved. Our paper [108] provides an algorithm that includes n and thus reduces the interval stability needed to solve consensus even further.

Summary of results on consensus under non-oblivious message adversaries

In the last three sections, we solved consensus under different adversaries by very different algorithmic techniques. We conclude this focus on consensus with a brief comparison of the different approaches. First, note that the network model used in Section 6.3.1 is a subset of both the model in Section 6.4 and Section 6.5, which implies that the solution in Section 6.3.1 is somewhat redundant. For the algorithm in Section 6.3.1, the adversary has to guarantee the largest stability interval coupled with rootedness in every graph. On the plus side, the algorithm is rather simple and, in comparison to the other two, the only one which is memory-efficient. In Section 6.4, we reduced the amount of stability needed and furthermore weakened the rooted per round restriction. This comes at the cost of unbounded memory needed in worst case scenarios. In Section 6.5, the unbounded memory demand is the same as in Section 6.4 but the stability is reduced to a minimum. On the other hand, we need a upper bound on n and rootedness per round.

This shows that depending on the requirements of the system one can optimize for different criteria in the non-oblivious message adversary setting: We provided optimal solutions for stability and rootedness. In the next section, we will expand on weakening the rootedness restriction even further, which will come with its own trade offs.

6.6 From consensus to k-set agreement

In this section (which is based on [22]), we will explore an even stronger message adversary than in Section 6.4, albeit with the aim to solve k-set agreement: The constraint to generate only one decision value gets relaxed to $k \leqslant n$. In turn, k is also a parameter related to the message adversary, i.e., one that relaxes the constraints on the network model.

Definition 89 below defines the generic message adversary $\mathrm{VSSC}_{D,H}(k,d)$, which allows at most k VSSCs per round and guarantees a common window of vertex stability of duration at least d rounds. Note that it involves both the dynamic source diameter D and the dynamic network depth H according to Definition 49 and Definition 52 (that have to be enforced by the message adversary).

Definition 89 (Message adversary $\mathrm{VSSC}_{D,H}(k,d)$)**.** *For $k > 0$, $d > 0$, the message adversary $VSSC_{D,H}(k,d)$ is the set of all sequences of communication graphs $(\mathcal{G}^r)_{r>0}$, where*

(i) for every round r, $\mathcal{G}^r$ contains at most k source components,

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

(iii) for each $(\mathcal{G}^r)_{r>0}$, there exists some $r_{ST} > 0$ and an interval of rounds $J = [r_{ST}, r_{ST}+d-1]$ with a H-influencing set of $1 \leqslant \ell \leqslant k$ D-bounded J-vertex-stable source components $S_1, \ldots, S_\ell$.

Note that the message adversary $\mathrm{VSSC}_{D,H}(k,1)$ guarantees at most k VSSCs in every $\mathcal{G}^r$, $r > 0$.

For $k = 1$, we can relate $\text{VSSC}_{D,H}(k,d)$ and $\text{VSSC}_{D,E}(d)$ given in Definition 55. First, $\text{VSSC}_{D,H}(1,d)$ differs from $\text{VSSC}_{D,E}(d)$ in that item (iii) rests on H-influencing VSSCs (Definition 52) rather than on E-influencing ones (Definition 49). Therefore, every run that is feasible w.r.t. item (iii) of $\text{VSSC}_{D,E}(d)$ for $E := H$ is also feasible w.r.t. item (iii) of $\text{VSSC}_{D,H}(1,d)$. Second, item (ii) of $\text{VSSC}_{D,E}(d)$ is also more demanding than item (ii) of $\text{VSSC}_{D,H}(1,d)$ as it requires all VSSCs to be both D-bounded and E-influencing. Consequently, it follows that $\text{VSSC}_{D,H}(d) = \text{VSSC}_{D,H}(1,d)$.

Note that, despite this relation, [22] shows that $\text{VSSC}_{D,H}(k,d)$ is not sufficient to solve k-set agreement.

6.6.1 Algorithms for k-Set Agreement

In this subsection, we will provide a message adversary MAJINF(k) (Definition 95) that is sufficiently weak for solving k-set agreement if combined with $\text{VSSC}_{D,H}(n, 3D+H)$ (Definition 89). Although we can of course not claim that it is a strongest one in terms of problem solvability (we did not even define what this means), we have some intuitions that it is not too far from the solvability/impossibility border.

Set agreement

To illustrate some of the ideas that will be used in our message adversary for general k-set agreement, we start with the simple case of $n-1$-set agreement (also called *set agreement*) first. A straightforward way to achieve this is to just forbid n different decisions obtained in source components consisting of a single process. Achieving this is easy under the Σ_{n-1}-influence message adversary given in Definition 90, the name of which has been inspired by the Σ_{n-1} failure detector [25].

Definition 90 (Σ_{n-1}-influence message adversary)**.** *The message adversary Σ_{n-1}-MAJ is the set of all sequences of communication graphs $(\mathcal{G}^r)_{r>0}$ that satisfy the following: if each process $p_i \in \Pi$ becomes a single-node source component during a non-empty set of Intervals X_i, then any selection $\{I_1, \ldots, I_n\}$ with $I_i \in X_i$ for $1 \leqslant i \leqslant n$, contains two distinct $I_i = [a,b]$ and $I_j = [a',b']$ such that $\mathtt{s}_i^b \rightsquigarrow \mathtt{s}_j^{a'}$.*

It is easy to devise a set agreement algorithm that works correctly in a dynamic network under Definition 90, provided (a bound on) n is known: In Algorithm 6, process p_i maintains a proposal value v_i, initially x_i, and a decision value y_i, initially $\perp$, which are broadcast in every round. If p_i receives no message from any other process in a round, it decides by setting $y_i = v_i$. If p_i receives a message from some p_j that has already decided ($y_j \neq \perp$), it sets $y_i = y_j$. Otherwise, it updates v_i to the maximum of v_i and all received values v_j. At the end of round n, a process that has not yet decided sets $y_i := v_i$, and all processes terminate.

Theorem 91 (Correctness Algorithm 6)**.** *Algorithm 6 solves $n-1$-set agreement in a dynamic network under message adversary Σ_{n-1}-MAJ given in Definition 90.*

Proof. Termination (after n rounds) and also validity are obvious, so it only remains to show $n-1$-agreement. Assume, w.l.o.g., that the processes $p_1, p_2, \ldots$ are ordered according to their initial values $x_{p_1} \leqslant x_{p_2} \leqslant \ldots$, and let R^k be the set of different values (in y_i or, if still $y_i = \perp$, in v_i) present in the system at the beginning of round $k \geqslant 1$; R^1 is the set of initial values.

Algorithm 6 Set agreement algorithm for message adversary Σ_{n-1}-MAJ.

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

Obviously, $R^1 \supseteq R^2 \supseteq \ldots$, and since $n-1$-agreement is fulfilled if $|R^{n+1}| < n$, we only need to consider the case where all x_i are different.

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

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

Now consider p_n in round n in the above construction of P^n: Definition 90 prohibits the only case where $n-1$-agreement could possibly be violated, namely, when p_n also decides on x_n: During the first n rounds, we would have obtained n single-node source components no two of which influence each other in this case. Thus, we cannot extend the inductive construction of P^i to $i = n$, as the resulting execution would be infeasible. □

A message adversary for general k-set agreement

Whereas the set agreement solution introduced in the previous subsection is simple, it is apparent that Definition 90 is quite demanding. In particular, it requires explicit knowledge of (a bound on) n. We will now provide a message adversary MAJINF(k) (Definition 95), which is sufficient for general k-set agreement if combined with $\mathrm{VSSC}_{D,H}(k, 3D+H)$ (Definition 89) and even with $\mathrm{VSSC}_{D,H}(n, 3D+H)$.

To avoid non-terminating (i.e., forever undecided) executions as predicted results in [22], we require the *stable interval* constraint guaranteed by the message adversary $\mathrm{VSSC}_{D,H}(n, 3D+H)$

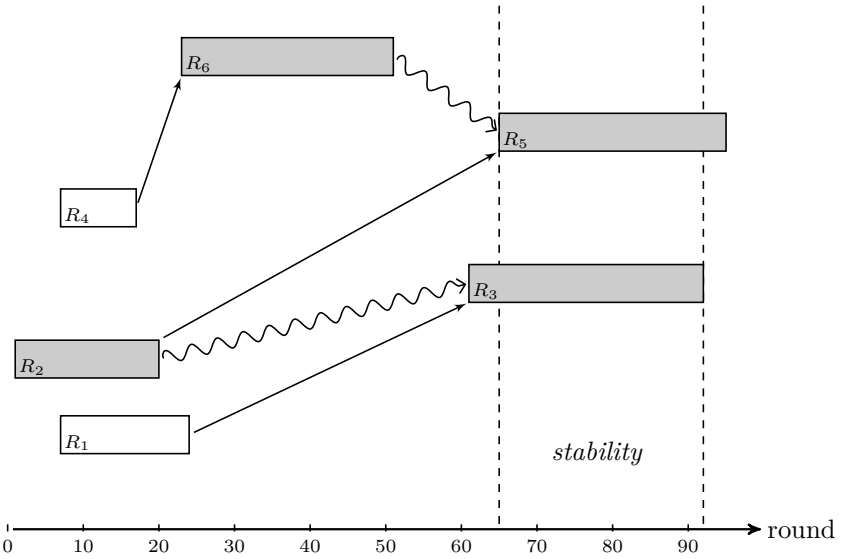


Figure 6.5: VSSCs influencing each other in a run, for $k = 2$. Time progresses from left to right; all gray rectangles are stable for more than $2D$ rounds, white rectangles are stable between $D + 1$ and $2D$ rounds. Snaked arrows represent majority influence, thin arrows represent (weak) influence. At most two gray rectangles may exist that are not majority-influenced by any other gray rectangles.

to hold. The parameter D, which can always be safely set to $D = n - 1$ according to Lemma 46, allows to adapt the message adversary to the actual dynamic source diameter guaranteed in the VSSCs of a given dynamic network. Note that, since $D > 0$, rounds where no message is received are not forbidden here (in contrast to Definition 90).

In order to also circumvent executions violating the k-agreement property established in [22], we introduce the *majority influence* constraint guaranteed by the message adversary MAJINF(k) given in Definition 95 below. Like Definition 90 for set agreement, it guarantees some (minimal) information flow between sufficiently long-lasting vertex-stable source components that exist at different times. As visualized in Figure 6.5, it implies that the information available in any such I-VSSC, with $|I| > 2D$, originates in at most k "initial" J-VSSCs, where $|J| > 2D$. Thereby, it enhances the very limited information propagation that could occur in our model solely under $\mathrm{VSSC}_{D,H}(k, 3D + H)$, which is too strong for solving k-agreement.

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

Definition 92 (Causal Influence Sets)**.** *Let $(S, I) \in \mathbb{V}_1$ with $I = [a, b]$, $(S', I') \in \mathbb{V}_1$ with $I' = [a', b']$, and let $a' > b$. The causal influence set of (S, I) and (S', I') is $CS((S, I), (S', I')) := \left\{ p_j \in S' \mid \exists p_i \in S \colon \mathbf{s}_i^b \rightsquigarrow \mathbf{s}_j^{a'} \right\}$.*

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

Definition 93 (Majority influence)**.** *Let $I = [a, b]$ and $I' = [a', b']$. For $(S, I) \in \mathbb{V}_{2D+1}$ and $(S', I') \in \mathbb{V}_{2D+1}$, we say that (S, I) exercises a* majority influence *on (S', I'), denoted*

$(S,I)\hookrightarrow_{\mathtt{m}}(S',I')$ *with* $\hookrightarrow_{\mathtt{m}} \subseteq \mathbb{V}_{2D+1}^2$*, if and only if for all* $(S'',I'') \in \mathbb{V}_{D+1}$ *with* $CS((S'',I''),(S,I)) = \emptyset$ *it holds that* $|CS((S,I),(S',I'))| > |CS((S'',I''),(S',I'))|$ *and for all* $(S'',I'') \in \mathbb{V}_{D+1}$ *with* $CS((S'',I''),(S,I)) \neq \emptyset$ *it holds that* $|CS((S,I),(S',I'))| \geqslant |CS((S'',I''),(S',I'))|$.

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

Lemma 94 (Properties $\hookrightarrow_{\mathtt{m}}$)**.** *The majority influence relation is antisymmetric, acyclic and intransitive.*

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

With these preparations, we are now ready to specify a message adversary MAJINF(k) given in Definition 95.

Definition 95 (k-majority influence message adversary)**.** *The message adversary MAJINF*(k) *is the set of all sequences of communication graphs* $(\mathcal{G}^r)_{r>0}$*, where in every run there is a set* $K \subseteq \mathbb{V}_{2D+1}$ *with* $|K| \leqslant k$ *s.t.* $\forall (S,I) \in \mathbb{V}_{2D+1} \setminus K\ \exists (S',I') \in \mathbb{V}_{2D+1}$ *with* $(S',I')\hookrightarrow_{\mathtt{m}}(S,I)$.

Informally, Definition 95 ensures that all but at most k "initial" I-VSSCs with $|I| \geqslant 2D+1$ are majority-influenced by some earlier I'-VSSCs with $|I'| \geqslant 2D+1$ (see Figure 6.5). Note carefully, though, that Definition 95 does not prohibit partitioning of the system in more than k simultaneous VSSCs.

We conclude this section with some straightforward stronger assumptions, which also imply Definition 95 and can hence be handled by the algorithm introduced in Section 6.6.1:

(i) Replacing majority influence in Definition 93 by majority intersection $|S \cap S'| > |S'' \cap S'|$, which is obviously the strongest form of influence.
(ii) Requiring $|S \cap S'| > |S'|/2$, i.e., a majority intersection with respect to the number of processes in S'. This could be interpreted as a changing VSSC, in the sense of "S' is the result of changing a minority of processes in S". Although this restricts the rate of growth of VSSCs in a run, it would apply, for example, in case of random graphs where the giant component has formed [49, 66].

Gracefully degrading consensus/k-set agreement

In this section, we provide a k-set agreement algorithm and prove that it works correctly under the message adversary $\mathrm{VSSC}_{D,H}(n, 3D+H) + \mathrm{MAJINF}(k)$, i.e., the conjunction of Definition 89 and Definition 95. Note that the algorithm needs to know D, but neither n nor H. It consists of a "generic" k-set agreement algorithm, which relies on the network approximation algorithm of Section 6.3.1 for locally detecting vertex-stable source components and a function `GetLock` that extracts candidate decision values from history information. Our implementation of `GetLock` uses

a vector-clock-like mechanism for maintaining "causally consistent" history information, which can be guaranteed to lead to proper candidate values thanks to $\text{VSSC}_{D,H}(n, 3D + H) + \text{MAJINF}(k)$.

In sharp contrast to classic k-set agreement algorithms, the algorithm is *k-universal*, i.e., the parameter k does not appear in its code. Rather, the number of system-wide decision values is determined by the number of (certain) $2D + 1$-VSSCs occurring in the particular run. As a consequence, if the network partitions into k weakly connected components, for example,[5] all processes in a component obtain the same decision value. On the other hand, if the network remains well-connected, the algorithm guarantees a unique decision value system-wide.

Properties. Our algorithm is in fact not only k-universal but even worst-case k-optimal, in the sense that (i) it provides at most k decisions system-wide in all runs that are feasible for $\text{VSSC}_{D,H}(n, 3D + H) + \text{MAJINF}(k)$, and (ii) that there is at least one feasible run under $\text{VSSC}_{D,H}(n, 3D + H) + \text{MAJINF}(k)$ where no correct k-set agreement can guarantee less than k decisions. (i) will be proved in Section 6.6.1, and (ii) follows immediately from the fact that a run consisting of k isolated partitions is also feasible for $\text{VSSC}_{D,H}(n, 3D + H) + \text{MAJINF}(k)$. Our algorithm can hence indeed be viewed as a consensus algorithm that degrades gracefully to k-set agreement, for some k determined by the actual network properties.

Network approximation. Like the consensus algorithm in Section 6.3.1, our k-set agreement algorithm consists of two reasonably independent parts, the network approximation algorithm Algorithm 4 and the k-set agreement core algorithm given in Algorithm 7. As in Section 6.3.1, we assume that the complete round r computing step of the network approximation algorithm is executed just before the round r computing step of the k-set algorithm, and that the round r message of the former is piggybacked on the round r message of the latter. Recall that this implies that the round r computing step of the k-set core algorithm, which terminates round r, can already access the result of the round r computation of the network approximation algorithm, i.e., its state at the end of round r.

Core algorithm. The general idea of our core k-set agreement algorithm in Algorithm 7 is to generate new decision values only at members of $2D + 1$-VSSCs, and to disseminate those values throughout the remaining network. Using the network approximation A_i, our algorithm causes process p_i to make a transition from the initially *undecided* state to a *locked* state when it detects some minimal "stability of its surroundings", namely, its membership in some $D + 1$-VSSC D rounds in the past (Line 19). Note that the latency of D rounds is inevitable here, since information propagation within a $D + 1$-VSSC may take up to D rounds since it is D-bounded, as guaranteed by item (ii) in Definition 89. If process p_i, while in the locked state, observes some period of stability that is sufficient for locally inferring a consistent view among *all* VSSC members (which occurs when the $D + 1$-VSSC has actually extended to a $2D + 1$-VSSC), p_i can safely make a transition to the *decided* state (Line 26). The decision value is then broadcast in all subsequent rounds, and adopted by any not-yet decided process in the system that receives it later on (Line 9). Note that $\text{VSSC}_{D,H}(n, 3D + H)$ (Definition 89) guarantees that this will eventually happen.

Since locking is done optimistically, however, it may also happen that the $D + 1$-VSSC does not extend to a $2D + 1$-VSSC (or, even worse, is not recognized to have done so by some members) later on. In this case, p_i makes a transition from the locked state back to the undecided state

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

Algorithm 7 k-universal k-set agreement algorithm, code for process p_i

Variables and Initialization:
1: $\text{hist}_i[*][*] := \emptyset$ /* $\text{hist}_i[j][r]$ holds p_i's estimate of the locks learned by p_j in round r */
2: $\text{hist}_i[i][0] := \{(\{p_i\}, x_i, 0)\}$ /* virtual first lock ($V(S) := \{p_i\}, v := x_i, \tau_{\texttt{create}} := 0$) at p_i */
3: $\ell := \perp$ // most recent lock round, $\perp$ if none
4: $decision_i := \perp$ // p_i's decision, $\perp$ if undecided
Emit round r messages:
5: send $\langle \text{hist}_i, decision_i \rangle$ to all neighbors
Receive round r messages:
6: for all p_j in p_i's neighborhood $\mathcal{N}_i^r$, receive $\langle \text{hist}_j, decision_j \rangle$
Round r computation:
7: **if** $decision_i = \perp$ **then**
8: **if** received any message m containing $m.decision \neq \perp$ **then**
9: decide $m.decision$ and set $decision_i := m.decision$
10: **else**
// update hist_i with hist_j received from neighbors
11: **for** $p_j \in \mathcal{N}_i^r$, where p_j sent hist_j **do**
12: $\text{hist}_i' := \text{hist}_i$ // remember current history
13: **for** all non-empty entries $\text{hist}_j[x][r']$ of hist_j, $x \neq i$ **do**
14: $\text{hist}_i[x][r'] := \text{hist}_i[x][r'] \cup \text{hist}_j[x][r']$
15: **end for**
// locally add all newly learned locks:
16: $\text{hist}_i[i] := \text{hist}_i \setminus \text{hist}_i'$
17: **end for**
// perform state transitions (undecided, locked, decided):
18: mySource := $\texttt{InStableSource}([r-2D, r-D])$
19: **if** $\ell = \perp$ and mySource $\neq \emptyset$ **then**
20: $\ell := r - 2D$
21: lock := $\texttt{GetLock}(\text{mySource}, \ell)$
22: $\text{hist}_i[i][r] := \text{hist}_i[i][r] \cup$ lock // create new lock
23: **else if** $\ell \neq \perp$ and mySource $= \emptyset$ **then**
24: $\ell := \perp$ // release unsuccessful lock
25: **else if** $\ell \neq \perp$ and $\texttt{InStableSource}([\ell, \ell + 2D]) \neq \emptyset$ **then**
26: decide lock.v and set $decision_i :=$ lock.v
27: **end if**
28: **end if**
29: **end if**

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

(Line 24). Unfortunately, this possibility has severe consequences: Mechanisms are required that, despite possibly inconsistently perceived unsuccessful locks, ensure both (a) an *identical* decision value among all members of a $2D+1$-VSSC who successfully detect this $2D+1$-VSSC and thus reach the decided state, and (b) no more than k different decision values originating from different $2D+1$-VSSCs.

Both goals are accomplished by a particular selection of the decision values (using function `GetLock`), which ultimately relies on an intricate utilization the network properties guaranteed by our message adversary $\text{VSSC}_{D,H}(n, 3D+H) + \text{MAJINF}(k)$ (Definition 89 and Definition 95): Our algorithm uses a suitable *lock history* data structure for this purpose, which is continuously exchanged and updated among all reachable processes. It is used to store sets of *locks* $L = (S, v, \tau_{\texttt{create}})$, which are created by every process that enters the locked state: S is the vertex-set of the detected $D+1$-VSSC, v is a certain proposal value (determined as explained below), and $\tau_{\texttt{create}}$ is the round when the lock is created.

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

(i) *Local lock creation:* Apart from the single *virtual* lock $(\{p_i\}, x_i, 0)$ created initially by p_i in Line 2 (which guarantees a non-empty lock history right from the beginning), all regular locks created upon p_i's transition from the undecided to the locked state are computed by the function `GetLock` in Line 21. Any lock locally created at p_i in round r (that is, in the round r computing step of the core k-set agreement algorithm that terminates round r) is of course put into $\text{hist}_i[i][r]$.

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

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

Note that the update rules (i)+(ii) resemble the ones of vector clocks [78].

Clearly, $\text{hist}_i[i][r']$ will always be accurate for current and past rounds $r' \leqslant r$, while $\text{hist}_i[j][r']$ may not always be up-to date, i.e., may lack some locks that are present in $\text{hist}_j[j][r']$. Nevertheless, if p_i and p_j are members of the same I-VSSC S with $I = [r-2D, r]$, Definition 49 ensures that p_i and p_j have consistent histories $\text{hist}_i[j][r']$ and $\text{hist}_j[i][r']$ at latest by (the end of) round $r'+D$, for any $r' \in [r-2D, r-D]$. Hence, if p_i creates a new lock L when it detects, in its round r computing step, that it was part of a $D+1$-VSSC that was stable from $r-2D$ to $r-D$, it is ascertained that any other member p_j will have locally learned the same lock L in the same round r, provided that the $D+1$-VSSC in fact extended to a $2D+1$-VSSC.

Consistent decisions. The resulting consistency of the histories is finally exploited by the function $\texttt{GetLock}(S, \ell)$, which computes (the value of) a new local lock (Line 21) created in round r. As its input parameters, it is provided with the members S of the detected $D+1$-VSSC and its starting round $\ell = r-2D$. `GetLock` first determines a multiset R, which contains all locks locally known to the members $p_j \in S$ by round $r-2D$ (Line 30). Note that the multiplicity of some lock $L = (S', v, r')$ in R is just the number of members of S who got to know L by round $r-2D$, which is just $|\text{CS}(S', S)|$ according to Definition 92. In order to determine a proper value for the new lock to be computed by `GetLock`, we exploit the fact that MAJINF(k) (given in Definition 95) ensures majority influence according to Definition 93: If the set $\text{mfrq}_{\texttt{latest}}(R)$, containing the most frequent locks in R with the same maximal lock creation round, contains a single lock L only, its value $L.v$ is used. Note that the restriction to the maximal lock creation date automatically filters unwanted, outdated locks that have merely been disseminated in preceding $2D+1$-VSSCs, see (1) below. Otherwise, i.e., if $\text{mfrq}_{\texttt{latest}}(R)$ contains multiple candidate locks, a consistent deterministic choice, namely, the maximum among all lock values in R, is used (Line 36). As a consequence, at most k different decision values will be generated system-wide.

Given the various mechanisms employed in our algorithm and their complex interplay, the question about a more light-weight alternative solution that omits some of these mechanisms might arise. We will proceed with some informal arguments that support the necessity of some of the pillars of our solution, namely, (1) the preference of most recently created locks in `GetLock`, (2) the creation of a new lock at every transition to the locked state, and finally (3) the usage of an a priori unbounded data structure hist_i. Although these arguments are also "embedded" in

the correctness proof in the following section, they do not immediately leap to the eye and are hence provided explicitly here.

(1) The preference of most recently created lock in `GetLock`, which is done by selecting the set $\text{mfrq}_{\texttt{latest}}(R)$ in Line 32, defeats the inevitable "amplification" of the number of processes that got to know some "old" lock: All members of a $2D+1$-VSSC have finally learned *all* "old" locks that were only known to *some* of its members at the starting round of the VSSC initially. In terms of multiplicity in R, this would falsely make any such old lock a preferable alternative to the most recently created lock.

(2) Instead of creating new locks at every newly detected $D+1$-VSSC, it might seem sufficient to simply update the creation time of an old lock that (dominantly) influences a newly detected VSSC. This is not the case, however: Consider a hypothesized algorithm where new locks are only generated if no suitable old locks can be found in the current history, and assume a run where two VSSCs with vertex sets $S_1 = \{p_1, p_2\}$ and $S_3 = \{p_1, p_2\}$ that are both stable for $D+1$ rounds and two source components $S_2 = \{p_1, p_3\}$ and $S_4 = \{p_1, p_3\}$ that are stable for $2D+1$ rounds are formed. Let these VSSCs be such that S_i is formed before S_j if $i < j$ and let there be no influence among the processes of $\{p_1, p_2, p_3\}$, apart from their influence on each other when they are members of the same VSSC. First, let the processes of S_1 lock on some old lock L'. Then, assume that the processes of S_2 lock on some lock[6] $L \neq L'$, a lock not known in S_1. Since $S_3 = \{p_1, p_2\}$, if S_3 is sufficiently well connected, p_1 might lock on L' in S_3, because L' is known to both p_1 and p_2 while L is known merely to p_1 at the start of S_3. Subsequently, this results in the situation in S_4 where there is neither a clear majority (L' and L are known to both members of S_4) nor a clear most recently adopted lock (for p_1, it seems that L' is the most recent lock, while for p_3, it seems that L is more recent). Consequently, in S_4, it is not clear whether to lock on $L.v$ or on $L'.v$. Nevertheless, the processes of S_4 should be able to determine that they must lock on L and not on L', since $S_2 \hookrightarrow_{\mathtt{m}} S_4$ holds in our example: $|\text{CS}(S_1, S_2)| = 1, |\text{CS}(S_1, S_4)| = 2, |\text{CS}(S_2, S_4)| = 2$ and $|\text{CS}(S_3, S_4)| = 1$. We can therefore conclude that merely adopting old locks is insufficient.

(3) Since the stabilization round r_{ST}, as implied by Definition 89, may be delayed arbitrarily, an unbounded number of $2D+1$-VSSCs can occur before r_{ST}. Since any of those might produce a critical lock, in the sense of exercising a majority influence upon some later $2D+1$-VSSC, no such lock can safely be deleted from hist_i of any p_i after bounded time.

Correctness Proof

In this final subsection, we will prove the following Theorem 96:

Theorem 96. *Algorithm 7 solves k-universal k-set agreement in a dynamic network under the message adversary $\text{VSSC}_{D,H}(n, 3D+H)+\text{MAJINF}(k)$, which is the conjunction of Definition 89 and Definition 95.*

The proof consists of a sequence of technical lemmas, which will allow us to establish all the properties of k-set agreement. First, validity according to Definition 7 is straightforward to see, as only the values of locks are ever considered as decisions (Line 26). Values of locks, on the other hand, are initialized to the initial value of a process (Line 2) and later on always have values of previous locks assigned to them (Line 34 and Line 36). Note that the claimed k-universality is obvious, as the code of the algorithm does not involve k.

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

To establish termination, we start with Lemma 97, Lemma 98 and Lemma 99 that are related to setting locks at all members of vertex stable source components.

Lemma 97. *Apart from processes adopting a decision sent by another process, only processes part of a vertex stable source component with interval length greater than D (resp. $2D$) lock (resp. decide).*

Proof. The if-statement in Line 19 (resp. Line 25) is evaluated to true only if `InStableSource` detects a stable member set S in some interval I of length $D+1$ (resp. of length $2D+1$) or larger, which implies by Corollary 60 that S is indeed a I-VSSC with $|I| = D+1$ (resp. $|I| = 2D+1$). □

Lemma 98. *All processes part of a I-VSSC S with $I = [a, b]$ and $|I| > 2D$, which did not start already before a, lock, i.e. set $\ell := a$, in round $a + 2D$.*

Proof. Because S is D-bounded by Definition 89, Corollary 62 guarantees that `InStableSource`$(a, a+D)$ returns S from round $a + 2D$ (of the k-set-algorithm) on, and that it cannot have done so already in round $a + 2D - 1$. Hence, $\ell = \perp$ in round $a + 2D$, the if-statement in Line 19 is entered and $\ell := a$ is set in Line 21. □

Lemma 99. *All processes part of a I-VSSC S with $I = [a, b]$ and $|I| > 3D$, which did not start already before a, have decided by round $a + 3D$.*

Proof. It follows from Lemma 98 that all members of the VSSC S set $\ell := a$ in round $a + 2D$. As the VSSC remains stable also in rounds $a + 2D, \ldots, a + 3D$, Line 24 will not be executed in these rounds, thus $\ell = a$ remains unchanged. Consequently, due to Corollary 62, the if-statement in Line 25 will evaluate to true at the latest in round $\ell + 3D = a + 3D$, causing all the processes to decide via Line 26 by round $a + 3D$ as asserted. □

Lemma 100. *The algorithm eventually terminates at all processes.*

Proof. Pick any process p_j. If p_j is part of a source component during the stable interval, guaranteed by Definition 89, Lemma 99 ensures termination by $r_{ST} + 3D$ at the latest. Thus, we assume p_j is not part of a source component during the stable interval. From Definition 52, it follows that there exists a causal chain of length at most H to p_j from some member p_i of a VSSC after its termination. Therefore, it must receive the decide message and decide via Line 9 by $r_{ST} + 3D + H$ at latest. □

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

Lemma 101. *Given some I-VSSC S with $I = [a, b]$ and $b \geqslant a + D$, in all rounds $x \in [a + D, b]$ it holds that $\forall p_i, p_j \in S\colon \bigcup_{r' \leqslant a} \mathit{hist}_i[j][r'] = \bigcup_{r' \leqslant a} \mathit{hist}_j[j][r']$*

Proof. Because S is D-bounded, a message from round a has reached every member of S by round $a + D$. Moreover, no message sent by a process not in S during I can reach a member of S during I because S is a source component. Therefore, since hist_i is sent by each process p_i in every round (Line 5) and p_i adds only newly learned entries to hist_i (Line 22 and Line 16), all these updates of hist_i during I, regarding any round $r' \leqslant a$, occur at the latest in round $a + D$. □

Lemma 102. *All processes of a I-VSSC S of $\mathbb{V}_{2D+1}$ with $I = [a, b]$ adopt the same lock (and hence decide the same).*

Proof. Such a lock is created by $p_i \in S$ in round $a + 2D$, when it recognizes S as having been vertex-stable for $D + 1$ rounds according to Lemma 98. As the lock (value) is computed based on hist_i present in round $a + 2D$, which is consistent among all VSSC members by Lemma 101, the lemma follows. □

Finally, we show that, given that the system satisfies Definition 95, there will be at most k decision values in any run of Algorithm 7, which proves k-agreement: Since there are at most k VSSCs of $\mathbb{V}_{2D+1}$ that are not majority-influenced by other VSSCs, it remains to show that any majority-influenced VSSC decides the same as the VSSC it is majority-influenced by. In order to do so, we will first establish a key property of our central data structure hist_i.

Lemma 103. *Given I-VSSC S with, $I = [a, b]$, and I'-VSSC R', $I' = [a', b']$, where $|I| > 2D$ and $|I'| \geqslant 1$, let L be a lock known to all members of S by b, i.e., for all $p_i \in S$ it holds that, by the end of round b, $L \in \bigcup_{r' \leqslant b} \text{hist}_i[i][r']$. For any process $p_j \in S'$, it holds that if there exists some $p_i \in S$, s.t. $\mathtt{s}_i^b \rightsquigarrow \mathtt{s}_j^{a'}$, then $L \in \bigcup_{r' \leqslant a'} \text{hist}_j[j][r']$.*

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

Lemma 104. *Given I-VSSC $S \in \mathbb{V}_{2D+1}$, $I = [a, b]$, and I'-VSSC $S' \in \mathbb{V}_{2D+1}$, $I' = [a', b']$, assume that the processes of S created the (same) lock L when locking. If $(S, I) \hookrightarrow_{\mathtt{m}} (S', I')$, then the processes in S' will choose a lock L' where $L.v = L'.v$ (and hence decide the same as the processes in S).*

Proof. From the definition of $\hookrightarrow_{\mathtt{m}}$ (Definition 93), it follows that no I''-VSSC of $\mathbb{V}_{D+1}$ has a larger influence set on S' than S. By Lemma 97, this implies that no lock that was generated by some I''-VSSC in $\mathbb{V}_{D+1}$ can be known to more members of S' than the lock L generated by S. Since process p_i puts only newly learned locks into hist_i (Line 22 and Line 16), by Lemma 103, this means that in round a' no "bad" lock L_b is present in more elements of $R = \bigcup_{p_i \in R_{I'}, r' \leqslant a'} \text{hist}_i[i][r']$ than L. We now show that $L.\tau_{\mathtt{create}} > L_b.\tau_{\mathtt{create}}$ for all L_b occurring in as many elements of R as L with $L_b \neq L$. Obviously, the only locks L_b that could occur in as many elements of R as L are locks that have been in hist_i of some $p_i \in S$ at the beginning of round a already. Since for any such L_b, L was created after L_b, by Line 34 and Line 36, we have that $L.\tau_{\mathtt{create}} > L_b.\tau_{\mathtt{create}}$, as claimed. Because in round $a' + 2D$, at all processes p_i, p_j of S', Lemma 101 implies that $\bigcup_{r' \leqslant a'} \text{hist}_i[j][r'] = \bigcup_{r' \leqslant a'} \text{hist}_j[i][r']$, when locking in round $a' + 2D$ according to Lemma 98, every p_i of S' will find L as the unique most common lock in the elements of R with maximal $\tau_{\mathtt{create}}$. This leads to the evaluation of the if-statement in Line 32 to true and to the creation of a new lock L', where $L'.v = L.v$ in Line 34, as asserted. □

This completes the proof of Theorem 96.

CHAPTER 7

Model reductions

In this section, we introduce ways to relate different models to each other in terms of problem solvability power. Algorithmic reductions allow to employ already existing algorithms or impossibility results in models where previously no such results were known. Especially successful are message adversary simulations to related different message adversaries to each other and to provide solutions algorithm for certain problems. These results were published in [22, 101].

7.1 Failure detector reductions

A convenient way to characterize consensus and k-set solvability in distributed systems where processes are (usually) subject to crash failures are *failure detectors* [36]. Well-known results for message passing systems where a majority of processes may crash are the weakest failure detector (Σ, Ω) (defined below) for consensus [47], and the necessary failure detector Σ_k ($\Sigma = \Sigma_1$) for k-set agreement [25]. Note that, whereas the weakest failure detector for k-set agreement in message passing systems is still unknown, there are failure detectors like $\mathcal{L}_k$ [20] that are sufficiently strong for this purpose. These results imply that, from any solution that solves consensus resp. k-set agreement, it must be possible to implement Σ and Ω resp. Σ_k. Conversely, if Σ and Ω can be implemented in some system, then there are well-known algorithms for solving consensus in this system.

In this section based on [22], we follow the example of [94] and explore the relation between our message adversaries and the above failure detectors. It is important to note, though, that both $\text{VSSC}_{D,E}(d)$ and $\text{VSSC}_{D,H}(n, d) + \text{MAJINF}(k)$ are inherently incompatible with time-free failure detectors, as they involve explicit timing information, namely, the duration of the stability window. By contrast, the specifications of Ω, Σ and Σ_k are time-free, in the sense that they only involve eventual properties for liveness. Therefore, we will consider only the eventually-forever variants $\text{VSSC}_{D,E}(\infty)$ and $\text{VSSC}_{D,H}(n, \infty) + \text{MAJINF}(k)$ of our message adversaries in the comparison below.

7.1.1 Failure detector basics

We recall that a crash failure means that a faulty process may stop to perform any computation step after some point during an execution, possibly in a way that causes only a subset of the

processes to receive the message of the last broadcast. Given the time domain $\mathcal{T}$ of some system where processes are prone to crashes, for some given run, the function $F : \mathcal{T} \to 2^{\Pi}$ that maps each $t \in \mathcal{T}$ to the processes that are crashed by t is called the failure pattern of the run. Processes in the set $\mathcal{C} = \Pi \setminus \bigcup_{t \in \mathcal{T}} F(t)$ are called *correct*.

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

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

Two important failure detectors for consensus are Σ and Ω.

Definition 105. *The eventual leader failure detector Ω has range Π. For each failure pattern F, for every history $H \in \Omega(F)$, there is a time $t \in \mathcal{T}$ and a correct process p_j s.t. for every process p_i for every $t' \geqslant t$, $H(p_i, t') = p_j$.*

Definition 106. *The quorum failure detector Σ has range 2^{Π}. For each failure pattern F, for every $H \in \Sigma(F)$, two properties hold: (1) for every $t, t' \in \mathcal{T}$ and $p_i, p_j \in \Pi$ we have $H(p_i, t) \cap H(p_j, t') \neq \emptyset$ and (2) there is a time $t \in \mathcal{T}$ s.t. for every process p_i, for every $t' \geqslant t$, $H(p_i, t') \subseteq \Pi \setminus \bigcup_{t \in \mathcal{T}} F(t)$.*

We denote by $\mathcal{AMP}_{n,n-1}[fd : \mathcal{D}]$ the $\mathcal{AMP}_{n,n-1}$ model where processes have access to failure detector $\mathcal{D}$. We use $\mathcal{AMP}_{n,n-1} = \mathcal{AMP}_{n,n-1}[fd : \emptyset]$ to denote the absence of any failure detector. The combination (Σ, Ω) of Σ and Ω is of particular importance, because it is the weakest failure detector for consensus [47] in $\mathcal{AMP}_{n,n-1}$, in the following sense: First, there is an algorithm that uses (Σ, Ω) for solving consensus. Second, let $\mathcal{D}$ be a failure detector s.t. consensus is solvable under $\mathcal{AMP}_{n,n-1}[fd : \mathcal{D}]$. Then, there is an algorithm $\mathcal{A}$ for $\mathcal{AMP}_{n,n-1}[fd : \mathcal{D}]$ that, for each failure pattern F, produces an output, denoted $out(p_i, t)$ for process p_i at time t, s.t. setting $H(p_i, t) := out(p_i, t)$ defines a valid history H of (Σ, Ω).

In order to relate such failure detector models to our message adversaries, which model dynamic link failures, we use the simple observation that the externally visible effect of a process crash can be expressed in our setting: Since correct processes in asynchronous message passing systems perform an infinite number of steps, we can assume that they send an infinite number of (possibly empty) messages that are eventually received by all correct processes. As in [94], we hence assume that the correct (= non-crashing) processes in the simulated $\mathcal{AMP}$ are the *strongly correct processes*. Informally, a strongly correct process is able to disseminate its state to all other processes infinitely often.

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

Definition 107. *Given an infinite sequence of communication graphs σ, process p_i is* faulty *in a run with σ if there is a round r s.t., for some process p_j, for all $r' > r$: $\mathtt{s}_i^r \not\rightsquigarrow \mathtt{s}_j^{r'}$.*
Let $\mathcal{C}(\sigma) = \left\{ p_i \in \Pi \mid \forall p_j \in \Pi, \forall r \in \mathbb{N}, \exists r' > r \colon \mathtt{s}_i^r \rightsquigarrow \mathtt{s}_j^{r'} \right\}$ denote the strongly correct (= non-faulty) processes in any run with σ.

If a given process influences just one strongly correct process infinitely often, it would transitively influence all processes in the system, hence would also be strongly correct. Therefore, in order

not to be strongly correct, a faulty process must not influence *any* strongly correct process infinitely often. We can hence define failure patterns as follows:

Definition 108 (Failure Pattern)**.** *The failure pattern associated with communication graph sequence σ is a function $F_\sigma : \mathbb{N} \to 2^\Pi$ s.t. $p_i \in F_\sigma(r)$ if, and only if, for all processes $p_j \in \mathcal{C}(\sigma)$, for all $r' > r$: $\mathtt{s}_i^r \not\rightsquigarrow \mathtt{s}_j^{r'}$.*

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

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

Definition 109. *For $x < n$, we define $CRASH(x)$ as the set of those communication graph sequences σ where $|\mathcal{C}(\sigma)| \geqslant n - x$.*

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

Corollary 110. *For $x < n/2$, any run with graph sequence σ of $\mathcal{SMP}_n[adv : CRASH(x)]$ can be transformed to a run in $\mathcal{AMP}_{n,x}[fd : \emptyset]$, which is indistinguishable for all simulated processes.*

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

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

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

7.1.2 Relation to consensus

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

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

Definition 111. *$\text{VSSC-PART}_{D,E}(\infty)$ contains those graph sequences where, for some round r_{ST}, there is D-bounded, E-influencing I-VSSC with $I = [r_{ST}, \infty)$*

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

Lemma 112. *Consensus is impossible under the message adversary $\text{VSSC-PART}_{D,E}(\infty)$.*

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

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

Lemma 113. *$\mathcal{SMP}_n[adv : \text{VSSC-PART}_{D,E}(\infty)]$ allows to implement $\mathcal{AMP}_{n,n-1}[fd : \Omega]$.*

Proof. Consider an algorithm that outputs, at process p_i, the process with the largest identifier in the source component that was detected E rounds ago, or itself if no such source component was detected. Clearly, this output is in the range of Ω. Furthermore, since $\text{VSSC-PART}_{D,E}(\infty)$ guarantees that eventually some D-bounded, E-influencing source component S remains the only VSSC forever, S will be eventually detected by every process p_i forever, and its member with the largest identifier will be written to the output of p_i eventually forever as well. By Definition 107, no processes of S is faulty, hence the specification of Ω is satisfied.

To simulate $\mathcal{AMP}$ with process crashes, exactly the same simulation as in [94, Sec.4.2] is used: Analogous to the simulation used in the proof of Corollary 110, a simulated process is only allowed to take its next step if all the messages sent in the previous step are already known by the simulator of the current output of Ω, which (eventually) will be a strongly correct process. □

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

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

Lemma 114. *$\mathcal{SMP}_n[adv : VSSC_{D,E}(\infty)]$ does not allow to implement $\mathcal{AMP}_{n,n-1}[fd : \Sigma]$.*

Proof. Again, we will prove our lemma for $n = 2$ for simplicity, as it is straightforward to generalize the proof for arbitrary n. Suppose that, for all rounds r and any processes p_i, some algorithm $\mathcal{A}$ computes $out(p_i, r)$ s.t. for any admissible failure pattern F, $out \in \Sigma(F)$. Consider the graph sequence $\sigma = (p_i \to p_j)_{r \geqslant 1}$. Clearly, the failure pattern associated with σ is $F_\sigma(r) = \{p_j\}$. Hence, in the run ε starting from some initial configuration C^0 with sequence σ, there is some round r' s.t. $out(p_i, r) = \{p_i\}$ for any $r > r'$ by Definition 106. Let $\sigma' = (p_i \to p_j)_{r=1}^{r'}(p_i \leftarrow p_j)_{r>r'}$. By similar arguments as above, in the run ε' that starts from C^0 with sequence σ', there is a round r'' such that $out(p_j, r) = \{p_j\}$ for any $r > r''$. Finally, for $\sigma'' = (p_i \to p_j)_{r=1}^{r'}(p_i \leftarrow p_j)_{r=r'+1}^{r''}(p_i \leftrightarrow p_j)_{r>r''}$, let ε'' denote the run starting from C^0 with graph sequence σ''. Until round r', $\varepsilon'' \sim_{p_i} \varepsilon$, hence, as shown above, $out(p_i, r') = \{p_i\}$ in ε''. Similarly, until round r'', $\varepsilon'' \sim_{p_j} \varepsilon'$ and hence $out(p_j, r'') = \{p_j\}$ in ε''. Clearly, $\sigma, \sigma', \sigma'' \in \mathrm{VSSC}_{D,E}(\infty)$ and $F_{\sigma''}(r) = \{\}$, that is, no process is faulty in σ''. However, in ε'', $out(p_i, r') \cap out(p_j, r'') = \emptyset$, a contradiction to Definition 106. □

The above result may come as a surprise, since the proof of the necessity of Σ_k for k-set agreement (hence the necessity of $\Sigma = \Sigma_1$ for consensus) developed by Raynal et. al. [25] only relies on the availability of a correct k-set agreement algorithm. However, their reduction proof works only in $\mathcal{AMP}_{n,n-1}$, i.e., crash-prone asynchronous message passing systems: It relies crucially on the fact that there is no safety violation (i.e., a decision on a value that eventually leads to a violation of k-agreement) in any prefix of a run. This is not the case in $\mathcal{SMP}_n$, however, as processes may decide after a certain number of rounds also if no message is received. Hence, we cannot reuse their proof in our setting.

Taken together, Lemmas 112, 113, and 114 allow us to conclude the following:

(i) Since $\mathrm{VSSC}_{D,E}(\infty)$ (not to speak of $\mathrm{VSSC}_{D,E}(d)$, which is not compatible with failure detector specifications) does not allow to implement (Σ, Ω), we cannot derive consensus algorithms from (Σ, Ω)-based solutions. And indeed, our consensus algorithms are algorithmically very different.

(ii) The message adversaries SOURCE and QUORUM considered in [94], which allow to implement (Σ, Ω), are equivalent to $\mathrm{VSSC}_{D,E}(\infty)$ in terms of consensus solvability, but strictly weaker in terms of sequence inclusion, i.e., $(\mathrm{SOURCE}, \mathrm{QUORUM}) \subset \mathrm{VSSC}_{D,E}(\infty)$.

7.1.3 Relation to k-set agreement

We start with the definitions of generalized failure detectors for the k-set agreement setting in crash-prone asynchronous message passing systems, using the notation introduced in Section 7.1.1.

Definition 115. *The range of the failure detector Ω_k is all k-subsets of 2^Π. For each failure pattern F, for every history $H \in \Omega_k(F)$, there $\exists LD = \{q_1, \ldots, q_k\} \in 2^\Pi$ and $t \in \mathcal{T}$ such that $LD \cap \mathcal{C} \neq \emptyset$ and for all $t' \geqslant t, p_i \in \mathcal{C} : H(p_i, t') = LD$.*

Definition 116. *The failure detector Σ_k has range 2^Π. For each failure pattern F, for every $H \in \Sigma_k(F)$, two properties must hold: (1) for every $t, t' \in \mathcal{T}$ and $S \in \Pi$ with $|S| = k + 1$, $\exists p_i, p_j \in S : H(p_i, t) \cap H(p_j, t') \neq \emptyset$, (2) there is a time $t \in \mathcal{T}$ s.t. for every process p_i, for every $t' \geqslant t$: $H(p_i, t') \subseteq \mathcal{C}$.*

k-set agreement in our lock-step round model with link failures allows non-temporary partitioning, which in turn makes it impossible to use the definition of crashed and correct processes from the previous section: In a partitioned system, every process p_i has at least one process p_j such that $\forall r' > r : \mathtt{s}_i^r \rightsquigarrow \mathtt{s}_j^{r'}$, but no p_i usually reaches all $p_j \in \Pi$ here. Definition 107 hence implies that there is no correct process in this setting. Hence, we employ the following generalized definition:

Definition 117. *Given a infinite graph sequence σ, let a* minimal source set *S in σ be a set of processes with the property that $\forall p_j \in \Pi, \forall r > 0$ there exists $p_i \in S, r' > r$ such that $\mathtt{s}_i^r \rightsquigarrow \mathtt{s}_j^{r'}$. The set of* weakly correct *processes $\mathcal{WC}(\sigma)$ of a sequence σ is the union of all minimal source sets S in σ.*

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

Lemma 118. *For every $\sigma \in VSSC_{D,H}(n, \infty) + MAJINF(k)$, it holds that $\mathcal{WC}(\sigma) \neq \emptyset$.*

Proof. By Definition 89, for any $\sigma \in \mathrm{VSSC}_{D,H}(n, \infty)$, there is some non-empty, H-influencing set of D-bounded VSSCs $S_1, \dots S_\ell$ from some round onward in σ. According to Definition 117, $\bigcup_{i=1}^{\ell} S_i \subseteq \mathcal{WC}$. □

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

Lemma 119. *Σ_k cannot be implemented under $VSSC_{D,H}(n, \infty) + MAJINF(k)$.*

Proof. For $k = 1$, we can rely on Lemma 114, as every $\sigma \in \mathrm{VSSC}_{D,E}(\infty)$ is also admissible in $\mathrm{VSSC}_{D,H}(n, \infty) + MAJINF(1)$. Hence, $\Sigma_1 = \Sigma$ cannot be implemented in $\mathrm{VSSC}_{D,H}(n, \infty) + MAJINF(1)$.

The impossibility can be expanded to $k > 1$ by choosing some σ that (i) perpetually partitions the system into k components $\widetilde{P} = \{P_1, \dots, P_k\}$ that each have a single source component and consist of the same processes throughout the run, and (ii) demands eventually a vertex stable source component in every partition forever. Pick an arbitrary partition $P \in \widetilde{P}$. If $|P| > 1$, such a sequence does not allow to implement Σ in P (e.g., the message adversary could emulate the graph sequence used in Lemma 114 in P). We hence know that $\exists p, p' \in P$ and $\exists r, r'$ such that $out(p, r) \cap out(p', r') = \emptyset$. Furthermore, and irrespective of $|P|$, as for every $p_i \in P$, it is indistinguishable whether any $p_j \in \widetilde{P} \setminus P$ is faulty in σ or not, p_i has to assume that every process $p_j \in \widetilde{P} \setminus P$ is faulty. Hence, for every $p_i \in P$, we must eventually have $out(p_i, r_i) \subseteq P$ for some sufficiently large r_i.

We now construct a set S of $k + 1$ processes that violates Definition 116: fix some $P \in \widetilde{P}$ with $|P| > 1$ and add the two processes $p, p' \in P$, as described above, to S. For every partition $P_j \in \widetilde{P} \setminus P$, add one process p_i from P_j to S. Since there exist r, r' such that $out(p, r) \cap out(p', r') = \emptyset$, and $\forall P_j \in \widetilde{P} \setminus P, \forall p_i \in P_j, \exists r_i \colon out(p_i, r_i) \subseteq P_i$ and, by the construction of S, we have that $\forall p_i, p_j \in S$, $\exists r_i, r_j$ such that $out(p_i, r_i) \cap out(p_j, r_j) = \emptyset$. This set S clearly violates Definition 116, as required. □

As for Ω_k, we note that Lemma 113 reveals also that $\Omega_1 = \Omega$ can be implemented under $\text{VSSC-PART}_{D,E}(\infty)$. By contrast, however, Ω_k it is not implementable under $\mathrm{VSSC}_{D,H}(n, \infty) + MAJINF(k)$ for $k > 1$:

Lemma 120. *For $k > 1$, Ω_k cannot be implemented under $VSSC_{D,H}(n, \infty) + MAJINF(k)$.*

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

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

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

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

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

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

Let $\varepsilon_a, \ldots, \varepsilon_e$ be the runs resulting from the above sequences applied to the same initial configuration. By Definition 117 and Definition 115, LD has to include p_1 in ε_a, p_2 in ε_b, and p_3 in ε_c. By Definition 115, in ε_d, because $\varepsilon_a \sim_{p_1} \varepsilon_d$ and $\varepsilon_c \sim_{p_3} \varepsilon_d$ in all rounds, for some $t > 0$, for all $t' > t$, $out(p_1, t') = \{p_1, p_3\}$. A similar argument shows that in ε_e, for some $t > 0$, for all $t' > t$, $out(p_1, t') = \{p_1, p_2\}$, because $\varepsilon_a \sim_{p_1} \varepsilon_e$ and $\varepsilon_b \sim_{p_2} \varepsilon_e$. The indistinguishability $\varepsilon_d \sim_{p_1} \varepsilon_e$ provides the required contradiction, as for some $t > 0$, for all $t' > t$, $out(p_1, t')$ should be the same in ε_d and ε_e. □

We conclude that the necessary and sufficient conditions for consensus and k-set agreement message adversaries in dynamic networks are not identical to the conditions in failure detectors.

7.2 Message adversary reductions

The lemmas in the previous section showed that $\mathcal{AMP}_{n,n-1}[fd : \Sigma, \Omega]$ cannot be simulated atop of $\mathcal{SMP}_n[adv : MA]$ with some message adversary MA that allows to solve consensus. Even though failure detectors cannot hence be used directly to find a strongest message adversary, the concept of comparing models with different restrictions in terms of their computational power is nevertheless attractive. This idea was already used in [41] to structure communication predicates in the HO model, albeit the "general translations" introduced for this purpose suffered from the fact that one would need to solve repeated consensus. The results below, which remove this constraint can be found in [101].

7.2.1 The heard-of-model

The HO model consists of a non-empty set $\Pi = \{p_1, \ldots, p_n\}$ of n processes with unique ids, and a set of messages M, which includes a null placeholder indicating the empty message. Each process $p \in \Pi$ consists of the following components: a set of states denoted by $states_p$, a subset $init_p$ of initial states, for each positive integer $r \in \mathbb{N}*$, called round number, a message sending function S_p^r mapping states $p \times \Pi$ to a unique (possibly null) message m_p, and a state-transition function T_p^r mapping $states_p$ and partial vectors (indexed by Π) z_p of elements of M to $states_p$.

The collection of the pairs of message sending function and state-transition function of the processes for every round $r > 0$ is called an algorithm on Π.

Computations in the HO model are composed of infinitely many rounds, which are communication-closed layers in the sense that any message sent in a round can be received only at that round. In each round r, process p first applies S_p^r to the current state $\mathtt{s}_p^{r-1} \in states_p$, emits the "messages" to be sent to each process, and then, for a subset $HO(p,r)$ of Π (indicating the processes which p hears of), applies T_p^r to its current state and the partial vector of incoming messages whose support is $HO(p,r)$ to compute $\mathtt{s}_p^r$.

A communication predicate P is defined to be a predicate over heard-of collections, that is a Boolean function over the collections of subsets of Π indexed by $\Pi \times \mathbb{N}*$:

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

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

7.2.2 Message adversary simulation

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

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

(E1) $q \in NewHO_p^{(\rho)}$ iff $\mathtt{s}_q^{r_1-1} \rightsquigarrow \mathtt{s}_p^{r_k}$, i.e., if there exist an integer l in $\{1, ..., k\}$, a chain of $l + 1$ processes $p_0, p_1, ..., p_l$ from $p_0 = q$ to $p_l = p$, and a subsequence of l increasing round numbers $r_1, ..., r_l$ in macro-round ρ such that, for any index $i, 1 \leqslant i \leqslant l$, we have $p_{i-1} \in HO(p_i, r_i)$.

(E2) The collection $NewHO_p^{(\rho)}$ for all $p \in \Pi, \rho > 0$ satisfies M.

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

Implementing the above emulation, i.e., the emulation algorithm A, is trivial: Let $m_{p \to q}^r$ represent the message sent by p to q in round r in $\mathcal{SMP}_n[adv : M']$, and $m_{p \to q}^{(\rho)}$ the message sent in macro-round ρ in the simulated $\mathcal{SMP}_n[adv : M]$. A just piggy-backs $m_{p \to q}^{(\rho)}$ on message $m_{p \to q}^j$, for every $(\rho - 1)k + 1 \leqslant j \leqslant \rho k$, and delivers $m_{p \to q}^{(\rho)}$ in $z_q^{(\rho)}$ in macro-round ρ, along with putting

q into $NewHO_p^{(\rho)}$, when some $m_{p\to q}^j$ has been successfully received. Unfortunately, however, this emulation is too restrictive for our purpose.

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

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

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

Using this type of message adversary simulations, in conjunction with the fact that every communication predicate can be viewed as a message adversary, we will prove in Lemma 122 below that consensus solvability and the ability to simulate the communication predicate SP_UNIF introduced in [41] are equivalent.

Definition 121. *Let SP_UNIF be the communication predicate where for all $p, q, r : HO(p,r) = HO(q,r)$.*

Lemma 122. *The following assertions are equivalent:*

(1) For any set of initial values V, there is an algorithm A that solves consensus in $\mathcal{SMP}_n[adv : M']$.
(2) M' allows to simulate $\mathcal{SMP}_n[adv : SP\_UNIF]$ in the execution of every algorithm $\mathcal{A}$.

Proof. The direction (2) $\to$ (1) follows from the fact that [41] provided a (trivial) algorithm $\mathcal{A}$ that solves multi-valued consensus. We can hence plug-in $\mathcal{A}$ in (2) to obtain a consensus algorithm in $\mathcal{SMP}_n[adv : M']$.

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

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

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

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

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

Definition 123 (Strongest message adversary)**.** *A message adversary M is a strongest message adversary for some problem $\mathcal{P}$, if for every M' for which $\mathcal{P}$ is solvable in $\mathcal{SMP}_n[adv : M']$, there exists an algorithm A that allows to simulate $\mathcal{SMP}_n[adv : M]$ in the execution of every algorithm $\mathcal{A}$ that solves $\mathcal{P}$.*

A property that follows directly from Definition 123 is:

Corollary 124. *If a strongest message adversary for multi-valued consensus allows to solve some problem $\mathcal{P}$, it holds that every message adversary that allows to solve multi-valued consensus also allows to solve $\mathcal{P}$.*

By Lemma 122, SP_UNIF is a strongest message adversary for multi-valued consensus. Even more, as the simulation algorithm A used in the proof of Lemma 122 actually simulates the message adversary $STAR \subset SP\_UNIF$, where $HO(p, r) = HO(q, s)$ for every $r, s \geqslant 0$ and every $p, q \in \Pi$, it reveals that $STAR$ is also a strongest message adversary for multi-valued consensus.

Since every other message adversary that contains $STAR$ is also a strongest message adversary by definition, we finally obtain the following Corollary 125:

Corollary 125 (Class of strongest message adversaries for consensus)**.** *Let $STAR$ be the message adversary that consist of all sequences of all possible perpetual stars. Every message adversary that includes $STAR$ is a strongest message adversary for multi-valued consensus.*

Examples for such message adversaries are (SOURCE, QUORUM), $\text{VSSC}_{D,E}(\infty)$ and SP_UNIF.

Interestingly, the findings above can be easily be adopted for k-set agreement as well: The same simulation algorithm B as used in the proof of Lemma 122 can be used to simulate k perpetual stars atop of a message adversary M' that allows to solve k-set agreement: As any k-set agreement algorithm A guarantees at most k different decision values, B indeed allows to simulate at most k perpetual stars with the k decisions as the centers. Hence:

Corollary 126 (Class of strongest message adversaries for k-set agreement)**.** *Every message adversary that contains all sequences of all possible perpetual k-stars is a strongest message adversary for k-set agreement.*

An example for a strongest message adversary for k-set agreement is the message adversary $\text{VSSC}_{D,H}(n, \infty) + MAJINF(k)$ introduced in [22].

7.3 Consequences of our Results

The results of Section 7.1, in particular, Lemma 114, reveal the following facts:

(i) Since $\text{VSSC}_{D,E}(\infty)$ does not allow to implement Σ, we cannot hope to run Σ, Ω-based consensus algorithms on top if it.
(ii) The message adversary (SOURCE, QUORUM) considered in [94], $\text{VSSC}_{D,E}(\infty)$ and SP_UINF are all incomparable in terms of graph sequence inclusion, even though they all belong to the class of strongest message adversaries.
(ii) There are message adversaries like the one introduced in [102], which (unlike $\text{VSSC}_{D,E}(\infty)$) do not even guarantee a single strongly correct process in some runs. Implementing Σ subject to Definition 106 atop of such message adversaries is trivially impossible, as its specification becomes void.

On the other hand, the results of Section 7.2.2, in particular, Lemma 122, reveals that it is possible to simulate the message adversary SP_UNIF atop of any message adversary (hence also $\text{VSSC}_{D,E}(\infty)$) that allows to solve multi-valued consensus. However, it is trivial to simulate Σ, Ω in $\mathcal{AMP}$ in $\mathcal{SMP}_n[adv : SP\_UNIF]$: Initially, process p outputs p as the leader and Π as the quorum. At the end of round 1, both the leader and the quorum is set to $NewHO(p, r)$. Therefore, we seem to have arrived at a contradiction of Lemma 114!

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

Hence, it finally turns out that the impossibility of implementing Σ established in Lemma 114 depends crucially on the assumption to consider strongly correct processes as correct in the simulated $\mathcal{AMP}$. In principle, it might be possible to implement Σ (and also Ω) atop of any message adversary that allows to solve consensus if a weaker alternative of Definition 107 of correct processes in $\mathcal{AMP}$ was used: For ℓ, it would essentially be sufficient if it managed to disseminate its initial state to all processes in the system once. Quite obviously, though, such a definition of a correct process would severely affect the semantics of failure detectors and hence the wealth of known results.

In addition, the principal ability to simulate Σ, Ω atop of the simulated system $\mathcal{SMP}_n[adv : SP\_UNIF]$ is not very useful in practice, as it hinges on the availability of a consensus algorithm for the bottom-level message adversary M'. Consequently, this possibility does not open up a viable alternative to the development of consensus algorithms tailored to specific message adversaries like the ones introduced in [102, 108].

Overall, it turns out that strongest message adversaries according to Definition 123 do not have much discriminating power, as essentially all message adversaries known to us that allow to solve consensus are strongest according to Corollary 125. Finding a better definition of a strongest message adversary is a topic of future research. Note, however, that naive ideas like one that (i) admits a solution algorithm and (ii) is maximal w.r.t. its set of admissible graph sequences it may generate do not easily work out: Given that the latter set is usually uncountable, as admissible graph sequences are infinite, it is not clear whether (ii) is well-defined in general.

CHAPTER 8

Conclusion and outlook

In this thesis we studied various agreement problems in directed dynamic networks under different message adversaries.

For oblivious message adversaries it was possible to provide tight bounds on the contraction rate of asymptotic consensus in the setting of non-split graphs for dimensions 1 and 2 as well as to provide a relation to exact terminating consensus. For higher dimensions the lower bounds hold but algorithms to reach this bounds are still missing. The core idea to achieve this results comes from studying valency diameters along executions in relation to contraction rates. Furthermore we obtained a general lower bound of $1/(D+1)$ for any network model in which exact consensus is not solvable; here D denotes the newly introduced α diameter of the network model. We established a connection between the topological structure of valencies and the solvability of exact consensus, and finally, extended our bounds to lower bounds on termination times of approximate consensus algorithms in arbitrary network models. Furthermore, the novel definition of valency for asymptotic consensus will hopefully also be useful in other areas at it is a generalization of the very simple definition used until now.

For non-oblivious message adversaries, we focused on exact terminating consensus and vertex stability of source components, as it is one of the most general assumption used in the field. We showed that memory efficiency, stability intervals and rootedness can be balanced to tailor algorithms to different demands. Furthermore, we could show that stability intervals can be reduced below two times the network depth if and only if an upper bound of n is known. If one can guarantee longer intervals of source stability, one can relax the safety constraint of rootedness. Moreover, we made a significant step towards determining the solvability/impossibility border of general k-set agreement in our model: We provided results, which led us to the first gracefully degrading consensus/k-universal k-set agreement algorithm under a fairly strong message adversary proposed so far.

Our results are complemented by relating our message adversaries to failure detectors and a proof of the fact that weakest resp. necessary failure detectors for consensus resp. k-set agreement cannot be implemented under our message adversaries. We also defined a suitable notion of message adversary simulation, which led to a alternative definition of strongest message adversaries.

Outlook

For oblivious message adversaries, we believe that the study of valencies is an interesting vehicle to approach several open questions: In future work, we plan to close the remaining gaps illustrated in Table 5.1. Especially interesting are improvements in regards to the more general rooted graphs instead of non-split graphs. Moreover, we aim to find simple exact consensus algorithms for general network models, based on an in-depth understanding of how valencies change over executions. Another line of inquiry is how the new definition of valency can provide lower bounds under different failure scenarios, for example crashes failures.

Furthermore, in the area of non-oblivious message adversaries, we are especially interested in a necessary and sufficient property similar to [46], which allows to a priori check whereas an arbitrary non-oblivious message adversary allows a solution for consensus or k-set agreement or not.

List of Tables

List of Algorithms

Bibliography

[1] I. Abraham, Y. Amit, and D. Dolev. Optimal resilience asynchronous approximate agreement. In *International Conference On Principles Of Distributed Systems*, pages 229–239. Springer, 2004.

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

[3] Y. Afek, E. Gafni, and A. Rosen. The slide mechanism with applications in dynamic networks. In *ACM PODC*, pages 35–46, 1992.

[4] M. K. Aguilera, W. Chen, and S. Toueg. Using the heartbeat failure detector for quiescent reliable communication and consensus in partitionable networks. *Theoretical Computer Science*, 220(1):3–30, June 1999.

[5] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393 – 422, 2002.

[6] D. Alistarh, S. Gilbert, R. Guerraoui, and C. Travers. Of choices, failures and asynchrony: The many faces of set agreement. In *ISAAC 2009*, pages 943–953. Springer, Heidelberg, 2009.

[7] D. Angeli and P.-A. Bliman. Stability of leaderless discrete-time multi-agent systems. *MCSS*, 18(4):293–322, 2006.

[8] D. Angluin, J. Aspnes, Z. Diamadi, M. J. Fischer, and R. Peralta. Computation in networks of passively mobile finite-state sensors. In *Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, PODC '04, pages 290–299, New York, NY, USA, 2004. ACM.

[9] D. Angluin, J. Aspnes, D. Eisenstat, and E. Ruppert. The computational power of population protocols. *Distributed Computing*, 20(4):279–304, 2007.

[10] J. Augustine, G. Pandurangan, and P. Robinson. Fast byzantine agreement in dynamic networks. In *Proceedings PODC'13*, pages 74–83, 2013.

[11] J. Augustine, G. Pandurangan, P. Robinson, S. Roche, and E. Upfal. Enabling efficient and robust distributed computation in highly dynamic networks. In *56th Annual IEEE Symposium on Foundations of Computer Science*, FOCS, 2015. To appear.

[12] J. Augustine, G. Pandurangan, P. Robinson, and E. Upfal. Towards robust and efficient computation in dynamic peer-to-peer networks. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 551—569. SIAM, 2012.

[13] B. Awerbuch and S. Even. Efficient and reliable broadcast is achievable in an eventually connected network(extended abstract). In *Proceedings of the third annual ACM symposium on Principles of distributed computing*, PODC '84, pages 278–281, New York, NY, USA, 1984. ACM.

[14] B. Awerbuch, B. Patt-Shamir, D. Peleg, and M. E. Saks. Adapting to asynchronous dynamic networks. In *STOC'92*, pages 557–570, 1992.

[15] I. Ben-Zvi and Y. Moses. Beyond Lamport's happened-before: On the role of time bounds in synchronous systems. In N. Lynch and A. Shvartsman, editors, *Distributed Computing*, volume 6343 of *Lecture Notes in Computer Science*, pages 421–436. Springer Berlin / Heidelberg, 2010.

[16] M. Biely, B. Charron-Bost, A. Gaillard, M. Hutle, A. Schiper, and J. Widder. Tolerating corrupted communication. In *Proceedings of the 26th ACM Symposium on Principles of Distributed Computing (PODC'07)*, pages 244–253, Portland, OR, USA, Aug. 2007. ACM.

[17] M. Biely, P. Robinson, and U. Schmid. Easy impossibility proofs for k-set agreement in message passing systems. *CoRR*, abs/1103.3671, 2011.

[18] M. Biely, P. Robinson, and U. Schmid. Solving k-set agreement with stable skeleton graphs. In *IPDPS Workshops*, pages 1488–1495, 2011. (also available in arxiv:1102.4423).

[19] M. Biely, P. Robinson, and U. Schmid. Agreement in directed dynamic networks. In *Proceedings 19th International Colloquium on Structural Information and Communication Complexity (SIROCCO'12)*, LNCS 7355, pages 73–84. Springer-Verlag, 2012.

[20] M. Biely, P. Robinson, and U. Schmid. The generalized loneliness detector and weak system models for k-set agreement. *IEEE Transactions on Parallel and Distributed Systems*, 25(4):1078–1088, Apr. 2014.

[21] M. Biely, P. Robinson, U. Schmid, M. Schwarz, and K. Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. In *Revised selected papers Third International Conference on Networked Systems (NETYS'15)*, Springer LNCS 9466, pages 109–124, Agadir, Morocco, 2015. Springer International Publishing.

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

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

[24] P.-A. Bliman, A. Nedic, and A. E. Ozdaglar. Rate of convergence for consensus with delays. In *Proceedings of the 47th IEEE Conference on Decision and Control, and the European Control Conference (CDC-ECC)*, pages 2226–2231. IEEE, 2008.

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

[26] E. Borowsky and E. Gafni. Generalized flp impossibility result for t-resilient asynchronous computations. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, pages 91–100, New York, NY, USA, 1993. ACM.

[27] M. Cao, A. S. Morse, and B. D. O. Anderson. Reaching a consensus in a dynamically changing environment: convergence rates, measurement delays, and asynchronous events. *SIAM Journal on Control and Optimization*, 47(2):601–623, 2008.

[28] M. Cao, D. A. Spielman, and A. S. Morse. A lower bound on convergence of a distributed network consensus algorithm. In H. Frey, X. Li, and S. Rührup, editors, *ADHOC-NOW*, pages 2356–2361. IEEE, 2005.

[29] A. Casteigts, S. Chaumette, and A. Ferreira. Characterizing topological assumptions of distributed algorithms in dynamic networks. In *Proceedings of the 16th international conference on Structural Information and Communication Complexity*, SIROCCO'09, pages 126–140, Berlin, Heidelberg, 2010. Springer-Verlag.

[30] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. Deterministic Computations in Time-Varying Graphs: Broadcasting under Unstructured Mobility. In *Proc. of 5th IFIP Conference on Theoretical Computer Science (TCS)*, 2010.

[31] A. Casteigts, P. Flocchini, B. Mans, and N. Santoro. A strict hierarchy of dynamic graphs for shortest, fastest, and foremost broadcast. *CoRR*, abs/1210.3277, 2012.

[32] A. Casteigts, P. Flocchini, W. Quattrociocchi, and N. Santoro. Time-varying graphs and dynamic networks. *IJPEDS*, 27(5):387–408, 2012.

[33] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin. Statistical model of lossy links in wireless sensor networks. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, IPSN '05, Piscataway, NJ, USA, 2005. IEEE Press.

[34] A. Cerpa, J. L. Wong, M. Potkonjak, and D. Estrin. Temporal properties of low power wireless links: Modeling and implications on multi-hop routing. In *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '05, pages 414–425, New York, NY, USA, 2005. ACM.

[35] T. D. Chandra, V. Hadzilacos, and S. Toueg. The weakest failure detector for solving consensus. *Journal of the ACM*, 43(4):685–722, June 1996.

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

[37] B. Charron-Bost, M. Függer, and T. Nowak. Approximate consensus in highly dynamic networks: The role of averaging algorithms. In *Proceedings of ICALP*, pages 528–539, 2015.

[38] B. Charron-Bost, M. Függer, and T. Nowak. Fast, robust, quantizable approximate consensus. In *Proceedings of the 43rd International Colloquium on Automata, Languages, and Programming, ICALP16*, pages 137:1–137:14, 2016.

[39] B. Charron-Bost, M. Függer, and T. Nowak. Multidimensional asymptotic consensus in dynamic networks. *CoRR*, abs/1611.02496, 2016.

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

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

[42] S. Chaudhuri. More *choices* allow more *faults*: Set consensus problems in totally asynchronous systems. *Information and Control*, 105(1):132–158, July 1993.

[43] B. Chazelle. The total s-energy of a multiagent system. *SIAM Journal on Control and Optimization*, 49(4):1680–1706, 2011.

[44] H. C. Chung, P. Robinson, and J. L. Welch. Regional consecutive leader election in mobile ad-hoc networks. In *Proceedings of the DIALM-POMC Joint Workshop on Foundations of Mobile Computing*, pages 81–90, 2010.

[45] H. C. Chung, P. Robinson, and J. L. Welch. Optimal regional consecutive leader election in mobile ad-hoc networks. In *Proceedings of the 7th ACM SIGACT/SIGMOBILE International Workshop on Foundations of Mobile Computing*, FOMC '11, pages 52–61, New York, NY, USA, 2011. ACM.

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

[47] C. Delporte-Gallet, H. Fauconnier, R. Guerraoui, V. Hadzilacos, P. Kouznetsov, and S. Toueg. The weakest failure detectors to solve certain fundamental problems in distributed computing. In *Proceedings of the 23rd ACM Symposium on Principles of Distributed Computing (PODC'04)*, pages 338–346. ACM Press, 2004.

[48] D. Dolev, N. Lynch, S. S. Pinter, E. W. Stark, and W. E. Weihl. Reaching approximate agreement in the presence of faults. *Journal of the ACM*, 33(3):499–516, July 1986.

[49] S. N. Dorogovtsev, J. F. F. Mendes, and A. N. Samukhin. Giant strongly connected component of directed networks. *Phys. Rev. E*, 64:025101, Jul 2001.

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

[51] C. Dwork, D. Peleg, N. Pippenger, and E. Upfal. Fault tolerance in networks of bounded degree. *SIAM J. Comput.*, 17(5):975–988, 1988.

[52] M. Faezipour, M. Nourani, A. Saeed, and S. Addepalli. Progress and challenges in intelligent vehicle area networks. *Commun. ACM*, 55(2):90–100, Feb. 2012.

[53] A. D. Fekete. Asymtotically optimal algorithms for approximate agreement. *Distributed Computing*, 4(1):9–29, March 1990.

[54] A. Ferreira. Building a reference combinatorial model for manets. *Netwrk. Mag. of Global Internetwkg.*, 18(5):24–29, Sept. 2004.

[55] M. Fischer, N. Lynch, and M. Merritt. Easy impossibility proofs for the distributed consensus problem. *Distributed Computing*, 1(1):26–39, 1986.

[56] M. J. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility of distributed consensus with one faulty process. *Journal of the ACM*, 32(2):374–382, Apr. 1985.

[57] M. Függer, T. Nowak, and M. Schwarz. Lower bounds for asymptotic consensus in dynamic networks. *CoRR*, abs/1705.02898, 2017.

[58] M. Fussen, R. Wattenhofer, and A. Zollinger. Interference arises at the receiver. In *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, volume 1, pages 427–432 vol.1, June 2005.

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

[60] A. Goiser, S. Khattab, G. Fassl, and U. Schmid. A new robust interference reduction scheme for low complexity direct-sequence spread-spectrum receivers: Performance. In *Proceedings 3rd International IEEE Conference on Communication Theory, Reliability, and Quality of Service (CTRQ'10)*, pages 15–21, Athens, Greece, June 2010.

[61] F. Greve, L. Arantes, and P. Sens. What model and what conditions to implement unreliable failure detectors in dynamic networks? In *Proceedings of the 3rd International Workshop on Theoretical Aspects of Dynamic Distributed Systems*, TADDS '11, pages 13–17, New York, NY, USA, 2011. ACM.

[62] F. Harary and G. Gupta. Dynamic graph models. *Mathematical and Computer Modelling*, 25(7):79 – 87, 1997.

[63] S. Holzer, Y. A. Pignolet, J. Smula, and R. Wattenhofer. Monitoring churn in wireless networks. *Theor. Comput. Sci.*, 453:29–43, Sept. 2012.

[64] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(04):439–562, August 2006.

[65] R. Ingram, P. Shields, J. E. Walter, and J. L. Welch. An asynchronous leader election algorithm for dynamic networks. In *IPDPS*, pages 1–12, 2009.

[66] S. Janson, D. E. Knuth, T. Luczak, and B. Pittel. The birth of the giant component. *Random Structures Algorithms*, 4:233–358, 1993.

[67] W. Kiess and M. Mauve. A survey on real-world implementations of mobile ad-hoc networks. *Ad Hoc Netw.*, 5(3):324–339, Apr. 2007.

[68] F. Kuhn, N. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *ACM STOC*, pages 513–522, 2010.

[69] F. Kuhn, N. A. Lynch, and R. Oshman. Distributed computation in dynamic networks. In *STOC*, pages 513–522, 2010.

[70] F. Kuhn, Y. Moses, and R. Oshman. Coordinated consensus in dynamic networks. In *30th Annual ACM Symposium on Principles of Distributed Computing*, pages 1–10, New York City, 2011. ACM.

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

[72] F. Kuhn, R. Oshman, and Y. Moses. Coordinated consensus in dynamic networks. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Prin ciples of distributed computing*, PODC '11. ACM, 2011.

[73] F. Kuhn, S. Schmid, and R. Wattenhofer. Towards worst-case churn resistant peer-to-peer systems. *Distributed Computing*, 22(4):249–267, 2010.

[74] F. Legendre, T. Hossmann, F. Sutton, and B. Plattner. 30 years of wireless ad hoc networking research: What about humanitarian and disaster relief solutions? What are we still missing? In *International Conference on Wireless Technologies for Humanitarian Relief (ACWR 11)*, Amrita, India, 2011. IEEE.

[75] I. Litovsky, Y. Métivier, and E. Sopena. Handbook of graph grammars and computing by graph transformation. chapter Graph relabelling systems and distributed algorithms, pages 1–56. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1999.

[76] R. Lubitch and S. Moran. Closed schedulers: a novel technique for analyzing asynchronous protocols. *Distributed Computing*, 8(4):203–210, 1995.

[77] N. Lynch. *Distributed Algorithms*. Morgan Kaufman, 1996.

[78] F. Mattern. Virtual time and global states of distributed systems. In *Parallel and Distributed Algorithms*, pages 215–226. North-Holland, 1989.

[79] H. Mendes, M. Herlihy, N. Vaidya, and V. K. Garg. Multidimensional agreement in byzantine systems. *Distributed Computing*, 28(6):423–441, 2015.

[80] C. Metra, M. Omana, D. Rossi, J. Cazeaux, and T. Mak. The other side of the timing equation: a result of clock faults. In *20th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT 2005)*, pages 169–177, Oct. 2005.

[81] A. Nedic, A. Olshevsky, A. E. Ozdaglar, and J. N. Tsitsiklis. On distributed averaging algorithms and quantization effects. *IEEE Transactions on Automatic Control*, 54(11):2506–2517, 2009.

[82] C. Newport, D. Kotz, Y. Yuan, R. S. Gray, J. Liu, and C. Elliott. Experimental Evaluation of Wireless Simulation Assumptions. *SIMULATION: Transactions of The Society for Modeling and Simulation International*, 83(9):643–661, Sept. 2007.

[83] R. O'Dell and R. Wattenhofer. Information dissemination in highly dynamic graphs. In *Proceedings of the 2005 joint workshop on Foundations of mobile computing*, DIALM-POMC '05, pages 104–110, New York, NY, USA, 2005. ACM.

[84] A. Olshevsky. Linear time average consensus on fixed graphs. *IFAC-PapersOnLine*, 48(22):94–99, 2015.

[85] A. Olshevsky and J. N. Tsitsiklis. Convergence speed in distributed consensus and averaging. *SIAM Review*, 53(4):747–772, 2011.

[86] D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Society for Industrial and Applied Mathematics Monographs on Discrete Mathematics ans Applications, Philadelphia, 2000.

[87] K. J. Perry and S. Toueg. Distributed agreement in the presence of processor and communication faults. *IEEE Transactions on Software Engineering*, SE-12(3):477–482, March 1986.

[88] D. Pfleger. Knowledge and communication complexity. Master's thesis, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/2/182-2, 1040 Vienna, Austria, 2018.

[89] D. Pfleger and U. Schmid. A framework for connectivity monitoring in wireless sensor networks. Research Report TUW-241107, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-2, 1040 Vienna, Austria, 2015. http://publik.tuwien.ac.at/files/PubDat_241107.pdf (to appear in Proc. IARIA SENSORCOMM'16, Nice, France, 2016).

[90] D. Pfleger and U. Schmid. A framework for connectivity monitoring in wireless sensor networks. In *Proceedings 10th International Conference on Sensor Technlogies and Applications (SENSORCOMM'16)*, pages 40–48. IARIA, 2016. https://www.thinkmind.org/download.php?articleid=sensorcomm_2016_3_10_10013.

[91] P. Raipin Parvedy, M. Raynal, and C. Travers. Strongly terminating early-stopping k-set agreement in synchronous systems with general omission failures. *Theor. Comp. Sys.*, 47(1):259–287, July 2010.

[92] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems*, SenSys '03, pages 181–192, New York, NY, USA, 2003. ACM.

[93] R. Ramanathan, P. Basu, and R. Krishnan. Towards a formalism for routing in challenged networks. In *Proceedings of the second ACM workshop on Challenged networks*, CHANTS '07, pages 3–10, New York, NY, USA, 2007. ACM.

[94] M. Raynal and J. Stainer. Synchrony weakened by message adversaries vs asynchrony restricted by failure detectors. In *Proceedings ACM Symposium on Principles of Distributed Computing (PODC'13)*, pages 166–175, 2013.

[95] M. Saks and F. Zaharoglou. Wait-free k-set agreement is impossible: the topology of public knowledge. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, STOC '93, pages 101–110, New York, NY, USA, 1993. ACM.

[96] N. Santoro and P. Widmayer. Time is not a healer. In *Proc. 6th Annual Symposium on Theor. Aspects of Computer Science (STACS'89)*, LNCS 349, pages 304–313, Paderborn, Germany, Feb. 1989. Springer-Verlag.

[97] U. Schilcher, C. Bettstetter, and G. Brandner. Temporal correlation of interference in wireless networks with rayleigh block fading. *IEEE Transactions on Mobile Computing*, 11(12):2109–2120, 2012.

[98] U. Schmid. FWF-proposal ADynNet: Gracefully degrading agreement in directed dynamic networks. Research Report TUW-235381, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-2, 1040 Vienna, Austria, 2014. http://publik.tuwien.ac.at/files/PubDat_235381.pdf.

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

[100] M. Schwarz. Solving k-set agreement in dynamic networks. Master's thesis, Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 3/2/182-2, 1040 Vienna, Austria, 2013.

[101] M. Schwarz and U. Schmid. On the strongest message adversary for consensus in directed dynamic networks, 2018. (submitted).

[102] M. Schwarz, K. Winkler, and U. Schmid. Fast consensus under eventually stabilizing message adversaries. In *Proceedings of the 17th International Conference on Distributed Computing and Networking*, ICDCN '16, pages 7:1–7:10, New York, NY, USA, 2016. ACM.

[103] A. Sealfon and A. A. Sotiraki. Brief announcement: Agreement in partitioned dynamic networks. In *Proceedings 28th International Symposium on Distributed Computing (DISC'14), Springer LNCS 8784*, pages 555–556, 2014.

[104] W. Su, S.-J. Lee, and M. Gerla. Mobility prediction and routing in ad hoc wireless networks. *Int. J. Netw. Manag.*, 11(1):3–30, Jan. 2001.

[105] N. H. Vaidya and D. K. Pradhan. Degradable agreement in the presence of Byzantine faults. In *International Conference on Distributed Computing Systems*, pages 237–244, 1993.

[106] 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.

[107] K. Winkler. *Exploring the Impossibility/Solvability Border for Agreement in Directed Dynamic Networks (working title).* PhD thesis, Technische Universität Wien, Fakultät für Informatik, 2018.

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

[109] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, 52(12):2292 – 2330, 2008.

[110] Y. Yuan, G.-B. Stan, L. Shi, M. Barahona, and J. Goncalves. Decentralized minimum-time consensus. *Automatica*, 49(5):1227–1235, 2013.