Effects of Topology Knowledge and Relay Depth on Asynchronous Appoximate Consensus

Dimitris Sakavalas
Boston College, USA
dimitris.sakavalas@bc.edu

Lewis Tseng
Boston College, USA
lewis.tseng@bc.edu

Nitin H. Vaidya[1]
Georgetown University, USA
nitin.vaidya@georgetown.edu

Abstract

Consider a point-to-point message-passing network. We are interested in the *asynchronous* crash-tolerant consensus problem in *incomplete networks*. We study the feasibility and efficiency of approximate consensus under different restrictions on topology knowledge and the *relay depth*, i.e., the maximum number of hops any message can be relayed. These two constraints are common in large-scale networks, and are used to avoid memory overload and network congestion respectively. Specifically, for positive integer values k and k', we consider that each node knows all its neighbors of at most k-hop distance (*k-hop topology knowledge*), and the relay depth is k'. We consider both *directed* and *undirected* graphs. More concretely, we answer the following question in asynchronous systems:

> *What is a tight condition on the underlying communication graphs for achieving approximate consensus if each node has only a k-hop topology knowledge and relay depth k'?*

To prove that the necessary conditions presented in the paper are also sufficient, we have developed algorithms that achieve consensus in graphs satisfying those conditions:

- The first class of algorithms requires k-hop topology knowledge and relay depth k. Unlike prior algorithms, these algorithms *do not* flood the network, and each node *does not* need the full topology knowledge. We show how the convergence time and the message complexity of those algorithms is affected by k, providing the respective upper bounds.
- The second set of algorithms requires only one-hop neighborhood knowledge, i.e., immediate incoming and outgoing neighbors, but needs to flood the network (i.e., relay depth is n, where n is the number of nodes). One result that may be of independent interest is a *topology discovery* mechanism to learn and "estimate" the topology in asynchronous directed networks with crash faults.

2012 ACM Subject Classification Computer systems organization → Fault-tolerant network topologies

Keywords and phrases Asynchrony, crash, consensus, incomplete graphs, topology knowledge

Digital Object Identifier 10.4230/LIPIcs.OPODIS.2018.14

Related Version A full version of the paper is available at [21], `https://arxiv.org/abs/1803.04513`.

[1] This research is supported in part by National Science Foundation awards 1421918. Any opinions, findings, and conclusions or recommendations expressed here are those of the authors and do not necessarily reflect the views of the funding agencies or the U.S. government.

(a) (b)

Figure 1 Effect of increased k-hop knowledge and relay depth k. In both figures, asynchronous consensus with $f = 1$ is impossible for $k = 1$, but possible for $k = 2$.

1 Introduction

The fault-tolerant consensus problem proposed by Lamport et al. [20] has been studied extensively under different point-to-point network models, including complete networks (e.g., [20, 12]) and undirected networks (e.g., [13, 11]). Recently, many works are exploring various consensus problems in directed networks, e.g., [7, 5, 16], including our own work [23, 25, 22]. More precisely, these works address the problem in *incomplete directed* networks, i.e., not every pair of nodes is connected by a channel, and the channels are not necessarily bi-directional. We will often use the terms *graph* and *network* interchangeably. In this work, we explore the crash-tolerant approximate consensus problem in *asynchronous* incomplete networks under different restrictions on *topology knowledge* – where we assume that each node knows all its neighbors of at most k-hop distance – and *relay depth* – the maximum number of hops that information (or a message) can be propagated. These constraints are common in large-scale networks to avoid memory overload and network congestion, e.g., neighbor table and Time-to-live (TTL) (or hop limit) in the Internet Protocol (IP). We consider both undirected and directed graphs in this paper.

Motivation. Prior results [23] showed that exact crash-tolerant consensus is solvable in *synchronous* networks with only one-hop knowledge and relay depth 1, i.e., each node only needs to know its immediate incoming and outgoing neighbors, and no message needs to be relayed (or forwarded). Such a local algorithm is of interest in practice due to low deployment cost and low message complexity. In *asynchronous* undirected networks, there exists a simple flooding-based algorithm adapted from [13, 11] that achieves approximate consensus with up to f crash faults if the network satisfies $(f + 1)$ node-connectivity[2] and $n > 2f$, where n is the number of nodes. However, these two conditions are *not* sufficient for an *iterative* algorithm with one-hop knowledge and relay depth 1, in which each node maintains a state and exchanges state values with only one-hop neighbors in each iteration.

Consider Figure 1a, which is a ring network of four nodes. There is no iterative algorithm with one-hop knowledge and relay depth 1 under one crash fault. The adversary can divide the nodes into disjoint sets $\{a, b\}$ and $\{c, d\}$ such that the communication delay across sets is so large that a thinks d has crashed, and d thinks a has crashed, and similarly for the pair b and c. As a result, no exchange of state values is possible across the sets in the execution; hence, consensus is not possible (a more precise discussion in Section 3). On the other hand, suppose each node has two-hop knowledge, i.e., a complete topology knowledge in this network, and relay depth 2. Then a knows that it will be able to receive state values from at least two of the other nodes since the node connectivity is 2, and up to one node may fail.

[2] For brevity, we will simply use the term "connectivity" in the presentation below.

Following this observation, it is easy to design a flooding-based algorithm in the ring network based on [13, 11]. This example shows that both topology knowledge and relay depth affect the feasibility of asynchronous approximate consensus.

Interestingly, increasing connectivity alone does *not* make iterative algorithm feasible. In the full version [21], we show that no fault-tolerant approximate consensus algorithm with one-hop topology and relay depth 1 exists in the network in Figure 1b, which has two sparsely-connected cliques of size $n/2$ and connectivity $n/2 - 1$. Motivated by these observations, this work addresses the following question in *asynchronous* systems:

> What is a tight condition on the underlying communication graphs for achieving approximate consensus if each node has only a k-hop topology knowledge and relay depth k'?

The problem. We consider the asynchronous approximate consensus problem. The system consists of n nodes, of which at most f nodes may crash. Each node is given an input, and after a finite amount of time, each fault-free node should produce an output, which satisfies *validity* and *agreement* conditions (formally defined later). Intuitively, the state at fault-free nodes must be in the range of all the inputs, and are guaranteed to be within ϵ of each other for some $\epsilon > 0$ after a sufficiently large number of rounds.

In [23], we presented Condition CCA (Crash-Consensus-Asynchronous, see definition in Section 2) and showed that it is necessary and sufficient on the underlying directed graphs for achieving *approximate* consensus in asynchronous systems [23]. The approximate consensus algorithms in prior work [23, 13, 11] are based on flooding (i.e., relay depth n) and assume that each node has n-hop topology knowledge. However, such an algorithm is *not* practical in a large-scale network, since, (*i*) nodes' local memory may not be large enough to store the entire network, (*ii*) flooding-based algorithms (e.g., [23, 13, 11]) incur prohibitively high message overhead for each phase, and (*iii*) complete topology knowledge may require a high deployment and configuration cost. Therefore, we explore algorithms that only require "local" knowledge and limited message relay.

Contributions. We identify tight conditions on the graphs under different assumptions on topology knowledge and relay depth. Particularly, we have the following results:

- *Limited Topology Knowledge and Relay Depth* (Section 3): We consider the case with k-hop topology knowledge and relay depth k. The family of algorithms that captures these constrains are iterative k-hop algorithms – nodes only have topology knowledge of their k-hop neighborhoods, and propagate state values to nodes that are at most k-hops away. Note that *no* other information is relayed. For iterative k-hop algorithms, we derive a family of tight conditions, namely Condition k-CCA for $1 \leq k \leq n$, for solving *approximate* consensus in directed networks. To prove the tightness of the conditions, we propose a family of iterative algorithms called k-LocWA and show how the convergence time and the message complexity of those algorithms is affected by k, providing the respective upper bounds.
- *Topology Discovery and Unlimited Relay Depth* (Section 4): We consider the case with one-hop topology knowledge and relay depth n. In other words, nodes initially only know their immediate incoming and outgoing neighbors, but nodes can flood the network, learn (some part of) the topology, and eventually solve consensus based on the learned topology. We show that Condition CCA from [23] is also sufficient in this case. Since we assume only one-hop knowledge, our result implies that Condition CCA is tight for any k-hop topology knowledge. One contribution that may be of independent interest is

a *topology discovery* mechanism to learn and "estimate" the topology in asynchronous directed networks with crash faults. Such a discovery mechanism will be useful for self-stabilization and reconfiguration of a large-scale system.

In Section 5, we discuss fault-tolerance implications of the derived conditions and Condition CCA. We also discuss how to speed up our algorithms in terms of real time delay.

Related Work. There is a large body of work on fault-tolerant consensus. Here, we discuss related works exploring consensus in different assumptions on graphs. Fischer et al. [13] and Dolev [11] characterized necessary and sufficient conditions under which Byzantine consensus is solvable in *undirected* graphs. In synchronous systems, Charron-Bost et al. [7, 8] solved *approximate* crash-tolerant consensus in dynamic directed networks using local averaging algorithms, and in the asynchronous setting, Charron-Bost et al. [7, 8] addressed approximate consensus with crash faults in *complete* graphs which are necessarily undirected. We solve the problem in *incomplete directed* graphs in asynchronous systems. Moreover, in [7, 8], nodes are *constrained* to only have the one-hop topology knowledge. We study different types of algorithms, including the ones that allow nodes to learn the topology (i.e., we allow topology discovery).

There were also works studying limited topology knowledge. Su and Vaidya [22] identified the condition for solving synchronous Byzantine consensus using a variation of k-hop algorithms. Alchieri et al. [1] studied the synchronous Byzantine problem under *unknown* participants. We consider *asynchronous* systems in this work. Nesterenko and Tixeuil [17] studied the topology discovery problem in the presence of Byzantine faults in *undirected* networks, whereas we present a solution that works in *directed* networks with crash faults.

Extensive prior works studied graph properties for other similar problems in the presence of Byzantine failures, such as (i) Byzantine approximate consensus in directed graphs using "local averaging" algorithms wherein nodes only have one-hop neighborhood knowledge (e.g., [25, 24, 22, 26, 10]), (ii) Byzantine consensus with unknown participants [1], (iii) Byzantine consensus with *authentication* in *undirected* networks [3]. These papers only consider synchronous systems, and our algorithms and analysis are significantly different from those developed for Byzantine algorithms, and (iv) consensus problems in *synchronous* dynamic networks where the adversary can change the network topology. In this line of work, impossibility results for consensus and k-set agreement are given in [4, 6] and sufficiency is guaranteed by requiring a period of stability, during which certain nodes are strongly connected; the first tight condition for the feasibility of consensus and broadcast is presented in [9]. Additionally, in [2], Byzantine corruptions and a dynamic node set is assumed and a $O(\log^3 n)$-round randomized algorithm is presented. Our work is different from all these works because of the assumption of asynchronous systems and limited topology knowledge.

2 Preliminary

Before presenting the results, we introduce our system model, some terminology, and our prior results from [23] to facilitate the discussion.

System Model. The point-to-point message-passing network is *static*, and it is represented by a simple *directed* graph $G(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of n nodes, and $\mathcal{E}$ is the set of directed edges between the nodes in $\mathcal{V}$. The communication links are reliable. We assume that $n \geq 2$, since the consensus problem for $n = 1$ is trivial. Node i can transmit messages to another node j directly if directed edge (i, j) is in $\mathcal{E}$. Each node can send messages to itself as well;

however, for convenience, we exclude self-loops from set $\mathcal{E}$. We will use the terms *edge* and *link* interchangeably.

Up to f nodes may suffer crash failures in an execution. A node that suffers a crash failure simply stops taking step (i.e., fail-stop model). We consider the *asynchronous* message-passing communication, in which a message may be delayed arbitrarily but eventually delivered if the receiver node is fault-free. We assume that the adversary has both the control of crashing nodes and delaying messages at any point of time during the execution.

Terminology. Upper case letters are used to name sets. Lower case italic letters are used to name nodes. All paths used in our discussion are directed paths.

Node j is said to be an incoming neighbor of node i if $(j,i) \in \mathcal{E}$. Let N_i^- be the set of incoming neighbors of node i, i.e., $N_i^- = \{j \mid (j,i) \in \mathcal{E}\}$. Define N_i^+ as the set of outgoing neighbors of node i, i.e., $N_i^+ = \{j \mid (i,j) \in \mathcal{E}\}$.

For set $B \subseteq \mathcal{V}$, node i is said to be an incoming neighbor of set B if $i \notin B$, and there exists $j \in B$ such that $(i,j) \in \mathcal{E}$. Given subsets of nodes A and B, set B is said to have k incoming neighbors in set A if A contains k distinct incoming neighbors of B.

▶ **Definition 1.** Given disjoint non-empty subsets of nodes A and B, $A \overset{x}{\Rightarrow} B$ if B has at least x distinct incoming neighbors in A. When it is not true that $A \overset{x}{\Rightarrow} B$, we will denote that fact by $A \overset{x}{\not\Rightarrow} B$.

Approximate Consensus. For the approximate consensus problem (e.g., [12, 15, 23]), it is usually assumed that each node i maintains and regularly updates a *state*, with $v_i[p]$ denoting the p-th update of the state of node i. In asynchronous systems, value $v_i[p]$ is also called the state of node i at the end of phase (or iteration) p. The initial state of node i, $v_i[0]$, is equal to the initial input provided to node i. At the start of phase p ($p > 0$), the state of node i is $v_i[p-1]$.

Let $U[p]$ and $\mu[p]$ be the maximum and the minimum state at nodes that have not crashed by the end of phase p. Then, a *correct* approximate consensus algorithm needs to satisfy the following two conditions:

- *Validity*: $\forall p > 0, U[p] \leq U[0]$ and $\mu[p] \geq \mu[0]$; and
- *Convergence*: $\lim_{p\to\infty} U[p] - \mu[p] = 0$.

Equivalently the Convergence condition can be stated as:

$\forall \epsilon > 0$, there exists a phase p_ϵ such that for $p > p_\epsilon, U[p] - \mu[p] < \epsilon$.

Towards facilitating the study of the number of phases needed for convergence and the corresponding message complexity, observe that convergence with respect to a specific ϵ must be considered. Therefore we will also use the following convergence notion.

- ϵ-Convergence: $\exists p_\epsilon$, $\forall p \geq p_\epsilon$, $U[p] - \mu[p] \leq \epsilon$.

Remark on Termination. The variation of approximate consensus defined above, also appears in the literature under the name of *asymptotic consensus* (cf. [14]). The main difference is that in the original approximate consensus problem defined in [12], termination for any ϵ is also required. However, we stress that the algorithms we present can trivially be extended to achieve termination using an approach similar to the ones presented in [12, 15]. In these works and in most of the approximate consensus literature, the problem is solved by iterative algorithms of similar structure and each node can locally compute an upper bound on the number of iterations that are needed for ϵ-convergence. Thus termination is easily guaranteed.

Prior Result. In [23], we identified necessary and sufficient conditions on the underlying communication graphs $G(\mathcal{V}, \mathcal{E})$ for achieving *crash-tolerant consensus* in directed networks. The theorem below, presented in [23], states that condition **CCA** (Crash-Consensus-Asynchronous) is tight for approximate consensus under full topology knowledge and relay depth. Observe that, naturally, the impossibility result holds regardless of those parameters.

▶ **Theorem 2** (from [23]). *Approximate crash-tolerant consensus in asynchronous systems, under full topology knowledge and relay depth, is feasible iff for any partition L, C, R of $\mathcal{V}$, where L and R are both non-empty, either $L \cup C \overset{f+1}{\Rightarrow} R$ or $R \cup C \overset{f+1}{\Rightarrow} L$.* ***(Condition CCA)***

3 Limited Topology Knowledge and Relay Depth

In this section, we study how topology knowledge and the relay depth affect the *tight* conditions on the directed communication network. Particularly, we consider the case with k-hop topology knowledge and relay depth k for $1 \leq k \leq n$. Prior works (e.g., [23, 13, 11]) assumed that each node has n-hop topology knowledge and relay depth n. However, in large-scale networks, such an assumption may not be realistic. Partial knowledge models have been recently explored in [18, 19]. We are interested in algorithms that only require nodes to exchange a small amount of information within their local neighborhoods. One other benefit is that these algorithms do not require flooding [23] or all-to-all communication [13, 11] in each asynchronous phase.

We are interested in iterative k-hop algorithms – nodes only have topology knowledge in their k-hop neighborhoods, and propagate state values to nodes that are at most k-hops away. We introduce a family of conditions, namely Condition k-CCA for $1 \leq k \leq n$, which we prove necessary and sufficient for achieving asynchronous approximate consensus, through the use of iterative k-hop algorithms. The results presented in this section also imply how the parameter k affects the *tight* conditions on the directed networks. To the best of our knowledge, two prior papers [1, 22] examined a similar problem – *synchronous* Byzantine consensus. In [22], Su and Vaidya identified the condition under different relay depths. Alchieri et al. [1] studied the problem under *unknown* participants. The technique developed for asynchronous consensus in this section is significantly different.

Observe that since the system is asynchronous, any algorithm has to be event-oriented. For this reason, we define a locally verifiable condition *WAIT*, which dictates the end of the reception step and the start of the state update step as seen below. Different *WAIT* conditions need to be defined for each algorithm we consider.

Iterative k-hop Algorithms. The iterative algorithms considered here have relay depth k and require each node i to perform the following three steps in *asynchronous* phase t:

1. *Transmit*: Transmit messages of the form $(v_i[t-1], t-1)$ to nodes that are reachable from node i via *at most k hops away*, where $v_i[t-1]$ is the current state value, which is accompanied by the phase tag $t-1$. If node i is an intermediate node on the route of some message, then node i forwards that message as instructed by the source;
2. *Receive*: Until a condition *WAIT* is satisfied, receive messages from the nodes that can reach node i via at most k hops. Denote by $R_i[t]$ the set of messages that node i received at phase t; and
3. *Update*: Once condition *WAIT* is satisfied, update state using a transition function Z_i, where Z_i is a part of the specification of the algorithm, and takes as input the set $R_i[t]$. i.e.,

$$v_i[t] := Z_i(R_i[t], v_i[t-1]) \quad \text{at node} \quad i$$

Note that (i) no exchange of topology information takes place in this class of algorithms, and (ii) each node's state only propagates within its k-hop neighborhood. For a node i, its *k-hop incoming neighbors* are defined as the nodes j which are connected to i by a directed path in G that has $\leq k$ hops. The notion of k-hop outgoing neighbors is defined similarly.

Technique. The algorithms presented in this section are motivated by prior work [12, 22] including our own work [23]. The algorithms are iterative and simple; thus, the proof structure shares some similarity with prior work [12, 23, 25].

Generally speaking, the proof proceeds as following: (i) nodes are divided into two disjoint sets, say L and R so that nodes have "closer" state values in each set; (ii) because each node receives an adequate set of messages, we show that under any delay and crash scenarios, at least one non-crashed node in either L or R will receive one message from the other set of nodes in each phase; and (iii) after enough phases, the value of all non-crashed nodes in either L or R will move "closer" to the values in the other set. Two key novelties in this paper are: identifying the "adequate set" of messages that needs to be received before updating local state in each asynchronous phase, and showing that even with limited k-hop propagation, some node is still able to receive messages from the other set (satisfying the above point (ii)).

3.1 $k = 1$ Case

To initiate the study, we first consider the one-hop case, where each node only knows its one-hop incoming and outgoing neighbors. The following notion is crucial for the characterization of graphs in which asynchronous approximate consensus is feasible with relay depth 1.

▶ **Definition 3** ($A \to B$)**.** Given disjoint non-empty subsets of nodes A and B, we will use the notation $A \to B$ if there exists a node i in B such that i has at least $f+1$ distinct incoming neighbors in A. When it is not true that $A \to B$, we will denote that fact by $A \not\to B$.

Condition 1-CCA, presented below proves to be necessary and sufficient for achieving asynchronous approximate consensus with relay depth 1. Note that 1-CCA requires the existence of a single node that has at least $f+1$ incoming neighbors, while CCA requires the distinct incoming neighbors of the corresponding set to be at least $f+1$.

▶ **Definition 4** (Condition 1-CCA)**.** For any partition L, C, R of $\mathcal{V}$, where L and R are both non-empty, either $L \cup C \to R$ or $R \cup C \to L$.

The necessity of Condition 1-CCA for the specific class of iterative 1-hop algorithms, is similar to the necessity proof of Condition CCA in [23] and is presented in the full version [21].

For sufficiency, we present Algorithm LocWA (Local-Wait-Average), which is inspired by Algorithm WA [23] [3]. Note that LocWA utilizes only one-hop information. Recall that by definition, no message relay with depth greater than 1 is allowed. In Algorithm LocWA, $heard_i[p]$ is the set of one-hop incoming neighbors of i from which i has received values during phase p. Each node i performs the averaging operation to update its state value when Condition 1-WAIT below holds for the first time in phase p.

[3] The main difference lies in the WAIT condition and the fact that no relay of messages takes place in LocWA. Also, the termination of LocWA can be dealt with as argued in the corresponding remark in Section 2.

Algorithm 1: LocWA for node $i \in \mathcal{V}$.

$v_i[0]$:= input at node i

For phase $p \geq 1$:

* On entering phase p:

$R_i[p] := \{v_i[p-1]\}$

$heard_i[p] := \{i\}$

Send message $(v_i[p-1], i, p)$ to all the outgoing neighbors

* When message (h, j, p) is received for the *first time*:

$R_i[p] := R_i[p] \cup \{h\}$ // $R_i[p]$ is a multiset

$heard_i[p] := heard_i[p] \cup \{j\}$

* When Condition *1-WAIT* holds for the first time in phase p:

$$v_i[p] := \frac{\sum_{v \in R_i[p]} v}{|R_i[p]|} \tag{1}$$

Enter phase $p+1$

Condition 1-WAIT. The condition is satisfied at node i, in phase p, when $|heard_i[p]| \geq |N_i^-| - f$, i.e., when i has not received values from a set of at most f incoming neighbors.

To prove the correctness of LocWA, we will use the supplementary definitions below.

▶ **Definition 5.** For disjoint sets A, B, $in(A \to B)$ denotes the set of all the nodes in B that each have at least $f+1$ incoming edges from nodes in A. When $A \not\to B$, define $in(A \to B) = \emptyset$. Formally, $in(A \to B) = \{\ v \mid v \in B \text{ and } \ f+1 \leq |N_v^- \cap A|\ \}$.

▶ **Definition 6.** For *non-empty disjoint* sets A and B, set A is said to *propagate to* set B in l steps, where $l > 0$, if there exist sequences of sets $A_0, A_1, A_2, \cdots, A_l$ and $B_0, B_1, B_2, \cdots, B_l$ (propagating sequences) such that

- $A_0 = A, \quad B_0 = B, \quad A_l = A \cup B, \quad B_l = \emptyset, \quad B_\tau \neq \emptyset$ for $\tau < l$, and
- for $0 \leq \tau \leq l-1$, (i) $A_\tau \to B_\tau$; (ii) $A_{\tau+1} = A_\tau \cup in(A_\tau \to B_\tau)$; and (iii) $B_{\tau+1} = B_\tau - in(A_\tau \to B_\tau)$.

Observe that A_τ and B_τ form a partition of $A \cup B$, and for $\tau < l$, $in(A_\tau \to B_\tau) \neq \emptyset$. We say that set A propagates to set B if there is a propagating sequence for some steps l as defined above. Note that the number of steps l in the above definition is upper bounded by $n-f-1$, since set A must be of size at least $f+1$ for it to propagate to B; otherwise, $A \not\to B$.

Now, we present two key lemmas whose proofs are presented in the full version [21]. In the discussion below, we assume that G satisfies Condition 1-CCA.

▶ **Lemma 7.** *For any partition A, B of $\mathcal{V}$, where A, B are both non-empty, either A propagates to B, or B propagates to A.*

The lemma below states that the interval to which the states at all the fault-free nodes are confined shrinks after a finite number of phases of Algorithm LocWA. Recall that $U[p]$ and $\mu[p]$ denote the maximum and minimum states at the fault-free nodes at the end of the p-th phase and. We also denote with $F[p]$, the nodes that have not computed value $v[p]$ in phase p, i.e., nodes in $F[p]$ have crashed before computing $v[p]$.

▶ **Lemma 8.** *Suppose that at the end of the p-th phase of Algorithm LocWA, $\mathcal{V}$ can be partitioned into non-empty sets R and L such that (i) R propagates to L in l steps, and (ii)*

the states of fault-free nodes in $R - F[p]$ are confined to an interval of length $\leq \frac{U[p]-\mu[p]}{2}$. Then, with Algorithm LocWA,

$$U[p+l] - \mu[p+l] \;\leq\; \left(1 - \frac{\alpha^l}{2}\right)(U[p] - \mu[p]), \qquad \textit{where} \quad \alpha = \min_{i \in \mathcal{V}} \frac{1}{|N_i^-|} \tag{2}$$

Using lemma 8 and simple algebra, we can prove the following Theorem. For the sake of space, we present only a proof sketch. The complete proof is deferred to the full version [21].

▶ **Theorem 9.** *If $G(\mathcal{V}, \mathcal{E})$ satisfies Condition 1-CCA, then Algorithm LocWA achieves both Validity and Convergence.*

Proof Sketch. To prove the Convergence of LocWA, we show that given any $\epsilon > 0$, there exists τ such that $U[t] - \mu[t] \leq \epsilon, \forall t \geq \tau$. Consider p-th phase, for some $p \geq 0$. If $U[p] - \mu[p] = 0$, then the algorithm has already converged; thus, we consider only the case where $U[p] - \mu[p] > 0$. In this case, we can partition $\mathcal{V}$ into two subsets, A and B, such that, for each fault-free node $i \in A$, $v_i[p] \in \left[\mu[p], \frac{U[p]+\mu[p]}{2}\right)$, and for each fault-free node $j \in B$, $v_j[p] \in \left[\frac{U[p]+\mu[p]}{2}, U[p]\right]$. (Full proof in [21], identifies how to partition the nodes.) By Lemma 7, we have that either A propagates to set B or B propagates to A. In both cases above, we have found two non-empty sets $L = A$ (or $L = B$) and $R = B$ (or $L = A$) partitioning $\mathcal{V}$ and satisfy the hypothesis of Lemma 8, since R propagates to L and the states of all fault-free nodes in R are confined to an interval of length $\leq \frac{U[p]-\mu[p]}{2}$. The theorem is then proven by using simple algebra and the fact that the interval to which the states of all the fault-free nodes are confined shrinks after a finite number of phases. ◀

3.2 General k Case

Now, consider the case when each node only knows its k-hop neighbors and the relay depth is k. In the following, we generalize the notions presented above to the k-hop case. For node i, denote by $N_i^-(k)$ the set of i's k-hop incoming neighbors, For a set of nodes A, let N_A^- be the set of A's one-hop incoming neighbors. Formally, $N_A^- = \{i \mid i \in \mathcal{V} - A, \text{ and } \exists j \in A, (i,j) \in \mathcal{E}\}$. Next we define the relation $A \to B$ for the k-hop case.

▶ **Definition 10** ($A \to_k B$)**.** Given disjoint non-empty subsets of nodes A and B, we will say that $A \to_k B$ holds if there exists a node i in B for which there exist at least $f+1$ node-disjoint paths of length at most k from distinct nodes in A to i. More formally, if $\mathcal{P}_i^A(k)$ is the family of all sets of k-length node-disjoint paths (with i being their only common node) initiating in A and ending in node i, $A \to_k B$ means that $\exists i \in B$, $\max_{P \in \mathcal{P}_i^A(k)} |P| \geq f+1$.

▶ **Definition 11** (Condition k-CCA)**.** For any partition L, C, R of $\mathcal{V}$, where L and R are both non-empty, either $L \cup C \to_k R$ or $R \cup C \to_k L$.

The necessity of Condition k-CCA for achieving asynchronous approximate consensus through an iterative k-hop algorithm holds analogously with the one-hop case, where a set of x incoming neighbors of node i has to be replaced with a set of x distinct nodes that reach i through disjoint paths. For sufficiency, we next present a generalization of Algorithm LocWA for the k-hop case. There are two differences between Algorithms k-LocWA and LocWA: (i) nodes transmit their state to all their k-hop outgoing neighbors, and (ii) Algorithm k-LocWA relies on the generalized version of Condition 1-WAIT, presented below.

Algorithm 2: k-LocWA for node $i \in \mathcal{V}$.

$v_i[0]$:= input at node i
For phase $p \geq 1$:
* On entering phase p:
  $d_i[p] := 1$
  $R_i[p] := \{v_i[p-1]\}$
  $heard_i[p] := \{i\}$
  Send message $(v_i[p-1], i, p)$ to nodes in $N_i^+(k)$, all k-hop outgoing neighbors[a]
* When message (h, j, p) is received for the *first time*:
  $R_i[p] := R_i[p] \cup \{h\}$ // $R_i[p]$ is a multiset
  $heard_i[p] := heard_i[p] \cup \{j\}$
* When Condition k-*WAIT* holds for the first time in phase p:
  $v_i[p] := \frac{\sum_{v \in R_i[p]} v}{|R_i[p]|}$
  Enter phase $p+1$

[a] For brevity, we do not specify how the network routes the messages within the k-hop neighborhood – this can be achieved by using local flooding through tagging a hop counter in each message.

Condition k-WAIT. For $F_i \subseteq N_i^-(k)$, we denote with $reach_i^k(F_i)$ the set of nodes that have paths of length $l \leq k$ to node i in G_{V-F_i}. That is, the set of k-hop incoming neighbors of i that remain connected with i even when all nodes in set F_i crash. The condition is satisfied at node i, in phase p if there exists $F_i \subseteq N_i^-(k)$ with $|F_i[p]| \leq f$ such that $reach_i^k(F_i[p]) \subseteq heard_i[p]$.

Correctness of Algorithm k-LocWA. Proving the correctness of k-LocWA follows a similar reasoning of the correctness of LocWA. The key here is to identify Condition k-CCA and Condition k-WAIT so that the proof structure remains almost identical. To adapt the arguments to the general case, one should define the analogous $in(A \to_k B)$ definition based on the general $A \to_k B$ notion.

▶ **Definition 12.** For disjoint sets A, B, $in(A \to_k B)$ denotes the set of all the nodes i in B such that there exist at least $f+1$ incoming disjoint paths of length at most k from distinct nodes in $N_i^- \cap A$ to i. When $A \not\to_k B$, define $in(A \to_k B) = \emptyset$. Formally, in the terminology of Definition 10: $in(A \to B) = \{i \in B : \max\{|p| : p \in P_i^A(k)\} \geq f+1\}$

The following proof sketch outlines necessary adaptations for general k case proof.

▶ **Theorem 13.** *Approximate crash-tolerant consensus in an asynchronous system using iterative k-hop algorithms is feasible iff G satisfies Condition k-CCA.*

Proof Sketch. Having defined the basic notion $in(A \to_k B)$, Definition 6 of the notion A *propagates to* B is the same for the k-hop case. Intuitively, if A *propagates to* B, information will be propagated gradually from A to B in l steps. Any faulty set of f nodes will *not* be able to block propagation from A to a specific node $i \in B$ because the definition of $in(A \to_k B)$ guarantees that i will receive information from at least $f+1$ disjoint paths if it has not crashed. A difference with the $k=1$ case is that for every of the l steps needed to propagate from A to B, k communication steps will be required in the worst case, since information may be propagated through paths of length k. Lemma 8 is intuitively the same since it is based on the general propagation notion but value α which is defined based on

the number of incoming neighbors will now be defined on the number of k-hop incoming neighbors, i.e., $\alpha_k = \min_{i \in \mathcal{V}} \frac{1}{|N_i^-(k)|}$. The main correctness proof remains essentially the same since it repeatedly makes use of the abstract propagation notion between various sets, without focusing on how the values are propagated. ◀

3.3 Condition Relation and Convergence Time Comparison

Next, we first compare the feasibility of approximate consensus for different values of k by presenting a relation among the various k-CCA conditions as well as their relation with Condition CCA from [23].

Condition Relation

Intuitively, achieving approximate consensus for a lower k requires the existence of more paths in the graph; this can be observed by definition and is summarized in the following theorem.

▶ **Theorem 14.** *For values $k, k' \in \mathbb{N}$ with $k \leq k'$, Condition k-CCA implies Condition k'-CCA.*

Proof. Let Condition k-CCA hold and assume, without loss of generality that $L \cup C \rightarrow_k R$ holds for a partition L, C, R. This means that there exists a node i in R that has at least $f+1$ incoming disjoint paths of length at most k initiating from distinct nodes in $L \cup C$. Consequently, the same $f+1$ paths will consist of i's incoming disjoint paths of length at most k', since $k' \geq k$, and thus, $L \cup C \rightarrow_{k'} R$ which means that k'-CCA holds. ◀

We next show that Condition CCA is equivalent to Condition n-CCA. The proof illustrates how the locally defined Condition k-CCA naturally coincides with the globally defined condition CCA in the extreme case.

▶ **Theorem 15.** *Condition CCA is equivalent to Condition n-CCA.*

Proof. It is easy to see that Condition n-CCA implies Condition CCA. If Condition CCA is violated in G, then Condition n-CCA does not hold either, since L and R have at most f one-hop incoming neighbors.

Now, we show the other direction. Assume for the sake of contradiction that Condition CCA holds but Condition n-CCA does not. Then, there exists a partition L, C, R with $L, R \neq \emptyset$ such that $L \cup C \not\rightarrow_k R$ and $R \cup C \not\rightarrow_k L$. Since Condition CCA holds, we have that either $L \cup C \overset{f+1}{\Rightarrow} R$ or $R \cup C \overset{f+1}{\Rightarrow} L$. Now consider the case that $L \cup C \overset{f+1}{\Rightarrow} R$ and $R \cup C \overset{f+1}{\not\Rightarrow} L$. This means that $|N_R^-| \geq f+1$ and $|N_L^-| \leq f$. The case of $L \cup C \overset{f+1}{\not\Rightarrow} R$ and $R \cup C \overset{f+1}{\Rightarrow} L$ is symmetrical and the case of $L \cup C \overset{f+1}{\Rightarrow} R$ and $R \cup C \overset{f+1}{\Rightarrow} L$ can be proved by applying the argument below once for set R and once for set L.

Let i be the node in R with the maximum number m of disjoint paths initiating from distinct nodes in $V - R$ (as implied by Definition 10). The fact $L \cup C \not\rightarrow_k R$ implies that $m \leq f$. Subsequently, $|N_R^-| \geq f+1$ implies that the set $A = N_R^- - N_i^-(n)$ is non-empty (the maximal subset of N_R^- which does not contain any n-hop incoming neighbors of i). Let $B = N_A^+(n) \cap R$ be the set of all the outgoing n-hop neighbors of all nodes $j \in A$ confined in the set R. By definition of B and A, it holds that $N_i^-(n) \cap B = \emptyset$. We can now create a new partition $L' = L, C' = C \cup B, R' = R - B$ by moving B from R to C. For partition L', C', R' it holds that $L', R' \neq \emptyset$ since $i \in R'$ and $L' = L$. Moreover, it holds that (i) $|N_{R'}^-| \leq f$, since

$|N^-_{R'}| = |N^-_R - A|$ and $A \neq \emptyset$; and (ii) $|N^-_L| \leq f$ since $L = L'$. The latter points imply that $R \cup C \overset{f+1}{\not\Rightarrow} L$ and $L \cup C \overset{f+1}{\not\Rightarrow} R$, which yield a contradiction to the hypothesis that Condition CCA holds. This completes the proof. ◀

Convergence Time Comparison

We derive upper bounds on the number of *asynchronous* phases needed for ϵ-convergence of Algorithm k-LocWA and its message complexity up to this ϵ-convergence point p_ϵ. These upper bounds are functions of values ϵ, k, f, n and $\delta = U[0] - \mu[0]$ which are naturally expected to affect the convergence time and message complexity. Moreover, since the bounds depend on k, it provides a way to compare the convergence time and message complexity of Algorithms k-LocWA for different values of k. Next, we present the upper bound on the convergence time of k-LocWA, the proof of the theorem is deferred to the full version [21].

▶ **Theorem 16** (Convergence-time complexity). *The number of phases required by Algorithm k-LocWA to ϵ-converge is* $O\left(\dfrac{(n-f)\log \epsilon/\delta}{\log\left(1 - \frac{\alpha_k^{n-f-1}}{2}\right)}\right)$.

Comparison of Algorithms k-LocWA Convergence. Observe that the above bound decreases, as the maximum number of k-hop incoming neighbors increases, since $\alpha_k = \min_{i \in \mathcal{V}} \frac{1}{|N_i^-(k)|}$. Since the maximum number of k-hop incoming neighbors increases with k we have that for $k' > k$, Algorithm k'-LocWA ϵ-converges faster than k-LocWA by a factor implied by the bound. Moreover, given the upper bound on phases for ϵ-convergence of Theorem 16 we can easily derive an upper bound on the message complexity of k-LocWA as is shown in [21].

4 Topology Discovery and Unlimited Relay Depth

In this section, we consider the case with one-hop topology knowledge and relay depth n. In other words, nodes initially only know their immediate incoming and outgoing neighbors, but nodes can flood the network and learn the topology. The study of this case is motivated by the observation that full topology knowledge at each node (e.g., [23, 13, 11]) requires a much higher deployment and configuration cost. We show that Condition CCA from [23] is necessary and sufficient for solving approximate consensus with one-hop neighborhood knowledge and relay depth n in asynchronous directed networks. Compared to the iterative k-hop algorithms in Section 3, the algorithms in this section are *not* restricted in the sense that nodes can propagate any messages to all the reachable nodes.

The necessity of Condition CCA is implied by our prior work [23]. The algorithms presented below are again inspired by Algorithm WA from [23]. The main contribution is to show how each node can learn "enough" topology information to solve approximate consensus – this technique may be of interest in other contexts as well. In the discussion below, we present an algorithm that works in any directed graph that satisfies Condition CCA.

Algorithm LWA. The idea of Algorithm LWA (Learn-Wait-Average) is to piggyback the information of incoming neighbors when propagating state values. Then, each node i will locally construct an *estimated* graph $G^i[p]$ in every phase p, and check whether Condition

Algorithm 3: LWA for node $i \in \mathcal{V}$.

$v_i[0]$:= input at node i
$G^i[0] := G_{N_i^- \Rightarrow i}$
For phase $p \geq 1$:
* On entering phase p:
  $R_i[p] := \{v_i[p-1]\}$
  $heard_i[p] := \{i\}$
  Send message $(v_i[p-1], N_i^-, i, p)$ to all the outgoing neighbors
* When message (h, N, j, p) is received for the *first time*:
  $R_i[p] := R_i[p] \cup \{h\}$ // $R_i[p]$ is a multiset
  $heard_i[p] := heard_i[p] \cup \{j\}$
  $G^i[p] := G^i[p] \cup G_{N \Rightarrow j}$ [a]
  Send message (h, N, j, p) to all the outgoing neighbors
* When Condition n-*WAIT* holds on $G^i[p]$ for the first time in phase p:
  $v_i[p] := \frac{\sum_{v \in R_i[p]} v}{|R_i[p]|}$
  $G^i[p+1] := G_{N_i^- \Rightarrow i}$ // "Reset" the learned graph
  Enter phase $p+1$

[a] $G_1(\mathcal{V}_1, \mathcal{E}_1) \cup G_2(\mathcal{V}_2, \mathcal{E}_2) \equiv G_3(\mathcal{V}_3, \mathcal{E}_3)$, where $\mathcal{V}_3 = \mathcal{V}_1 \cup \mathcal{V}_2$ and $\mathcal{E}_3 = \mathcal{E}_1 \cup \mathcal{E}_2$. Note that this is *not* a multiset, there is only one copy of each node or edge.

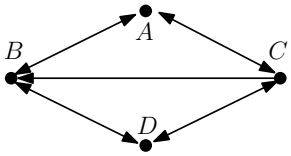
n-WAIT holds in $G^i[p]$ or not. Note that $G^i[p]$ may not equal to G, as node i may not receive messages from some other nodes due to asynchrony or failures. We say Condition n-WAIT holds in the local estimated graph $G^i[p](\mathcal{V}^i[p], \mathcal{E}^i[p])$ if **there exists** a set $F_i[p] \subseteq \mathcal{V}^i[p] - \{i\}$, where $|F_i[p]| \leq f$, such that $reach'_i(F_i[p]) \subseteq heard_i[p]$. Here, $reach'_i(F_i)$ is the set of nodes that have paths to node i in the subgraph induced by the nodes in $\mathcal{V}^i[p] - F_i[p]$ for $F_i[p] \subseteq \mathcal{V}^i[p] - \{i\}$ and $|F_i[p]| \leq f$.

Recall that N_i^- denotes the set of i's one-hop incoming neighbors. Given a set of nodes N and node i, we also use the notation $G_{N \Rightarrow i}$ to describe a directed graph consisting of nodes $N \cup \{i\}$ and set of directed edges from each node in N to i. Formally, $G_{N \Rightarrow i} = (N \cup \{i\}, E')$, where $E' = \{(j, i) \mid j \in N\}$.

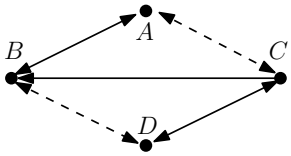
Correctness of Algorithm LWA. The key lemma to prove the correctness of Algorithm WA in [23] is to show that for any pair of nodes that have not crashed in phase p, they must receive a state value from at least one common node. In the full version [21], we show that Algorithm LWA achieves the same property. Intuitively, if Condition n-WAIT does not hold in the local estimated graph $G^i[p]$, then node i knows it can learn more states in phase p. Also, when Condition n-WAIT is satisfied in $G^i[p]$, there exists a scenario that node i cannot receive any more information; hence, it should not wait for any more message. This is why the Algorithm LWA allows each node to learn enough state values to achieve approximate consensus. We rely on this observation to prove the correctness in [21]. Algorithm LWA works on undirected graphs as well, as is shown in the full version [21].

5 Discussion

In asynchronous systems, the real time communication delay is arbitrary but finite. In a formal framework, it is common to assume that execution proceeds in rounds representing real time intervals, but the nodes do not have knowledge of the round index. To model the



(a) Graph G, i-CCA holds for any $i \in \{1, \ldots, 4\}$ and $f = 1$.



(b) Arbitrary delay in directed edges $(A, C), (C, A), (B, D), (D, B)$

Figure 2 Real time delay example.

worst-case real time delay in the execution of a system we can use the notion of *delay scenario* which is a description of the delays, incurring on the communication through all edges of the network. The delivery delay of a message sent over a channel e will be described by the number of rounds (amount of real time) that are needed for the delivery to be completed.

We first compare the real time performance of Algorithms k-LocWA for different values of k with respect to the real time delay. Specifically we show that there is a case where Algorithm LocWA terminates each phase in one round (one interval of real time), while it may take arbitrary number of rounds for Algorithm 2-LocWA to terminate phase 1.

▶ **Example 17.** Consider the graph of Figure 2a. For $f = 1$, it is easy to verify that Condition 1-CCA holds, which implies that Conditions i-CCA, for $i \in \{1, \ldots, n\}$ hold. Assume that the delivery of messages through *directed* edges $(A, C), (C, A), (B, D), (D, B)$ is delayed by d rounds while the communication in all the other edges is instant (one round). For ease of presentation assume that no node crashes. Then, in an execution of Algorithm LocWA, it is clear that every node i will finish phase t in time t. On the other hand, in an execution of Algorithm 2-LocWA, node D will only receive a message from C in one round, since (C, B) is a directed edge, and delay on edges (A, C) and (B, D) is d. in this case, D will not be able to decide before round d, the first round where Condition 2-WAIT will be satisfied. Specifically, for the first phase it will hold that $reach_D^2 \subseteq heard_D[1]$ only after round d since, if D considers $F_D = \{B\}$ as a possible corruption set, it has to wait for a message from A which will be propagated by C and setting $F_D = \{C\}$, it has to wait for a message from B. For similar reasons, the same holds for nodes A, C. Since d may be an arbitrary integer, there is a delay scenario where the ϵ-convergence time for Algorithm 2-LocWA is arbitrarily larger than the ϵ-convergence time of Algorithm LocWA.

Strong version of k-LocWA with respect to real time. As shown in the full version [21], the k-WAIT condition of k-LocWA algorithm can be strengthened such that, for $k' \geq k$ and any ϵ, Algorithm k'-LocWA will ϵ-converge faster than Algorithm k-LocWA. This can be achieved by condition strong k-WAIT: wait until $\bigvee_{i=1}^{k} (i\text{-WAIT}) = true$ for the first time.

References

1 EduardoA.P. Alchieri, AlyssonNeves Bessani, Joni Silva Fraga, and Fabíola Greve. Byzantine Consensus with Unknown Participants. In *Principles of Distributed Systems*, volume 5401 of *Lecture Notes in Computer Science*, pages 22–40. Springer Berlin Heidelberg, 2008. doi:10.1007/978-3-540-92221-6_4.

2 John Augustine, Gopal Pandurangan, and Peter Robinson. Fast Byzantine Agreement in Dynamic Networks. In *Proceedings of the 2013 ACM Symposium on Principles of*

Distributed Computing, PODC '13, pages 74–83, New York, NY, USA, 2013. ACM. doi:10.1145/2484239.2484275.

3 Piyush Bansal, Prasant Gopal, Anuj Gupta, Kannan Srinathan, and Pranav Kumar Vasishta. Byzantine agreement using partial authentication. In *Proceedings of the 25th international conference on Distributed computing*, DISC'11, pages 389–403, Berlin, Heidelberg, 2011. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=2075029.2075079.

4 Martin Biely, Peter Robinson, and Ulrich Schmid. Agreement in Directed Dynamic Networks. In Guy Even and Magnús M. Halldórsson, editors, *Structural Information and Communication Complexity*, pages 73–84, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

5 Martin Biely, Peter Robinson, Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. Gracefully Degrading Consensus and k-Set Agreement in Directed Dynamic Networks. In Ahmed Bouajjani and Hugues Fauconnier, editors, *Networked Systems*, pages 109–124, Cham, 2015. Springer International Publishing.

6 Martin Biely, Peter Robinson, Ulrich Schmid, Manfred Schwarz, and Kyrill Winkler. Gracefully degrading consensus and k-set agreement in directed dynamic networks. *Theoretical Computer Science*, 726:41–77, 2018. doi:10.1016/j.tcs.2018.02.019.

7 Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Approximate Consensus in Highly Dynamic Networks. *CoRR*, abs/1408.0620, 2014. URL: http://arxiv.org/abs/1408.0620, arXiv:1408.0620.

8 Bernadette Charron-Bost, Matthias Függer, and Thomas Nowak. Approximate Consensus in Highly Dynamic Networks: The Role of Averaging Algorithms. In *Proceedings, Part II, of the 42Nd International Colloquium on Automata, Languages, and Programming - Volume 9135*, ICALP 2015, pages 528–539, New York, NY, USA, 2015. Springer-Verlag New York, Inc. doi:10.1007/978-3-662-47666-6_42.

9 Étienne Coulouma and Emmanuel Godard. A Characterization of Dynamic Networks Where Consensus Is Solvable. In Thomas Moscibroda and Adele A. Rescigno, editors, *Structural Information and Communication Complexity*, pages 24–35, Cham, 2013. Springer International Publishing.

10 S. M. Dibaji, H. Ishii, and R. Tempo. Resilient Randomized Quantized Consensus. *IEEE Transactions on Automatic Control*, PP(99):1–1, 2017. doi:10.1109/TAC.2017.2771363.

11 Danny Dolev. The Byzantine Generals Strike Again. *Journal of Algorithms*, 3(1), March 1982.

12 Danny Dolev, Nancy A. Lynch, Shlomit S. Pinter, Eugene W. Stark, and William E. Weihl. Reaching approximate agreement in the presence of faults. *J. ACM*, 33:499–516, May 1986. doi:10.1145/5925.5931.

13 Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. In *Proceedings of the fourth annual ACM symposium on Principles of distributed computing*, PODC '85, pages 59–70, New York, NY, USA, 1985. ACM. doi:10.1145/323596.323602.

14 Heath LeBlanc, Haotian Zhang, Xenofon D. Koutsoukos, and Shreyas Sundaram. Resilient Asymptotic Consensus in Robust Networks. *IEEE Journal on Selected Areas in Communications*, 31(4), 2013. doi:10.1109/JSAC.2013.130413.

15 Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.

16 Alexandre Maurer, Sébastien Tixeuil, and Xavier Défago. Reliable Communication in a Dynamic Network in the Presence of Byzantine Faults. *CoRR*, abs/1402.0121, 2014. URL: http://arxiv.org/abs/1402.0121, arXiv:1402.0121.

17 Mikhail Nesterenko and Sébastien Tixeuil. Discovering Network Topology in the Presence of Byzantine Faults. In *Structural Information and Communication Complexity*, pages 212–226, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

18 Aris Pagourtzis, Giorgos Panagiotakos, and Dimitris Sakavalas. Reliable broadcast with respect to topology knowledge. *Distributed Computing*, 30(2):87–102, 2017. doi:10.1007/s00446-016-0279-6.

19 Aris Pagourtzis, Giorgos Panagiotakos, and Dimitris Sakavalas. Reliable Communication via Semilattice Properties of Partial Knowledge. In Ralf Klasing and Marc Zeitoun, editors, *Fundamentals of Computation Theory - 21st International Symposium, FCT 2017, Bordeaux, France, September 11-13, 2017, Proceedings*, volume 10472 of *Lecture Notes in Computer Science*, pages 367–380. Springer, 2017. doi:10.1007/978-3-662-55751-8_29.

20 M. Pease, R. Shostak, and L. Lamport. Reaching Agreement in the Presence of Faults. *J. ACM*, 27(2):228–234, April 1980. doi:10.1145/322186.322188.

21 Dimitris Sakavalas, Lewis Tseng, and Nitin H. Vaidya. Asynchronous Crash-Tolerant Approximate Consensus in Directed Graphs: Topology Knowledge. *CoRR*, abs/1803.04513, 2018. arXiv:1803.04513.

22 Lili Su and Nitin Vaidya. Reaching Approximate Byzantine Consensus with Multi-hop Communication. In Andrzej Pelc and Alexander A. Schwarzmann, editors, *Stabilization, Safety, and Security of Distributed Systems*, volume 9212 of *Lecture Notes in Computer Science*, pages 21–35. Springer International Publishing, 2015. doi:10.1007/978-3-319-21741-3_2.

23 Lewis Tseng and Nitin H. Vaidya. Fault-Tolerant Consensus in Directed Graphs. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC '15, pages 451–460, New York, NY, USA, 2015. ACM. doi:10.1145/2767386.2767399.

24 Lewis Tseng and Nitin H. Vaidya. Iterative approximate Byzantine consensus under a generalized fault model. In *In International Conference on Distributed Computing and Networking (ICDCN)*, January 2013.

25 Nitin H. Vaidya, Lewis Tseng, and Guanfeng Liang. Iterative Approximate Byzantine Consensus in Arbitrary Directed Graphs. In *Proceedings of the thirty-first annual ACM symposium on Principles of distributed computing*, PODC '12. ACM, 2012.

26 H. Zhang and S. Sundaram. Robustness of distributed algorithms to locally bounded adversaries. In *Proceedings of ACC 2012, the 31st American Control Conference*, 2012.