Synchronous t-Resilient Consensus in Arbitrary Graphs *

Armando Castañeda[1], Pierre Fraigniaud[2], Ami Paz[2], Sergio Rajsbaum[1], Matthieu Roy[1,3], and Corentin Travers[4]

[1] UNAM, Mexico — {armando.castaneda,rajsbaum}@im.unam.mx
[2] CNRS and Université de Paris, France — {pierref,amipaz}@irif.fr
[3] LAAS, CNRS, Toulouse, France — roy@laas.fr
[4] CNRS and University of Bordeaux, France — travers@labri.fr

Abstract. We study the number of rounds needed to solve consensus in a synchronous network G where at most t nodes may fail by crashing. This problem has been thoroughly studied when G is a complete graph, but very little is known when G is arbitrary. We define a notion of radius that considers all ways in which t nodes may crash, and present an algorithm that solves consensus in radius rounds. Then we derive a lower bound showing that our algorithm is optimal for vertex-transitive graphs, among oblivious algorithms.

Keywords: Crash failures · Consensus · Combinatorial topology · Distributed graph algorithms

1 Introduction

The problem. We consider a synchronous message-passing distributed system, where at most t out of n nodes may fail by crashing. The nodes communicate by sending messages to each other over the edges of an undirected graph G. In the *consensus* problem each node is given an input, and after some number of rounds produces an output, such that all outputs are the same and must be equal to one of the inputs.

One of the earliest and most well-know facts in distributed computing is that the number of rounds needed to solve consensus when G is the complete graph, K_n, is $t+1$. Namely, consensus can be solved in $t+1$ rounds, for $t < n$, and any algorithm requires this number of rounds in the worst case. The round complexity to solve consensus in K_n has been thoroughly studied, but not for graphs other than the complete graph.

1.1 Results

This paper studies the number of rounds needed to solve consensus, as a function of G and t. It presents two main contributions.

* Supported by ANR Project DESCARTES, INRIA Project GANG, UNAM-PAPIIT IA102417 and IN109917, and Fondation des Sciences Mathématiques de Paris

First, it shows that for any given $(t+1)$-vertex-connected graph G, it is possible to solve consensus tolerating t failures, in $\mathsf{radius}(G,t)$ rounds. Roughly, the *eccentricity of v against t failures*, $\mathsf{ecc}(v,t)$, is the smallest number of rounds needed for a node v to broadcast its input value, independently of the failure pattern (when and how nodes crash). Then, $\mathsf{radius}(G,t)$ is equal to the smallest $\mathsf{ecc}(v,t)$, over all nodes v. For example, $\mathsf{radius}(K_n,t) = t+1$ for the complete graph and $\mathsf{radius}(C_n,1) = n-1$ for the cycle. For the wheel, $\mathsf{radius}(W_n,2) = n-1$ and $\mathsf{radius}(W_n,1) = 1 + \lfloor (n-1)/2 \rfloor$.

Second, we present a corresponding lower bound, showing that our algorithm is optimal among oblivious algorithms, in any graph that is vertex-transitive. In an *oblivious* algorithm, the decision value of a node is based on the set of input values it has seen so far, and not on the particular failure pattern. Roughly speaking, a graph is *vertex-transitive* if it is highly symmetric. This is a large and well studied class of graphs (see, e.g., [18]).

The question of achieving consensus in a network prone to failures was intensively studied when the communication pattern is the *complete graph*. However, it seems difficult to obtain direct generalizations of these classical upper and lower bound techniques from a complete graph to a *general graph*. Instead, both our upper and lower bounds use novel ideas, that we discuss next.

Our upper bound techniques. In a classic algorithm to solve consensus on a complete graph, e.g. [29], nodes repeatedly send all the inputs they know, and at the end of round $t+1$, each node that has not crashed, decides the smallest input value among the values it has seen. The usual agreement argument is that among the $t+1$ rounds there must be at least one in which no node crashes. All nodes that are alive at the end of such a round have seen the same set of inputs, i.e., there is *common knowledge* [14] on a set of inputs. This argument holds only under the assumption that the graph is complete. We use a similar algorithm on an arbitrary graph, but apply a more fine-grained argument, of *information flow*, to prove its correctness and running time.

Given a node v and its $\mathsf{ecc}(v,t)$, we show that at the end of round $\mathsf{ecc}(v,t)$, either all alive nodes have received v's input, or none has. For the complete graph, $\mathsf{ecc}(v,t) = t+1$ for all nodes v, and indeed, for any node v, either all nodes have received the input of v by round $t+1$, or no node will ever receive it. This implies the correctness of the algorithm for the complete graph described above. Notice that the eccentricity is not less than $t+1$, because the adversary may create a *hidden path*, $v_1, \ldots, v_t$ such that $v_1 = v$ and each v_i, $1 \leq i \leq t-1$, fails in round i and sends a message to only v_{i+1} before failing.

We use this information flow perspective to derive simple consensus algorithms for arbitrary graphs. Each node repeatedly forwards all the pairs (v, in_v) it knows about, where in_v is the input value of node v. Then, an algorithm is specified by two functions: $R(G,t)$ which returns the the number of rounds to execute, and $D(G,t)$ which tells a node which value to decide, among the input values it has seen. After $R(G,t)$ rounds, the active nodes have the same view of the inputs of a carefully chosen *subset* of $t+1$ nodes, thus, after $R(G,t)$ rounds,

$D(G,t)$ can pick deterministically the input of one of these nodes. Remarkably, our lower bound shows that this is not necessarily the case after less rounds.

Our lower bound techniques. There are several lower bound proofs for the number of rounds to solve consensus under crash failures for the case when G is a complete graph. The classic $t+1$ lower bound proof style proceeds by a rather complex backward induction (a detailed description appears in [25]). Later on, simpler forward induction proofs were discovered [1, 26], following the classical bivalency arguments that were originally developed for proving the impossibility of solving consensus in asynchronous systems [17].

The aforementioned proofs hold for general graphs as well, namely, $t+1$ rounds is a lower bound for solving consensus on any graph G. However, in general graphs this bound is very weak, as it does not take into consideration the graph's structure. An obvious example is a cycle with $t=1$: our lower bound is $n-1$, while the standard approaches give a lower bound of 2 rounds.

Our lower bound technique is different from both the backward and the forward arguments. It is inspired by the topological techniques for distributed computing [20], though we do not use topology explicitly in the current paper. Our lower bound technique is similar to the connectivity analysis of the *protocol complex*, the structure of states at the end of executions of an algorithm after a certain number of rounds. However, instead of working with the protocol complex, we consider an *information flow* directed graph version based on failure patterns, without including input values. We prove that consensus is solvable by an oblivious algorithm if and only if all connected components of the information flow graph have a *dominating* vertex, namely, a vertex with an edge from it to any other vertex in its connected component. In [6], we study these information flow techniques and their relation to *set agreement* and *approximate agreement*.

The seminal paper [14] shows that, as soon as there is common knowledge of a *clean* round (where a node that crashes does not send any messages), it is also common knowledge that nodes have identical views of the initial configuration. As a consequence, any action that depends on the system's initial configuration can be carried out simultaneously in a consistent way by the set of active nodes at any round $k \geq t+1$, if it can be carried out at all. Our lower bound is larger than $t+1$ on general graphs, and hence shows how the round in which nodes have common knowledge of a subset of the input configuration is affected also by the structure of the graph.

1.2 Related work

Consensus in the failure-prone synchronous model has been thoroughly studied since the beginning of the distributed computing field in the late 1970's [34]. A variety of aspects have been considered, including the number of rounds (in great detail, including worst case, early deciding, simultaneous, unbeatability, etc.), number and size of messages, variants of consensus, in static and dynamic networks, and under various failure models. We only mention some of the most

relevant papers, among a vast literature, which even surveys e.g. [8, 29] and textbooks on the field cover only partially, e.g., [4, 25, 30].

For general graphs, since early on there has been an interest in characterizing the graphs where consensus is solvable, initially for Byzantine failures [11,12,16]. It was observed early on [24] that $t+1$ connectivity is necessary and an exponential algorithm was described. The algorithms for Byzantine settings also work in our model. However, they have not been optimized for the number of rounds, and furthermore, our setting requires only $t+1$ vertex-connectivity, while an algorithm tolerating Byzantine failures requires $n \geq 3t+1$, and vertex-connectivity at least $2t+1$ [12]. Very recently, consensus algorithms for general graphs were designed, for *local broadcast* Byzantine failures [22]. One algorithm works in the local broadcast model on a graph under the weakest requirements—minimum degree $2t$, and $(\lfloor 3t/2+1 \rfloor)$ vertex-connected; however, it has an exponential time complexity. A different consensus algorithm terminates in $3n$ rounds, but only assuming the graph is $2t$-connected. There has also been work in characterizing the *directed* graphs for which fault tolerant synchronous consensus is solvable, both under crash and under Byzantine failures [32, 33].

We are not aware of any previous lower bounds techniques for arbitrary graphs. The $t+1$ lower bound on the number of rounds to solve consensus in K_n was originally proved in [15] for Byzantine failures, and was later extended to the case were digital signatures can be used [11], and finally to crash failures (see, e.g., [19]).

Our lower bound technique is mainly inspired by the topological techniques for distributed computing [20], and more specifically by the topological structure of the executions of a synchronous algorithm after a certain number of rounds [21]. Indeed, the technique used for deriving our second algorithm is reminiscent of topological existential upper bounds proofs used in the past [3]. Hidden paths have played an important role in the design of *early-deciding* consensus algorithms in the complete graph [7].

Research on *dynamic networks* also characterizes families of networks for which consensus (or a variant of it) is solvable [9, 10, 27, 31, 35]. Interestingly, dynamic networks research and works on synchronous fault-tolerant consensus [32, 33] share the idea of picking a node as a source, and having all nodes deciding on the input of this source. In Theorem 3 we present an information flow characterization for consensus, in terms of such a source. Our notion of a core set (see Section 3.2) can be seen as a refinement of such notions, defined in order to optimize the number of rounds. Interestingly, [27] presents a topological solvability characterization of consensus using the *point set topology* techniques introduced in [2].

The line of work on *almost everywhere agreement* initiated in [5, 13], was motivated by the impossibility of tolerating t crashes when the network is not $t+1$ connected (these works also consider Byzantine failures). They present algorithms for networks where consensus is actually unsolvable due to weak connectivity.

2 Preliminaries

Model of Computation. We consider the standard synchronous message-passing model of computation where at most t nodes may fail by crashing. A set of $n \geq 2$ nodes V communicate through bidirectional channels E defining a graph $G = (V, E)$. In the remainder of the paper, we fix G and t, and assume $t < \kappa(G)$, the *vertex connectivity* of G, i.e., the minimum number of nodes whose deletion disconnects G.

An *execution* proceeds in a infinite sequence of synchronous rounds, starting in round 1. In every round, each node v first performs some local computation, then sends a message to each of its neighbors in G, denoted $N(v)$, and then receives the messages sent to it from $N(v)$ in that round. When a node crashes in round r, it fails to send its message to some of its neighbors in round r, and sends no message in subsequent rounds.

A *failure pattern* φ for G, t specifies, for each node that fails, in which round number it fails, and which messages it fails to send. It is a set of triples of the form (v, F_v, f_v), indicating that v crashes in round f_v, in which it does not send the messages to $\emptyset \neq F_v \subseteq N(v)$. Since at most t nodes can fail, $|\varphi| \leq t$, and since nodes do not recover from a failure, if $(v, F_v, f_v), (u, F_u, f_u) \in \varphi$ then $v \neq u$.

For an execution with failure pattern φ, the *faulty* nodes are those that appear in a triplet in φ; the others are the *correct* nodes. A node is *active* in round r in φ if it is correct, or if it fails in a round later than r. A node that crashes with $F_v = N(v)$ is said to crash *cleanly* in φ.

Consider any input assignment to the nodes, and a failure pattern φ. Our algorithms are of the following form. Initially, for each node v with input in_v, its *view* is $\{(v, in_v)\}$. In each round, each node v sends its *view* to $N(v)$, and at the end of the round it updates its *view* with the new input value-pairs it receives.

We say that u *hears from* v *in* φ, if in some round u receives a message containing the input of v. Similarly, we way that u hears from v *by round* r in φ if u receives a message with v's input in round r, or before. In other words, there is a *causal path* from u to v [23] in an infinite execution with failure pattern φ. Clearly, the existence of such a path depends on φ, but not on the input assignment. Thus, to analyze the structure of all possible failure patterns, we ignore the input values. This is what we do next, where we may identify φ with the infinite execution with that failure pattern.

Eccentricity and Radius in Failure Patterns. Let $\mathsf{dist}_G(u, v)$ denote the distance between nodes u and v in $G = (V, E)$. The *eccentricity* of a node $v \in V$ is defined as $\mathsf{ecc}_G(v) = \max_{u \in V} \mathsf{dist}_G(u, v)$. The *diameter* of a graph is defined as $\max_{v \in V} \mathsf{ecc}_G(v)$, and its *radius* as $\min_{v \in V} \mathsf{ecc}_G(v)$. We generalize the notions of eccentricity and radius to the synchronous t-resilient model.

In the following, failure patterns are denoted by lower case Greek letters $\varphi, \psi, \ldots$, and sets of failure patterns are denoted by upper case Greek letters $\Phi, \Psi, \ldots$. We denote by $\Phi^{(t)}_{\text{all}}$ the set of *all* failure patterns for G and t. The failure pattern in which no nodes crash is $\varphi_\varnothing$, and hence $\Phi^{(0)}_{\text{all}} = \{\varphi_\varnothing\}$.

Definition 1. *Given a node $v \in V$ and a failure pattern $\varphi \in \Phi_{\text{all}}^{(t)}$, the* eccentricity $\mathsf{ecc}_G(v,\varphi) \in \mathbb{N} \cup \{\infty\}$ *of v in φ is the minimum number of rounds required for all correct nodes to hear from v (i.e., there is causal path from v to every correct node), or ∞ if not all correct nodes hear from v. If $\mathsf{ecc}_G(v,\varphi) \in \mathbb{N}$, we say that v* floods *to the correct nodes in φ.*

Consider any φ. Notice that since G is at least $(t+1)$-connected, and at most t nodes crash, if a correct node u hears from v, then every correct node receives a message from v (because it can get from u to every correct node). We thus have the following claim.

Fact 1 *For every $v \in V$, and every $\varphi \in \Phi_{\text{all}}^{(t)}$, if $\mathsf{ecc}_G(v,\varphi) = \infty$ then* no correct node *hears from v in φ.*

Definition 2. *For $v \in V$ and $\Phi \subseteq \Phi_{\text{all}}^{(t)}$, such that there is at least one $\varphi \in \Phi$ with $\mathsf{ecc}_G(v,\varphi) \in \mathbb{N}$, let*

$$\mathsf{ecc}_G(v,\Phi) = \max\{\mathsf{ecc}_G(v,\varphi) \ : \ \varphi \in \Phi, \mathsf{ecc}_G(v,\varphi) \in \mathbb{N}\}.$$

Notice that there is at least one $\varphi \in \Phi$ with $\mathsf{ecc}_G(v,\varphi) \in \mathbb{N}$, for any Φ containing failure patterns where v is correct.

Lemma 1. *For $v \in V$ and $\varphi \in \Phi_{\text{all}}^{(t)}$, let A be the set of all active nodes in round $\mathsf{ecc}_G(v,\Phi_{\text{all}}^{(t)})$ under φ. Either all nodes in A hear from v by round $\mathsf{ecc}_G(v,\Phi_{\text{all}}^{(t)})$, or no node in A hears from v by round $\mathsf{ecc}_G(v,\Phi_{\text{all}}^{(t)})$ in φ.*

Proof. Let $\varphi' \in \Phi_{\text{all}}^{(t)}$ be the failure pattern identical to φ in the first $\mathsf{ecc}_G(v,\Phi_{\text{all}}^{(t)})$ rounds, but with all the nodes of A correct in φ'. Then, the nodes in A have the same view in both φ and φ' in round $\mathsf{ecc}_G(v,\Phi_{\text{all}}^{(t)})$.

If $\mathsf{ecc}_G(v,\varphi') \in \mathbb{N}$, by Definition 1, all nodes in A hear from v by time $\mathsf{ecc}_G(v,\varphi')$, which is at most $\mathsf{ecc}_G(v,\Phi_{\text{all}}^{(t)})$, by Definition 2. The same is true for φ, as φ and φ' are identical in the first $\mathsf{ecc}_G(v,\Phi_{\text{all}}^{(t)})$ rounds.

If $\mathsf{ecc}_G(v,\varphi') = \infty$, no node in A hears from v in φ', by Fact 1, and then no node in A hears from v by round $\mathsf{ecc}_G(v,\Phi_{\text{all}}^{(t)})$ in φ because φ and φ' are identical in the first $\mathsf{ecc}_G(v,\Phi_{\text{all}}^{(t)})$ rounds. □

Definition 3. *Let $\Phi \subseteq \Phi_{\text{all}}^{(t)}$ such that for every $v \in V$ there is at least one $\varphi \in \Phi$ with $\mathsf{ecc}_G(v,\varphi) \in \mathbb{N}$. The radius of G with respect to Φ is defined as $\mathsf{radius}(G,\Phi) = \min_{v \in V} \mathsf{ecc}_G(v,\Phi)$.*

For $t = 0$, our notion of eccentricity and radius coincides with the classical graph-theoretic definition, i.e., $\mathsf{ecc}_G(v,\Phi_{\text{all}}^{(0)}) = \mathsf{ecc}_G(v)$ and $\mathsf{radius}(G,\Phi_{\text{all}}^{(0)}) = \mathsf{radius}(G)$. Moreover, in the complete graph K_n, we have $\mathsf{radius}(K_n,\Phi_{\text{all}}^{(t)}) = t+1$, which together with Lemma 1 implies the correctness of the simple algorithm discussed in the Introduction.

3 Consensus Algorithms in Arbitrary Graphs

We consider the usual *consensus* problem in which each node starts with an input value, defined by the following properties. **Termination**: Every correct node decides a value; **Validity**: The decision of a node is equal to the input of some node; **Agreement**: The decisions of any pair of nodes are the same.

Oblivious algorithms. Recall that in our algorithms, a node resends to its neighbors the set of input values it has received, each one together with the name of the node that has the corresponding input value. Thus, to specify a consensus algorithm, we define a function $\mathsf{R}(G,t)$ that returns a round number, stating that all correct nodes decide in round $\mathsf{R}(G,t)$. Also, we define a *decision function* $\mathsf{D}(G,t)$ used by a node to select a consensus value from its view (possibly taking in consideration the names of the nodes that proposed this inputs). Namely, in a t-fault tolerant oblivious consensus algorithm for G, after $\mathsf{R}(G,t)$ rounds of communication (independently of the failure pattern or the input assignment), each node selects a value from its view, as specified by the function $\mathsf{D}(G,t)$. We stress that $\mathsf{R}(G,t)$ and $\mathsf{D}(G,t)$ are not computed by the nodes, they are given as part of the algorithm (alternatively, if the nodes "know" G and t, then they can compute these functions locally).

3.1 A naive algorithm

We describe algorithm $\mathsf{P}^{G,t}_{\mathsf{ecc}} = (\mathsf{R}_{\mathsf{ecc}}(G,t), \mathsf{D}_{\mathsf{ecc}}(G,t))$, based on a simple idea. Let us order the n vertices of G as $v_1, \ldots, v_n$, with

$$\mathsf{ecc}_G(v_i, \Phi^{(t)}_{\text{all}}) \leq \mathsf{ecc}_G(v_{i+1}, \Phi^{(t)}_{\text{all}}) \tag{1}$$

for $1 \leq i < n$. In particular, we have $\mathsf{radius}(G, \Phi^{(t)}_{\text{all}}) = \mathsf{ecc}_G(v_1, \Phi^{(t)}_{\text{all}})$.

Let $\mathsf{R}_{\mathsf{ecc}}(G,t) = \mathsf{ecc}_G(v_{t+1}, \Phi^{(t)}_{\text{all}})$, and $\mathsf{D}_{\mathsf{ecc}}(G,t)$ be the function that returns the input of the smallest[5] node among the nodes in $\{v_1, \ldots, v_{t+1}\}$.

Theorem 1. *Algorithm* $\mathsf{P}^{G,t}_{\mathsf{ecc}}$ *solves consensus in* $\mathsf{ecc}_G(v_{t+1}, \Phi^{(t)}_{\text{all}})$ *rounds.*

Proof. The algorithm satisfies termination as all correct nodes run $\mathsf{R}_{\mathsf{ecc}}(G,t) = \mathsf{ecc}_G(v_{t+1}, \Phi^{(t)}_{\text{all}})$ rounds. For validity, the definition of $\mathsf{ecc}_G(v_{t+1}, \Phi^{(t)}_{\text{all}})$ and Equation 1 imply that all nodes receive at least one input of a node in $\{v_1, \ldots, v_{t+1}\}$ by round $\mathsf{ecc}_G(v_{t+1}, \Phi^{(t)}_{\text{all}})$, in every $\varphi \in \Phi^{(t)}_{\text{all}}$. For agreement, consider any $\varphi \in \Phi^{(t)}_{\text{all}}$ and the set A of all nodes that are active in round $\mathsf{ecc}_G(v_{t+1}, \Phi^{(t)}_{\text{all}})$ in φ. Lemma 1 and Equation 1 imply that either all nodes in A have received v_i's input, $1 \leq i \leq t+1$, in round $\mathsf{ecc}_G(v_{t+1}, \Phi^{(t)}_{\text{all}})$ in φ, or none of them has received it in that round. Therefore, all nodes in A have the same view of the inputs of the nodes $v_1, \ldots, v_{t+1}$, hence $\mathsf{D}_{\mathsf{ecc}}(G,t)$ returns the same value to all of them. □

It is easy to come up with graphs for which this solution is not optimal, in terms of number of rounds.

[5] Assuming V is a totally ordered set.

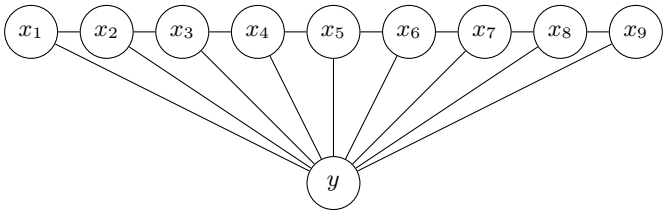

Fig. 1: A graph for which $\mathsf{P}_{\mathsf{ecc}}^{G,t}$ is not time optimal.

Lemma 2. *There is a graph G for which $\mathsf{P}_{\mathsf{ecc}}^{G,t}$ is not time optimal, with $t = 1$.*

Algorithm $\mathsf{P}_{\mathsf{ecc}}^{G,t}$ is not optimal in the graph in Figure 1 because $v_2 = x_4$ needs many rounds in order to broadcast its input, even when $v_1 = x_5$ crashes. Instead, y broadcasts very quickly when $v_1 = x_5$ crashes. As a consequence, y is a better choice for replacing x_5 whenever this latter node crashes. More generally, the sequence $v_1, \ldots, v_n$ defined in Eq. (1) is not adaptive. In the next subsection, we define an adaptive sequence, in which the performances of v_i are measured only for failure patterns in which $v_1, \ldots, v_{i-1}$ are prevented from flooding.

3.2 An adaptive-eccentricity based algorithm

The algorithm $\mathsf{P}_{\mathsf{ecc}}^{G,t}$ is based on a *core* set of nodes $\{v_1, \ldots, v_{t+1}\}$, consisting of the first $t+1$ nodes in order of ascending eccentricity. We show here that there is a more clever way of selecting a core set of $t+1$ nodes. The corresponding algorithm, $\mathsf{P}_{\mathsf{adapt}}^{G,t} = (\mathsf{R}_{\mathsf{adapt}}(G,t), \mathsf{D}_{\mathsf{adapt}}(G,t))$, is similar, except that, $\mathsf{R}_{\mathsf{adapt}}(G,t) = \mathsf{radius}(G, \Phi_{\mathrm{all}}^{(t)})$. As before, $\mathsf{D}_{\mathsf{adapt}}(G,t)$ returns the input of the smallest node among the core set, but now the core set is $\{s_1, \ldots, s_{t+1}\}$, as defined next.

The first node s_1 is the same v_1 as in $\mathsf{P}_{\mathsf{ecc}}^{G,t}$. To choose the i-th node, we consider all the un-chosen nodes, and their eccentricity *only among the failure patterns where the previously selected nodes have ∞ eccentricity*, and take the node that minimizes this quantity.

Formally, to define the core set of $t+1$ nodes, we construct a sequence of pairs (s_i, Φ_i), with $s_i \in V$, and $\Phi_i \subseteq \Phi_{\mathrm{all}}^{(t)}$, for $i = 1, \ldots, t+1$, inductively, as follows. For every node $v \in V$, let $\Phi_v^{\infty} = \{\varphi \in \Phi_{\mathrm{all}}^{(t)} : \mathsf{ecc}_G(v, \varphi) = \infty\}$ and $\Phi_v^{\mathbb{N}} = \{\varphi \in \Phi_{\mathrm{all}}^{(t)} : \mathsf{ecc}_G(v, \varphi) \in \mathbb{N}\}$. Let $\Phi_0 = \Phi_{\mathrm{all}}^{(t)}$, and, for $i = 1, \ldots, t+1$, let

$$\begin{cases} s_i = \arg\min_{v \in V \setminus \{s_1, \ldots, s_{i-1}\}} \mathsf{ecc}_G(v, \Phi_v^{\mathbb{N}} \cap \Phi_{i-1}), \\ \Phi_i = \Phi_{s_i}^{\infty} \cap \Phi_{i-1}, \end{cases} \tag{2}$$

where, for $i = 1$, we interpret $\{s_1, \ldots, s_{i-1}\}$ as the empty set. In other words, $\Phi_i = \Phi_{s_1}^{\infty} \cap \cdots \cap \Phi_{s_i}^{\infty}$, and also $\Phi_i = \Phi_{i-1} \setminus \Phi_{s_i}^{\mathbb{N}}$. Observe that, for every $i = 1, \ldots, t+1$, and every $v \in V \setminus \{s_1, \ldots, s_{i-1}\}$, $\Phi_v^{\mathbb{N}} \cap \Phi_{i-1}$ is not empty as it contains the failure pattern in which all nodes $s_1, \ldots, s_{i-1}$ crash cleanly at the first round, and no other node crashes. Also note that $\mathsf{ecc}_G(s_1, \Phi_{s_1}^{\mathbb{N}}) = \mathsf{radius}(G, \Phi_{\mathrm{all}}^{(t)})$.

For example, in K_n, we have $\mathsf{ecc}_{K_n}(s_i, \Phi^{\mathbb{N}}_{s_i}) = t - i + 2$ for $i = 1, \ldots, t+1$ whenever $t < n-1$. For $t = n-1$, we have $\mathsf{ecc}_{K_n}(s_i, \Phi^{\mathbb{N}}_{s_i}) = n - i$ for $i = 1, \ldots, n$. In the cycle C_n with $t = 1$, we have $\mathsf{ecc}_{C_n}(s_1, \Phi^{\mathbb{N}}_{s_1}) = n - 1$ and $\mathsf{ecc}_{C_n}(s_2, \Phi^{\mathbb{N}}_{s_2}) = \lfloor \frac{n-1}{2} \rfloor$. For the graph G in Figure 1, $s_1 = x_5$ and $s_2 = y$, $\mathsf{ecc}_G(s_1, \Phi^{\mathbb{N}}_{s_1}) = \mathsf{radius}(G, \Phi^{(1)}_{all}) = 4$, and $\mathsf{ecc}_G(s_2, \Phi^{\mathbb{N}}_{s_2}) = 1$.

The *core set* for G, t is $\{s_1, \ldots, s_{t+1}\}$, and the *core sequence* for G is the ordered sequence $(s_1, \ldots, s_{t+1})$. A crucial property of this sequence is that, while the sequence $(\mathsf{ecc}_G(v_i, \Phi^{\mathbb{N}}_{v_i}))_{1 \le i \le t+1}$ defined in Eq. (1) is non decreasing, and may even be increasing, the sequence $(\mathsf{ecc}_G(s_i, \Phi^{\mathbb{N}}_{s_i} \cap \Phi_{i-1}))_{1 \le i \le t+1}$ defined in Eq. (2) is non increasing, and is actually always decreasing. Intuitively, this is because the maximization in the computation of $\mathsf{ecc}_G(v, \Phi^{\mathbb{N}}_v \cap \Phi_i)$ for determining s_{i+1} is taken over the set $\Phi^{\mathbb{N}}_v \cap \Phi_i$ which is smaller than the set $\Phi^{\mathbb{N}}_v \cap \Phi_{i-1}$ used for the computation of s_i.

Lemma 3. *Consider the core sequence $(s_1, \ldots, s_{t+1})$ and the pairs defined in Eq. (2). Then, $\mathsf{ecc}_G(s_i, \Phi^{\mathbb{N}}_{s_i} \cap \Phi_{i-1})) > \mathsf{ecc}_G(s_{i+1}, \Phi^{\mathbb{N}}_{s_{i+1}} \cap \Phi_i))$, for $i \in \{1, \ldots, t\}$.*

The correctness proof of $\mathsf{P}^{G,t}_{\mathsf{adapt}}$ is very similar to that of $\mathsf{P}^{G,t}_{\mathsf{ecc}}$.

Theorem 2. *Algorithm $\mathsf{P}^{G,t}_{\mathsf{adapt}}$ solves consensus in $\mathsf{radius}(G, \Phi^{(t)}_{\mathrm{all}})$ rounds.*

Finally, observe that $\mathsf{P}^{\mathsf{G,t}}_{\mathsf{ecc}}$ performs in $\mathsf{ecc}_G(v_{t+1}, \Phi^{(t)}_{\mathrm{all}})$ rounds according to the notations of Eq (1), while $\mathsf{P}^{G,t}_{\mathsf{adapt}}$ performs in $\mathsf{radius}(G, \Phi^{(t)}_{\mathrm{all}}) = \mathsf{ecc}_G(v_1, \Phi^{(t)}_{\mathrm{all}})$ rounds according to the same notations.

4 The Lower Bound

In this section we show that $\mathsf{P}^{G,t}_{\mathsf{adapt}}$ is time optimal for vertex-transitive graphs, among oblivious algorithms. Recall that in an oblivious algorithm, the decision value of a node is based on the set of input values it has seen so far, and not on the particular failure pattern. Our algorithms $\mathsf{P}^{\mathsf{G,t}}_{\mathsf{ecc}}$ and $\mathsf{P}^{G,t}_{\mathsf{adapt}}$ are oblivious.

4.1 Information flow graph

Recall that the view of a node u in a given round r is the set of all pairs (v, in_v) such that u hears from v by round r. The vertices of the *information flow graph* have the form (v, view_v), meaning that node v has view view_v in round r, and there is a *directed* edge from (v, view_v) to (u, view_u) if and only if $(v, in_v) \in \mathsf{view}_u$, i.e., u hears from v by round r. Of course, these properties are conditioned by the actual failure pattern.

Consider a set of failure patterns $\Phi \subseteq \Phi^{(t)}_{\mathrm{all}}$. Let u be a node that is active in round r in φ, for some $r \ge 1$. Let $\mathsf{view}_G(u, \varphi, r)$ denote the view of u in round r in φ.

Definition 4. *The* information flow *graph in round r with respect to Φ is the directed graph $\mathbb{IF}_{G,\Phi,r}$:*

- $V(\mathbb{IF}_{G,\Phi,r}) = \{(u, \mathsf{view}_G(u,\varphi,r)) : u \in V \text{ is active in round } r \text{ in } \varphi \in \Phi\}$;
- $E(\mathbb{IF}_{G,\Phi,r}) = \big\{\big((u, \mathsf{view}_G(u,\varphi,r)), (v, \mathsf{view}_G(v,\varphi,r))\big) : u \in \mathsf{view}_G(v,\varphi,r)\big\}$.

Note that a node u may have the same view in two distinct $\varphi, \psi \in \Phi$ in round r, i.e., $\mathsf{view}_G(u,\varphi,r) = \mathsf{view}_G(u,\psi,r)$, in which case $(u, \mathsf{view}_G(u,\varphi,r))$ and $(u, \mathsf{view}_G(u,\psi,r))$ correspond to the same vertex of $\mathbb{IF}_{G,\Phi,r}$. Moreover, for any two distinct nodes u, v, we have $(u, \mathsf{view}_G(u,\varphi,r)) \neq (v, \mathsf{view}_G(v,\varphi,r))$, even if $\mathsf{view}_G(u,\varphi,r) = \mathsf{view}_G(v,\varphi,r)$.

The set $\mathsf{config}_G(\varphi,r) = \{(v, \mathsf{view}_G(v,\varphi,r)) : v \in V \text{ is active in round } r \text{ in } \varphi\}$ is called the r-round *configuration* for failure pattern φ. See Figure 2 for the information flow graph of the triangle K_3, with one failure, and one communication round.

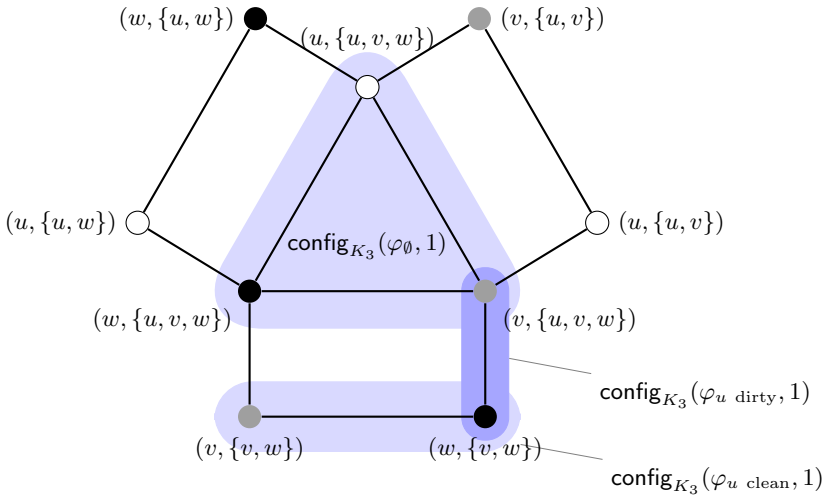

Fig. 2: $\mathbb{IF}_{K_3,\Phi^{(1)}_{\mathrm{all}},1}$, with the $\mathsf{config}_{K_3}(\varphi,1)$ sets marked, for some $\varphi \in \Phi^{(1)}_{\mathrm{all}}$; $\varphi_\emptyset$ denotes the failure pattern without failures, $\varphi_{u\ \mathrm{clean}}$ the failure patter where u fails cleanly in round 1 and $\varphi_{u\ \mathrm{dirty}}$ the failure patter where u fails in round 1 and sends a message only to v.

Lemma 4. *For every failure pattern $\varphi \in \Phi$, and every $r \geq 1$, the set $\mathsf{config}_G(\varphi,r)$ induces a connected subgraph of $\mathbb{IF}_{G,\Phi,r}$.*

Note that there is an edge from $(u, \mathsf{view}_G(u,\varphi,r))$ to $(v, \mathsf{view}_G(v,\psi,r))$ in $\mathbb{IF}_{G,\Phi,r}$ if and only if there exists $\varrho \in \Phi$ such that u and v are active in round r in ϱ, and $\mathsf{view}_G(u,\varphi,r) = \mathsf{view}_G(u,\varrho,r)$, $\mathsf{view}_G(v,\psi,r) = \mathsf{view}_G(v,\varrho,r)$ and $u \in \mathsf{view}_G(v,\varrho,r)$. Furthermore, if there are two failure patterns φ and ψ yielding the same view for a node v but two different views for a node u, then either the edges from the two views of u to the view of v both exist, or neither exists. This is specified in the following lemma.

Lemma 5. *Let $\varphi, \psi \in \Phi$ and $u, v \in V$ such that u and v are active in round r in both φ and ψ. If $\big((u, \mathsf{view}_G(u, \varphi, r)), (v, \mathsf{view}_G(v, \varphi, r))\big) \in E(\mathbb{IF}_{G,\Phi,r})$ and $\mathsf{view}_G(v, \varphi, r) = \mathsf{view}_G(v, \psi, r)$, then $\big((u, \mathsf{view}_G(u, \psi, r)), (v, \mathsf{view}_G(v, \psi, r))\big) \in E(\mathbb{IF}_{G,\Phi,r})$.*

4.2 The solvability characterization

The next result provides a solvability characterization for consensus by oblivious algorithms. In essence, it states that the number r of rounds should be large enough so that every connected component of $\mathbb{IF}_{G,\Phi,r}$ has a *dominating* node. A connected component of $\mathbb{IF}_{G,\Phi,r}$ is a connected component of the underlying, undirected graph of $\mathbb{IF}_{G,\Phi,r}$. We say that a node $v \in V$ of the graph G *dominates* a connected component C of $\mathbb{IF}_{G,\Phi,r}$, if the set $\{(v, \mathsf{view}_G(v, \varphi, r)) \ : \ \varphi \in \Phi\}$ dominates C. That is, for every $(w, \mathsf{view}_G(w, \varphi, r))$ in C, there is an arc from the vertex $(v, \mathsf{view}_G(v, \varphi, r))$ to $(w, \mathsf{view}_G(w, \varphi, r))$.

Theorem 3. *There is an oblivious algorithm solving consensus in r rounds under the set of failure patterns $\Phi \subseteq \Phi^{(t)}_{\text{all}}$ if and only if every connected component C of $\mathbb{IF}_{G,\Phi,r}$ has a dominating node in V.*

The two directions of the theorem are proved by the next two lemmas.

Lemma 6. *For any $\Phi \subseteq \Phi^{(t)}_{\text{all}}$, if every connected component C of $\mathbb{IF}_{G,\Phi,r}$ has a dominating node in V, then there is an oblivious algorithm solving consensus in r rounds under the set of failure patterns Φ.*

Proof. To solve consensus we only need to specify the decision function after r rounds of communication. For every connected component C of $\mathbb{IF}_{G,\Phi,r}$, pick a dominating node $v \in V$ of C. Let w be a node. The view view_w of w determines to which connected component C the vertex (w, view_w) belongs. The decision of w is the input value of the node v that dominates C.

Clearly, the algorithm satisfies termination and validity. For agreement, consider any $\varphi \in \Phi$. Let w and w' be two nodes that are active in round r in φ. By Lemma 4, the subgraph of $\mathbb{IF}_{G,\Phi,r}$ induced by $\mathsf{config}_G(\varphi, r)$ is connected. Therefore, $(w, \mathsf{view}(w, \varphi, r))$ and $(w', \mathsf{view}(w', \varphi, r))$ belongs to the same connected component C of $\mathbb{IF}_{G,\Phi,r}$, thus w and w' decide the input of the same node. □

Lemma 7. *For any $\Phi \subseteq \Phi^{(t)}_{\text{all}}$, if there is an oblivious algorithm solving consensus in r rounds under the set of failure patterns Φ, then every connected component C of $\mathbb{IF}_{G,\Phi,r}$ has a dominating node in V.*

Proof (Sketch of proof). We prove the contrapositive: if there is a connected component C of $\mathbb{IF}_{G,\Phi,r}$ with no dominating node in V, then there is no oblivious algorithm solving consensus in r rounds under Φ.

In the proof, we consider a standard connectivity argument a chain of failure patterns (executions). More specifically, we exhibit a sequence of failure patterns $\varphi_1, \ldots, \varphi_n$ such that (1) all nodes start with 0 in φ_1, (2) all nodes start with 1

in φ_n, and (3) there is a node v_i that has the same view in round r in both φ_i and φ_{i+1}. For proving (3), we exploit the fact that there is no node in V that dominates C, and thus it is possible to find a node that has the same view in both failure patterns, in round r. An algorithm cannot exist because the decision in φ_1 has to be 0, while the decision in φ_n has to be 1 and, then there are φ_i and φ_{i+1} with distinct decisions, which is a contradiction. □

4.3 Optimality of $\mathsf{P}^{G,t}_{\mathsf{adapt}}$ for symmetric graphs

To conclude, we use the characterization in Theorem 3 to show that $\mathsf{P}^{G,t}_{\mathsf{adapt}}$ is time optimal for vertex-transitive graphs, among oblivious algorithms.

An *automorphism* of G is a bijection $\pi : V \to V$ such that, for every two nodes u and v, $\{u, v\} \in E \iff \{\pi(u), \pi(v)\} \in E$. A graph $G = (V, E)$ is *vertex-transitive* if, for every two nodes u and v, there exists an automorphism π of G such that $\pi(u) = v$. For instance, the complete graphs K_n, the cycles C_n, the d-dimensional hypercubes Q_d, the d-dimensional toruses $C_{n_1} \times \cdots \times C_{n_d}$, the Kneser graphs $KG_{n,k}$, the Cayley graphs, etc., are all vertex-transitive. The wheel, composed of a cycle and a central node, is not vertex-transitive, since the center node has degree $n - 1$ while the cycle nodes have degree 3.

Theorem 4. *If G is vertex-transitive, then there is no oblivious algorithm that solves consensus in fewer than* $\mathsf{radius}(G, \Phi^{(t)}_{\mathrm{all}})$ *rounds.*

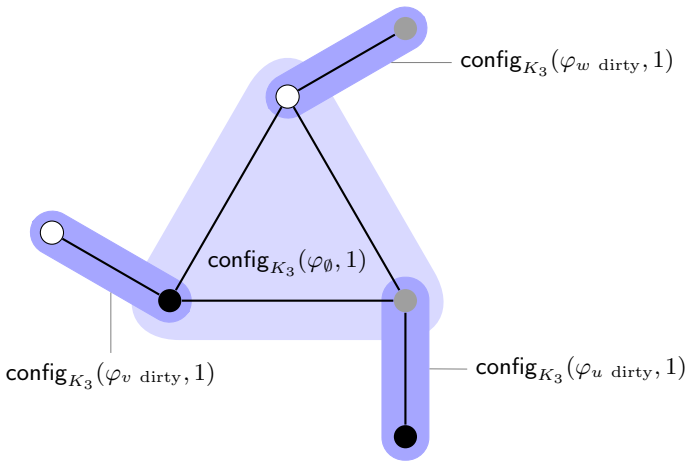

Fig. 3: The information flow graph $\mathbb{IF}_{K_3,\Phi,1}$ appearing in the proof of Theorem 4, for K_3 and the failure pattern Φ defined there. $\varphi_\emptyset$ denotes the failure patter without failures, while $\varphi_{x\ \mathrm{dirty}}$ denotes the failure patter where x fails in round 1, sending a message to only one node.

Proof (Sketch of proof). Clearly, the result holds if $\mathsf{radius}(G, \Phi^{(t)}_{\text{all}}) = 1$, as consensus is trivially not solvable in zero rounds in any graph with at least 2 nodes, even with no failures. So we assume now that $\mathsf{radius}(G, \Phi^{(t)}_{\text{all}}) \geq 2$.

In a vertex-transitive graph G, we have that for every $s \in V$, $\mathsf{radius}(G, \Phi^{(t)}_{\text{all}}) = \mathsf{ecc}_G(s, \Phi^{(t)}_{\text{all}})$. Therefore, for every $s \in V$, we can assign a failure pattern $\varphi_s \in \Phi^{(t)}_{\text{all}}$ such that $\mathsf{radius}(G, \Phi^{(t)}_{\text{all}}) = \mathsf{ecc}_G(s, \varphi_s)$. Let $\Phi = \{\varphi_s : s \in V\} \cup \{\varphi_\varnothing\}$. These execution sets $\mathsf{config}_G(\varphi, t)$ for $\varphi \in \Phi$ are depicted in Figure 3 for the case K_3 and $t = 1$. We will show a result stronger than the one expressed in the statement of the theorem. Namely, we show that no oblivious algorithms can solve consensus in a vertex-transitive graph G under Φ in less than $\mathsf{radius}(G, \Phi^{(t)}_{\text{all}})$ rounds. That is, even if the algorithm has only to deal with the $n + 1$ failure patterns in $\Phi \subseteq \Phi^{(t)}_{\text{all}}$, still consensus is not solvable in fewer than $\mathsf{radius}(G, \Phi^{(t)}_{\text{all}})$ rounds. To establish this result, let $R = \mathsf{radius}(G, \Phi^{(t)}_{\text{all}})$. Using Theorem 3, it is sufficient to prove that the following lemma:

Lemma 8. *The underlying graph of the information flow graph $\mathbb{IF}_{G,\Phi,R-1}$ is connected and has no dominating vertex.*

The theorem directly follows from the previous lemma and the characterization in Theorem 3. □

Theorem 5. *If G is vertex-transitive, $\mathsf{P}^{G,t}_{\mathsf{adapt}}$ is time optimal among oblivious algorithms.*

We conjecture that $\mathsf{P}^{G,t}_{\mathsf{adapt}}$ is time optimal for all graphs, among oblivious algorithms. This conjecture is grounded on the fact that Lemma 3 holds for all graphs, and not only for those that are vertex-transitive.

5 Conclusions

We have studied for the first time the number of rounds needed to solve fault-tolerant consensus in a crash prone synchronous network with arbitrary structure. We have defined a notion of *dynamic radius* of a graph G when t nodes may crash, which precisely determines the worst case number of rounds needed to solve oblivious consensus for vertex-transitive networks. The optimality of our algorithm was shown through a novel consensus solvability characterization in arbitrary networks, using the notion of *information flow*. A second consequence of the characterization is an abstract consensus algorithm that is optimal for all graphs. Our focus has been in the worst-case number of rounds. An interesting challenge would be to design early deciding algorithms (a problem that is well-studied in the case of the complete graph e.g. [8]).

An interesting future line of research is to study the case of non-oblivious algorithms (such algorithms have been considered in the past, e.g. [30]). Remarkably, for the case of the complete communication graph, there is no difference between these two types of algorithms: at the end of round $t + 1$, every pair of nodes have the same set of pairs (v, in_v) (formally, there is common knowledge on a set of inputs), hence decisions can be taken considering only this set.

Recall that, in our algorithms, $R(G,t)$ and $D(G,t)$ are hard-coded for a given G and t. It is worth exploring if our techniques are useful for the case where the graph G is not known to the nodes. Indeed, it is a challenge to combine fault-tolerant arguments with techniques of (failure-free) network computing [28]. Our results for $t = 0$ correspond to network computing. Yet, the case of $t > 0$ for arbitrary or evolving networks is an intriguing and complex research question.

References

1. Marcos Kawazoe Aguilera and Sam Toueg. A simple bivalency proof that t-resilient consensus requires t+1 rounds. *Information Processing Letters*, 71(3):155–158, 1999.
2. Bowen Alpern and Fred B. Schneider. Defining liveness. *Inf. Process. Lett.*, 21(4):181–185, 1985.
3. Hagit Attiya, Armando Castañeda, Maurice Herlihy, and Ami Paz. Bounds on the step and namespace complexity of renaming. *SIAM J. Comput.*, 48(1):1–32, 2019.
4. Hagit Attiya and Jenifer Welch. *Distributed computing: fundamentals, simulations, and advanced topics.* Wiley series on parallel and distributed computing. Wiley, 2004.
5. Piotr Berman and Juan A. Garay. Fast consensus in networks of bounded degree. *Distributed Computing*, 7(2):67–73, Dec 1993.
6. Armando Castañeda, Pierre Fraigniaud, Ami Paz, Sergio Rajsbaum, Matthieu Roy, and Corentin Travers. A topological perspective on distributed network algorithms. In *Structural Information and Communication Complexity - 26th International Colloquium, SIROCCO*, pages 3–18, 2019.
7. Armando Castañeda, Yannai A. Gonczarowski, and Yoram Moses. Unbeatable consensus. In *Distributed Computing - 28th International Symposium, DISC*, pages 91–106, 2014.
8. Armando Castañeda, Yoram Moses, Michel Raynal, and Matthieu Roy. Early decision and stopping in synchronous consensus: A predicate-based guided tour. In Amr El Abbadi and Benoît Garbinato, editors, *Networked Systems (NETYS), KNCS, vol. 10299*, pages 206–221, Cham, 2017. Springer.
9. Bernadette Charron-Bost and Shlomo Moran. Minmax algorithms for stabilizing consensus. *CoRR*, abs/1906.09073, 2019.
10. Étienne Coulouma, Emmanuel Godard, and Joseph G. Peters. A characterization of oblivious message adversaries for which consensus is solvable. *Theor. Comput. Sci.*, 584:80–90, 2015.
11. D. Dolev and H. Strong. Authenticated algorithms for byzantine agreement. *SIAM Journal on Computing*, 12(4):656–666, 1983.
12. Danny Dolev. The byzantine generals strike again. *Journal of Algorithms*, 3(1):14 – 30, 1982.
13. C Dwork, D Peleg, N Pippenger, and E Upfal. Fault tolerance in networks of bounded degree. In *Proceedings of the Eighteenth Annual ACM Symposium on Theory of Computing*, STOC '86, pages 370–379. ACM, 1986.
14. Cynthia Dwork and Yoram Moses. Knowledge and common knowledge in a byzantine environment: Crash failures. *Information and Computation*, 88(2):156 – 186, 1990.
15. Michael J. Fischer and Nancy A. Lynch. A lower bound for the time to assure interactive consistency. *Information Processing Letters*, 14(4):183 – 186, 1982.

16. Michael J. Fischer, Nancy A. Lynch, and Michael Merritt. Easy impossibility proofs for distributed consensus problems. *Distributed Computing*, 1(1):26–39, Mar 1986.
17. Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.
18. Chris Godsil and Gordon Royle. *Algebraic Graph Theory*. Graduate Texts in Mathematics, 207. Springer-Verlag, New York, 2001.
19. Vassos Hadzilacos. A lower bound for Byzantine agreement with fail–stop processors. Technical Report 21–83, Department of Computer Science, Harvard University, Cambridge, MA, July 1983.
20. Maurice Herlihy, Dmitry Kozlov, and Sergio Rajsbaum. *Distributed Computing Through Combinatorial Topology*. Morgan Kaufmann, 2013.
21. Maurice Herlihy, Sergio Rajsbaum, and Mark R. Tuttle. An axiomatic approach to computing the connectivity of synchronous and asynchronous systems. *Electr. Notes Theor. Comput. Sci.*, 230:79–102, 2009.
22. Muhammad Samir Khan, Syed Shalan Naqvi, and Nitin H. Vaidya. Exact byzantine consensus on undirected graphs under local broadcast model. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC*, pages 327–336, 2019.
23. Fabian Kuhn and Rotem Oshman. Dynamic networks: Models and algorithms. *SIGACT News*, 42(1):82–96, 2011.
24. Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982.
25. Nancy A. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1996.
26. Yoram Moses and Sergio Rajsbaum. A layered analysis of consensus. *SIAM J. Comput.*, 31(4):989–1021, 2002.
27. Thomas Nowak, Ulrich Schmid, and Kyrill Winkler. Topological characterization of consensus under general message adversaries. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC*, pages 218–227, 2019.
28. David Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM, Philadelphia, PA, 2000.
29. Michel Raynal. Consensus in synchronous systems: A concise guided tour. In *9th Pacific Rim International Symposium on Dependable Computing (PRDC)*, pages 221–228, 2002.
30. Michel Raynal. *Fault-Tolerant Message-Passing Distributed Systems - An Algorithmic Approach*. Springer, 2018.
31. Nicola Santoro and Peter Widmayer. Agreement in synchronous networks with ubiquitous faults. *Theor. Comput. Sci.*, 384(2-3):232–249, October 2007.
32. 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*, pages 451–460. ACM, 2015.
33. Lewis Tseng and Nitin H. Vaidya. A note on fault-tolerant consensus in directed networks. *SIGACT News*, 47(3):70–91, August 2016.
34. J. H. Wensley, L. Lamport, J. Goldberg, M. W. Green, K. N. Levitt, P. M. Melliar-Smith, R. E. Shostak, and C. B. Weinstock. Sift: Design and analysis of a fault-tolerant computer for aircraft control. In *Proceedings of the IEEE*, volume 66, pages 1240–1255, Oct 1978.
35. Kyrill Winkler and Ulrich Schmid. An overview of recent results for consensus in directed dynamic networks. *Bulletin of the European Association for Theoretical Computer Science (EATCS)*, 128:41–72, June 2019.