# Constant-space Localized Byzantine Consensus

Danny Dolev* and Ezra N. Hoch

School of Engineering and Computer Science,
The Hebrew University of Jerusalem, Israel,
{dolev,ezraho}@cs.huji.ac.il

**Abstract.** Adding *Byzantine* tolerance to large scale distributed systems is considered non-practical. The time, message and space requirements are very high. Recently, researches have investigated the *broadcast problem* in the presence of a $f_\ell$-local *Byzantine* adversary. The local adversary cannot control more than $f_\ell$ neighbors of any given node. This paper proves sufficient conditions as to when the synchronous *Byzantine consensus problem* can be solved in the presence of a $f_\ell$-local adversary. Moreover, we show that for a family of graphs, the *Byzantine* consensus problem can be solved using a relatively small number of messages, and with time complexity proportional to the diameter of the network. Specifically, for a family of bounded-degree graphs with logarithmic diameter, $O(\log n)$ time and $O(n \log n)$ messages. Furthermore, our proposed solution requires constant memory space at each node.

---



---

## 1 Introduction

Fault tolerance of a distributed system is highly desirable, and has been the subject of intensive research. *Byzantine* faults have been used to model the most general and severe failures. Classic *Byzantine*-tolerant research has concentrated on an "all mighty" adversary, which can choose up to $f$ "pawns" from the $n$ available nodes (usually, $f < \frac{n}{3}$). These *Byzantine* nodes have unlimited computational power and can behave arbitrarily, even colluding to "bring the system down". Much has been published relating to this model (for example, [11], [12], [2]), and many lower bounds have been proven.

One of the major drawbacks to the classic *Byzantine* adversarial model is its heavy performance cost. The running time required to reach agreement is linear in $f$ (which usually means it is also linear in $n$), and the message complexity is typically at least $O(n^2)$ (when $f = O(n)$, see [7]). This drawback stems from the "global" nature of the classic *Byzantine* consensus problem - i.e., the *Byzantine*

* Supported in part by ISF.

adversary has no restrictions on the spread of faulty nodes. Thus, the communication graph must have high connectivity (see [6] for exact bounds) - which leads to a high message load. Moreover, the global nature of the adversary requires every node to agree with every other node, leading to linear termination time and to a high out degree for each node (at least $2f + 1$ outgoing connections, see [6]). Such algorithms cannot scale; thus - as computer networks grow - it becomes infeasible to address global *Byzantine* adversaries.

To overcome this limitation, some research has assumed computationally-bounded adversaries, for example [5]. Alternatively, recent work has considered a $f_\ell$-local *Byzantine* adversary. The $f_\ell$-local adversary is restricted in the nodes it can control. For every node $p$, the adversary can control at most $f_\ell$ neighboring nodes of $p$. We call this stronger model "local" and the classic model "global". [10] has considered the *Byzantine* broadcast problem, which consists of all non-*Byzantine* nodes accepting a message sent by a given node. [13] classifies the graphs in which the broadcast problem can be solved, in the presence of a $f_\ell$-local adversary. In the current work we consider the *Byzantine* consensus problem, which consists of all non-*Byzantine* nodes agreeing on the same output value, which must be in the range of the input values of the non-*Byzantine* nodes. Sufficient conditions are given to classify graphs on which the problem can be solved, in the presence of a $f_\ell$-local adversary.

Solving the *Byzantine* consensus problem when facing a global adversary requires $O(n)$ memory space. When considering a local adversary, the memory space requirements can be reduced. This raises the question of possible tradeoff between fault tolerance and memory space requirement, which has been previously investigated in the area of self-stabilizing (see [8], [3], [9]). The current work provides a new tradeoff: for a family of graphs, consensus can be solved using constant memory space, provided that the *Byzantine* adversary is $f_\ell$-local.
**Contribution:** This work solves the *Byzantine* consensus problem in the local adversary model. For a large range of networks (a family of graphs with constant degree and logarithmic diameter), *Byzantine* consensus is solved within $O(\log n)$ rounds and with message complexity of $O(n \cdot \log n)$, while requiring $O(1)$ space at each node[1]. These results improve exponentially upon the classic setting which requires linear time, $O(n^2)$ messages, and (at least) linear space for reaching a consensus. We also present two additional results which are of special interest while using constant memory: first, we show a means of retaining identities in a local area of the network (see Section 4.1); second, we present a technique for waiting $\log n$ rounds using $O(1)$ memory space at each node (see Section 4.2).

## 2 Model and Problem Definition

Consider a synchronous distributed network of $n$ nodes, $\{p_0, \ldots, p_{n-1}\} = \mathcal{P}$, represented by an undirected graph $G = (E, V)$, where $V = \mathcal{P}$ and $(p, p') \in E$ if $p, p'$ are connected. Let $\Gamma(p)$ be the set of neighbors of $p$, including $p$. The

[1] Formally, we show that each node uses space polynomial in its degree; for constant degree it is $O(1)$.

communication network is assumed to be synchronous, and communication is done via message passing; each communication link is bi-directional.

*Byzantine* nodes have unlimited computational power and can communicate among themselves. The *Byzantine* adversary may control some portion of the nodes; however, the adversary is limited in that each node may not have more than $f_\ell$ *Byzantine* neighbors, where $f_\ell$ is a system-wide constant. Formally, a subset $S \subset \mathcal{P}$ is $f_\ell$-local if for any node $p \in \mathcal{P}$ it holds that $|S \bigcap \Gamma(p)| \leq f_\ell$. A *Byzantine* adversary is said to be $f_\ell$-local if (in any run) the set of *Byzantine* nodes is $f_\ell$-local.

### 2.1 Problem Definition

The following is a formal definition of the *Byzantine* consensus problem, followed by a memory-bounded variant of it.

**Definition 1.** *The* ***Byzantine*** **consensus problem** *consists of the following: Each node $p$ has an input value $v_p$ from an ordered set $\mathcal{V}$; all nodes agree on the same output $V \in \mathcal{V}$ within a finite number of rounds (agreement), such that $v_p \leq V \leq v_q$ for some correct nodes $p, q$ (validity).*

The goal of the current work is to show that for $f_\ell$-local adversaries, efficient deterministic solutions to the *Byzantine* consensus problem exist; efficient with respect to running time, message complexity and space complexity. A node $p$'s memory space depends solely on the local network topology - i.e. $p$'s degree or the degree of $p$'s neighbors.

Denote by $\ell$-MaxDeg($p$) the maximal degree of all nodes that are up to $\ell$ hops away from $p$. Notice that 0-MaxDeg($p$) $= |\Gamma(p)|$.

**Definition 2.** *The* **$\ell$-hop local *Byzantine* consensus problem** *is the* Byzantine *consensus problem with the additional constraint that each correct node $p$ can use memory space that is polynomially bounded by $\ell$-MaxDeg($p$).*

Informally, the above definition states that a node cannot "know" about all the nodes in the system, even though it can access all its local neighbors. That is, consensus must be reached with "local knowledge" only.

*Remark 1.* Definition 2 can be replaced by a requirement that the out degree of each node as well as the memory space are constant. We choose the above definition instead, as it generalizes the constant-space requirement.

## 3 *Byzantine* Consensus

In this section sufficient conditions for solving the *Byzantine* consensus problem for a $f_\ell$-local adversary are presented. First we discuss solving the *Byzantine* consensus problem in a network that is fully connected. We later give some definitions that allow us to specify sufficient conditions to ensure that *Byzantine* consensus can be solved in networks that are not fully connected.

### 3.1 Fully Connected *Byzantine* Consensus

We now define ByzCon, which solves the *Byzantine* consensus problem in a fully connected network. Take any *Byzantine* agreement[2] algorithm (for example, see [14]), and denote it by $BA$. ByzCon executes $n$ instance of $BA$, one for each node $p$'s input value, thus agreeing on the input value $v_p$. As a result, all correct nodes have an agreed vector of $n$ input values (notice that this vector may contain "$\perp$" values when $BA$ happens to return such a value for a *Byzantine* node). The output value of ByzCon is the median of the values of that vector, where "$\perp$" is considered as the lowest value.

*Claim.* ByzCon solves the *Byzantine* consensus problem in a fully-connected network.

**Definition 3.** *Given an algorithm $\mathcal{A}$ and a set of nodes $S \subset \mathcal{P}$,* Valid*($\mathcal{A}$, $S$) = true if the set $S$ upholds the connectivity requirements of $\mathcal{A}$.* Faulty*($\mathcal{A}$, $S$) denotes the maximal number of faulty nodes that $\mathcal{A}$ can "sustain" when executed on $S$ (for example,* Faulty$(BA, S) = \lfloor \frac{|S|-1}{3} \rfloor$*).*

*In a similar manner,* Time$(\mathcal{A}, S)$ *is the maximal number of rounds it takes $\mathcal{A}$ to terminate when executed on $S$, and* Msg$(\mathcal{A}, S)$ *is the maximal number of messages sent during the execution of $\mathcal{A}$ on $S$.*

Notice that for all $S$: Valid(ByzCon, $S$)= Valid($BA$, $S$), Faulty(ByzCon, $S$)= Faulty($BA$, $S$), Time(ByzCon, $S$) = Time($BA$, $S$) Msg(ByzCon, $S$) = $|S| \cdot$ Msg($BA$, $S$). That is, ByzCon's connectivity requirements, fault tolerance ratio and running time, are the same as in the *Byzantine* agreement algorithm of [14]; *i.e.,* ByzCon requires a fully connected graph among the nodes participating in ByzCon and it supports up to a third of them being *Byzantine*.

### 3.2 Sparsely Connected *Byzantine* Consensus

Consider a given algorithm that solves the *Byzantine* consensus problem when executed on a set of nodes $S$. ByzCon, defined in the previous section, requires $S$ to be fully-connected. The following discussion assumes ByzCon's existence and correctness.

The following definitions hold with respect to any $f_\ell$-local adversary.

**Definition 4.** *Given a subset $S \subset \mathcal{P}$, denote by $f_\ell$-Byz($S$) the maximal number of nodes from $S$ that may be* Byzantine *for a $f_\ell$-local adversary.*

**Definition 5.** *A non-empty subset $S$, $\emptyset \neq S \subset \mathcal{P}$, is a $f_\ell$-***decision group** *if* Valid*(*ByzCon*, $S$) = true and* Faulty(ByzCon, $S$) $\geq$ $f_\ell$-$Byz(S)$.

When considering ByzCon as constructed in Section 3.1, the above definition states that $S$ must be fully connected, and $|S| > 3 \cdot f_\ell$. Since 0-local adversaries are of no interest, the minimal size of $S$ satisfies, $|S| \geq 4$.

[2] *Byzantine* agreement is sometimes called *Byzantine* broadcast. This problem consists of a single leader broadcasting some value $v$ using point-to-point channels.

**Definition 6.** *A non-empty subset $S' \subseteq S$ of a $f_\ell$-decision group is a $f_\ell$-***common source** *if $f_\ell$-$Byz(S') + |S - S'| \leq \text{FAULTY}(\text{BYZCON}, S)$.*

*Claim.* If $S' \subseteq S$ is a common source and all correct nodes in $S'$ have the same initial value, $\nu$, then the output value of BYZCON when executed on $S$, is $\nu$.

*Proof.* Denote by $W$ the set of correct nodes in $S'$ and by $Y$ the set of all nodes in $S$ that are not in $W$; *i.e.,* $Y := S - W$. Since $S'$ is a common source, $|Y| \leq \text{FAULTY}(\text{BYZCON}, S)$. Assume by way of contradiction that all nodes in $W$ have the same initial value $\nu$, and the output of BYZCON (when executed on $S$) is not $\nu$; denote this execution by $\mathcal{R}$. If all the nodes in $Y$ are *Byzantine* and they simulate their part of $\mathcal{R}$, then it holds that all correct nodes in $S$ have the same initial value $\nu$, the number of *Byzantine* nodes is less than $\text{FAULTY}(\text{BYZCON}, S)$, and yet the output is not $\nu$. In other words, the "validity" of BYZCON does not hold. Therefore, if all nodes in $W = S - Y$ are correct and have the same initial value $\nu$, that must be the output value of BYZCON when executed on $S$. □

**Definition 7.** *Two subsets $S_1, S_2 \subset \mathcal{P}$ are $f_\ell$-***connected** *if $S_1, S_2$ are $f_\ell$-decision groups, and if $S_1 \cap S_2$ is a $f_\ell$-common source for both $S_1$ and $S_2$.*

**Definition 8.** *A list $C = S_1, S_2, \ldots, S_l$ is a $f_\ell$-***value-chain** *between $S_1$ and $S_l$ if the subsets $S_i, S_{i+1}$ are $f_\ell$-connected, for all $1 \leq i \leq l-1$. The* **length** *of the value-chain $C$ is $l-1$.*

**Definition 9.** *Let $\mathcal{G}$ be a set of $f_\ell$-decision groups,* i.e., *$\mathcal{G} \subseteq 2^{\mathcal{P}}$. $\mathcal{G}$ is an $f_\ell$-***entwined structure** *if for any two subsets $g, g' \in \mathcal{G}$ there is a $f_\ell$-value-chain $C = g_1, g_2, \ldots, g_l \in \mathcal{G}$ between $g, g'$. The* **distance** *between $g, g'$ is the minimal length among all $f_\ell$-value-chains between $g, g'$.*

**Definition 10.** *The* **diameter** *$\mathcal{D}$ of an $f_\ell$-entwined structure $\mathcal{G}$ is the maximal distance between any two $f_\ell$-decision groups $g, g' \in \mathcal{G}$.*

**Definition 11.** *An $f_\ell$-entwined structure $\mathcal{G}$ is said to* **cover** *graph $G$ if for any node $p$ in $G$ there is a subset $g \in \mathcal{G}$ such that $p \in g$. Formally: $\bigcup_{g \in \mathcal{G}} \{g\} = \mathcal{P}$.*

*Remark 2.* All entwined structures in the rest of this paper are assumed to cover their respective graphs. We will therefore sometimes say "an entwined structure $\mathcal{G}$" instead of "an entwined structure $\mathcal{G}$ that covers graph $G$".

Some of the above definitions were parameterized by $f_\ell$. When $f_\ell$ is clear from the context, we will remove the prefix $f_\ell$. (*i.e.,* "decision group" instead of "$f_\ell$-decision group").

**Definition 12.** *Let $G$ be a graph. Denote by $\Phi(G)$ the maximal value $f_\ell$ s.t. there is a $f_\ell$-entwined structure that covers $G$.*

The following theorem states the first result of the paper.

**Theorem 1.** *The* Byzantine *consensus problem can be solved on graph $G$ for any $\Phi(G)$-local adversary.*

Algorithm LocalByzCon /* *executed at node* $p$ */
/* $\mathcal{BC}_i$ *is an instance of* ByzCon */

**Initialization:**
1. **set** $v_p := p$'s initial value;
2. **set** $Output_p := \emptyset$; /* $\mathcal{G}_p := \{g \in \mathcal{G} | p \in g\}$ */
3. **for** all $g_i \in \mathcal{G}_p$ start executing $\mathcal{BC}_i$ with initial value $v_p$;

**For** $\Delta_{max} \cdot (2\mathcal{D}+1)$ **rounds:** /* $\Delta_{max} := max_i\{\text{Time}(\text{ByzCon}, g_i)\}$ */
1. **execute** a single round of each $\mathcal{BC}_i$ that $p$ participates in;
2. **for each** $\mathcal{BC}_i$ that terminated in the current round with value $V$:
   **set** $Output_p := Output_p \bigcup \{V\}$;
3. **for each** $\mathcal{BC}_i$ that terminated in the current round:
   **start** executing $\mathcal{BC}_i$ with initial value $\min\{Output_p\}$;

**Return value:**
**return** output as $\min\{Output_p\}$;

**Fig. 1.** Solving the *Byzantine* consensus problem for a $f_\ell$-local adversary.

Section 3.3 contains the algorithm LocalByzCon that given a $f_\ell$-entwined structure $\mathcal{G}$, solves the *Byzantine* consensus problem for any $f_\ell$-local adversary. Section 3.4 proves the correctness of LocalByzCon, thus completing the proof of Theorem 1.

### 3.3 Algorithm LocalByzCon

Figure 1 introduces the algorithm LocalByzCon that solves the *Byzantine* consensus problem for $f_\ell$-local adversaries, given an $f_\ell$-entwined structure $\mathcal{G}$. The main idea behind LocalByzCon is to execute ByzCon locally in each decision group. Each node takes the minimal agreement value among the decision groups it participated in. The fact that any two decision groups in $\mathcal{G}$ have a value-chain between them ensures that the minimal agreement value among the different invocations of ByzCon will propagate throughout $\mathcal{G}$. Since $\mathcal{G}$ covers $G$, all nodes will eventually receive the same minimal value.

Consider $\mathcal{G}$ to be a $f_\ell$-entwined structure, and $g_1, g_2, \ldots, g_m \in \mathcal{G}$ to be all the decision groups (of $\mathcal{P}$) in $\mathcal{G}$. For a node $p$, $\mathcal{G}_p$ is the set of all decision groups that $p$ is a member of; that is, $\mathcal{G}_p := \{g \in \mathcal{G} | p \in g\}$. Each node $p$ participates in repeated concurrent executions of ByzCon instances, where for each $g_i \in \mathcal{G}_p$, node $p$ will execute a ByzCon instance, and once that instance terminates $p$ will execute another instance, etc. For each $g_i \in \mathcal{G}_p$ denote by $\mathcal{BC}_i^1$ the first execution of ByzCon by all nodes in $g_i$; $\mathcal{BC}_i^2$ denotes the second instance executed by all nodes in $g_i$, and so on.

According to Definition 1 all nodes participating in ByzCon terminate within some finite time $\Delta$. Furthermore, each node can wait until $\Delta$ rounds elapse and terminate, thus achieving simultaneous termination. Therefore, in LocalByzCon, all nodes that participate in $\mathcal{BC}_i^j$ terminate it at the same round and start executing $\mathcal{BC}_i^{j+1}$ together at the following round.

### 3.4 Correctness Proof

For every $g_i \in \mathcal{G}$ denote by $r_i(1)$ the round at which the first consecutive instance of ByzCon executed on $g_i$ has terminated. Denote by $r(1) := \max\{r_i(1)\}$. Let $Output_p^r$ denote the value of $Output_p$ at the end of round $r$. Notice that $Output_p^r \subseteq Output_p^{r+1}$ for all correct $p$ and all $r$. Denote $Output^r := \bigcup\{Output_p^r\}$, the union of all $Output_p$ (for correct $p$) at the end of some round $r$. Using this notation, $Output^{r(1)}$ represents all the values in any $Output_p$ (for correct $p$) after at least one instance of ByzCon has terminated for each $g \in \mathcal{G}$. Consider some instance of ByzCon that terminates after round $r(1)$: it must be (at least) a second execution of that instance, thus all correct nodes that participated in it had their input values chosen from $Output^{r(1)}$. Thus, due to *validity*, the output value is in the range of $[\min\{Output^{r(1)}\}, \max\{Output^{r(1)}\}]$. Hence, we conclude that $\min\{Output^r\} \geq \min\{Output^{r(1)}\}$ for any $r \geq r(1)$. Denote by $v_{min} := \min\{Output^{r(1)}\}$; clearly no correct node $p$ will hold a lower value (in $Output_p$) for any $r \geq r(1)$.

**Lemma 1.** *If a correct node $p$ has $\min\{Output_p^r\} = v_{min}$ then it will never have a lower value in $Output_p$ for any round $r' \geq r$.*

**Lemma 2.** *At round $r(1)$, there exists $g_i \in \mathcal{G}$ such that for every correct node $p \in g_i$ it holds that $\min\{Output_p\} = v_{min}$.*

*Proof.* By definition, $v_{min} \in Output_p^{r(1)}$. Thus, $v_{min}$ was the output of some $\mathcal{BC}_i$ instance (on decision group $g_i$) at some round $r \leq r(1)$. Consider the nodes in $g_i$, they have all added $v_{min}$ to their $Output_p$. Thus, $v_{min} \in Output_p^{r(1)}$ and by definition it is the lowest value in $Output_p$ at round $r(1)$. Thus, at round $r(1)$, all correct nodes in $g_i$ have $\min\{Output_p\} = v_{min}$. □

**Lemma 3.** *If* LocalByzCon *has been executed for at least $\Delta_{max} \cdot (2\mathcal{D} + 1)$ rounds, then all correct nodes have $\min\{Output_p\} = v_{min}$.*

*Proof.* Divide the execution of LocalByzCon into "stages" of $\Delta_{max}$ rounds each. Clearly there are at least $2\mathcal{D} + 1$ stages, and in each stage each $\mathcal{BC}_i$ is started at least once. From the above lemma, for some decision group $g_i$, all correct nodes $p \in g_i$ have $\min\{Output_p\} = v_{min}$ at the end of the first stage.

Let $g'$ be some decision group, and let $g_i = g_1, g_2, \ldots, g_l = g'$ be a value-chain between $g_i$ and $g'$; there exists such a value-chain because $\mathcal{G}$ is an entwined structure, and its length is $\leq \mathcal{D}$.

Consider the second stage. Since $g_1, g_2$ are connected, and since all nodes in $g_1$ have the same initial value ($v_{min}$) during the entire stage 2, then in $g_1 \cap g_2$ all nodes have $v_{min}$ as their initial value during stage 2. Since $g_1 \cap g_2$ is a common source of $g_2$, it holds that instance $\mathcal{BC}_2$ that is started in stage 2 is executed with all nodes in $g_1 \cap g_2$ having initial value $v_{min}$. Thus, when $\mathcal{BC}_2$ terminates (no later than the end of stage 3), it terminates with the value $v_{min}$, thus all nodes in $g_2$ also have $v_{min} \in Output_p$. By Lemma 1, all nodes in $g_2$ choose $v_{min}$ as their initial value for any instance of ByzCon started during stage 4 and above.

Repeating this line of proof leads to the conclusion that after an additional $2\mathcal{D}$ stages all correct nodes in $g'$ have $v_{min} \in Output_p$.

Since any decision group $g'$ has a value-chain of no more than $\mathcal{D}$ length to $g_i$, we have that after $2\mathcal{D}+1$ stages, all correct nodes in all decision groups have $v_{min} \in Output_p$. Since $\mathcal{G}$ covers $G$, each correct node is a member of some decision group, thus all correct nodes in $G$ have $v_{min} \in Output_p$. Since $v_{min}$ is the lowest possible value, $\min\{Output_p\} = v_{min}$ for all $p \in \mathcal{P}$. □

*Remark 3.* Consider the value-chain $g_1, g_2, \ldots, g_l$ in the proof above. The proof bounds the time (in rounds) it takes $v_{min}$ to "propagate" from $g_1$ to $g_l$. The given bound $(2 \cdot l \cdot \Delta_{max})$ is not tight. In fact, instead of assuming $2 \cdot \Delta_{max}$ rounds for each "hop along the chain", one can accumulate $2 \cdot \textsc{Time}(\textsc{ByzCon}, g_i)$ when moving from $g_{i-1}$ to $g_i$. Thus, define $\textsc{Time}_{dist}(g, g')$ to be the shortest such sum on any value-chain between $g, g'$, and $\mathcal{D}_{\textsc{time}}$ as the maximal $\textsc{Time}_{dist}(g, g')$ on any $g, g' \in \mathcal{G}$; and we can conclude that it is enough to run LocalByzCon for $\mathcal{D}_{\textsc{time}}$ rounds. Notice that this analysis it tight up to a factor of 2.

**Lemma 4.** *Given an $f_\ell$-entwined structure $\mathcal{G}$ that covers $G$,* LocalByzCon *solves the* Byzantine *consensus problem, for a $f_\ell$-local adversary.*

*Proof.* From the above lemmas, after $\Delta_{max} \cdot (2\mathcal{D}+1)$ rounds, all correct nodes terminate with the same value, $v_{min}$. Notice that $v_{min}$ is the output of some ByzCon instance. Thus, there are two correct nodes $p, q$ such that $v_p \leq v_{min} \leq v_q$. Therefore, both "agreement" and "validity" hold. □

### 3.5 Complexity Analysis

The time complexity of LocalByzCon is $(2\mathcal{D}+1) \cdot \Delta_{max}$. In other words, let $g_{max}$ be the largest decision group in $\mathcal{G}$, that is $g_{max} := argmax_{g_i \in \mathcal{G}}\{|g_i|\}$; using this terminology we have that, $\textsc{Time}(\textsc{LocalByzCon}, \mathcal{P}) = (2\mathcal{D}+1) \cdot O(g_{max})$.

Similarly, the message complexity of LocalByzCon per round is the sum of all messages of all ByzCon instances each round, which is bounded by $\sum_{g_i} |g_i|^3 \leq |\mathcal{G}| \cdot |g_{max}|^3$. Thus, $\textsc{Msg}(\textsc{LocalByzCon}, \mathcal{P}) \leq (2\mathcal{D}+1) \cdot |\mathcal{G}| \cdot O(|g_{max}|^4)$ (messages per round times rounds).

## 4 Constant-space *Byzantine* Consensus

Section 3 proves a sufficient condition for solving the *Byzantine* consensus problem on a given graph $G$ for a $f_\ell$-local adversary. The current section gives a sufficient condition for solving the $\ell$-hop local *Byzantine* consensus problem.

**Definition 13.** *An $f_\ell$-entwined structure $\mathcal{G}$ is called* **$\ell$-hop local** *if for every $g \in \mathcal{G}$, $\ell$ bounds the distance between any two nodes in $g$.*

For ByzCon constructed in Section 3.1, any entwined structure $\mathcal{G}$ is 1-hop local, since for every decision group $g \in \mathcal{G}$, it holds that $\textsc{Valid}(\textsc{ByzCon}, g) = true$,

thus $g$ is fully connected. However, the following discussion holds for any algorithm that solves the *Byzantine* consensus problem, even if it does not require decision groups to be fully connected.

**Definition 14.** *An $f_\ell$-entwined structure $\mathcal{G}$ is called* $\ell$**-lightweight** *if $\mathcal{G}$ is $\ell$-hop local and for every $p \in \mathcal{P}$, $|\mathcal{G}_p|$ and $|g|$ (for all $g \in \mathcal{G}_p$) are polynomial in $|\Gamma(p)|$.*

**Definition 15.** *Let $G$ be a graph. Denote by $\ell$-$\Psi(G)$ the maximal value $f_\ell$ s.t. there is a $f_\ell$-entwined structure that is $\ell$-lightweight and covers $G$.*

The following theorem states the second contribution of this paper.

**Theorem 2.** *The $\ell$-hop local* Byzantine *consensus problem can be solved on graph $G$ for any $[\ell$-$\Psi(G)]$-local adversary.*

By Theorem 1, any $f_\ell$-entwined structure $\mathcal{G}$ that covers graph $G$ can be used to solve the *Byzantine* consensus problem on $G$, for a $f_\ell$-local adversary. To prove Theorem 2 we show that when LocalByzCon is executed using an $\ell$-lightweight $f_\ell$-entwined structure, each node $p$'s space requirements are polynomial in $\ell$-MaxDeg$(p)$. There are 3 points to consider: first, we show that the memory footprint of the different ByzCon instances that $p$ participates in is "small". Second, ByzCon assumes unique identifiers, which usually require $\log n$ space. We need to create identifiers that are locally unique (thus requiring space that is independent of $n$), such that ByzCon can be executed properly on each decision group. Third, the main loop of LocalByzCon requires to count at least up to $\mathcal{D}$, which requires $\log \mathcal{D}$ bits, possibly requiring space that is dependent on $n$.

The second point is discussed in Section 4.1 and the third in Section 4.2. To solve the first point notice that $\mathcal{G}$ is $\ell$-lightweight, thus each node participates in no more than $poly(|\Gamma(p)|)$ ByzCon instances concurrently, and each such instance contains $poly(|\Gamma(p)|)$ nodes. Assuming that the identifiers used by ByzCon require at most $polylog(\ell$-MaxDeg$(p))$ bits (see Section 4.1), $p$ requires at most $poly(\ell$-MaxDeg$(p))$ space to execute the ByzCon instances of LocalByzCon.

### 4.1 Locally Unique Identifiers

Consider the algorithm LocalByzCon in Figure 1 and an $\ell$-lightweight entwined structure $\mathcal{G}$. Different nodes communicate only within decision groups, *i.e.,* node $p$ sends or receives messages from node $q$ only if $p, q \in g$ for some $g \in \mathcal{G}$. Thus, the identifiers used can be locally unique.

To achieve this goal, node $p$ numbers each node $q$ in each decision group $g \in \mathcal{G}_p$, sequentially. Notice that the same node $q$ might "receive" different numbers in different decision groups that $p$ participates in. Thus, we can define Num$(p, g, q)$ to be the number $p$ assigns to $q$ for the decision group $g \in \mathcal{G}_p$. Notice that for an $\ell$-lightweight entwined structure, Num$(p, *, *)$ requires polynomial space in $|\Gamma(p)|$. Each node $z$ holds Num$(p, g, *)$ for all $g \in \mathcal{G}_z \bigcap \mathcal{G}_p$, along with a

mapping between $\text{NUM}(p, g, q)$ and $\text{NUM}(z, g, q)$. The memory footprint of this mapping is again polynomial in $|\Gamma(z)|$ (for $\ell$-lightweight entwined structures).

In addition, each node $p$ numbers the decision groups it is a member of: let $\text{INDX}(p, g)$ be the "number" of $g \in \mathcal{G}_p$ according to $p$'s numbering. Any node $z$ (such that $\mathcal{G}_p \bigcap \mathcal{G}_z \neq \phi$) holds a mapping between $\text{INDX}(p, g)$ and $\text{INDX}(z, g)$, for all $g \in \mathcal{G}_p \bigcap \mathcal{G}_z$. Notice that $\text{INDX}(p, *)$ is polynomial in $|\Gamma(p)|$, and the mapping requires memory space of size polynomial in $max\{|\Gamma(p)|, |\Gamma(z)|\}$. For $\ell$-lightweight entwined structures, the distance between $p$ and $z$ is $\leq \ell$, thus $max\{|\Gamma(p)|, |\Gamma(z)|\} \leq \ell\text{-MAXDEG}(p)$, resulting in a memory space footprint (of all the above structures) that is polynomial in $\ell\text{-MAXDEG}(p)$ for any node $p$.

When node $p$ wants to send node $q$'s identifier to $z$ regarding decision group $g$ (notice that $p, q, z \in g$ and $g \in \mathcal{G}_p, \mathcal{G}_q, \mathcal{G}_z$), it sends "$(\text{NUM}(p, g, q), \text{INDX}(p, g))$" and node $z$ uses its mapping to calculate $\text{INDX}(z, g)$, from which node $z$ can discern what $g$ is, and use its NUM mapping to calculate $\text{NUM}(z, g, q)$. Therefore, nodes can identify any node in their decision groups and can communicate these identities among themselves. Thus, "identities" can be uniquely used locally, with a low memory footprint. *i.e.,* the required memory space is polynomial in $\ell\text{-MAXDEG}(p)$. Notice that the above structures are constructed before executing LOCALBYZCON, once the system designer knows the structure of $\mathcal{G}$.

### 4.2 Memory-efficient Termination Detection

LOCALBYZCON as given in Figure 1 loops for $\Delta_{max} \cdot (2\mathcal{D}+1)$ rounds. Therefore, for $\mathcal{D}$ that depends on $n$, the counter of the loop will require too much memory. From the analysis of the execution of LOCALBYZCON it is clear that any node terminating after it ran for more than $\Delta_{max} \cdot (2\mathcal{D} + 1)$ rounds, terminates with the same value. Thus, it is only important that all nodes eventually terminate, and that they do so after at least $\Delta_{max} \cdot (2\mathcal{D} + 1)$ rounds; how can this be done without counting rounds? The following is an example of a solution requiring constant memory, using a simpler model.

Consider a synchronous network without any *Byzantine* nodes, where each node $p$ has $poly(|\Gamma(p)|)$ memory. Given that $\mathcal{D}$ (the diameter of the network) is not constant, how can one count until $\mathcal{D}$ in such a network? Mark two nodes $u, v$ as "special" nodes, such that the distance between $u$ and $v$ is $\mathcal{D}$ (clearly there exist $u, v$ that satisfy this condition). When the algorithm starts, $u$ floods the network with a *"start"* message. When $v$ receives this message, it floods the network with an *"end"* message. When any node receives the *"end"* message, it knows that at least $\mathcal{D}$ rounds have passed, and it can therefore terminate.

Using the above example, we come back to our setting of entwined structures and $f_\ell$-local *Byzantine* adversaries: consider two "special" decision groups $g_1, g_2$ from $\mathcal{G}$, such that the $\text{TIME}_{dist}$ between $g_1$ and $g_2$ is $\mathcal{D}_{\text{TIME}}$ (see Remark 3). Each node $p$, in addition to its initial value $v_p$, has two more initial values $v_p^1, v_p^2$ which are both set to "1". All nodes in $g_1$ set $v_p^1$ :="0". Instead of executing a single LOCALBYZCON, each node executes 3 copies of LOCALBYZCON: one on $v_p$, one on $v_p^1$ (denoted $\text{LOCALBYZCON}_1$) and one on $v_p^2$ (denoted $\text{LOCALBYZCON}_2$). Only nodes in $g_2$ perform the following rule: once $g_2$'s output in $\text{LOCALBYZCON}_1$

is "0", set $v_p^2 :=$ "0". Lastly, all nodes terminate one repetition after $Output_p$ of LocalByzCon$_2$ contains "0".

The analysis of this addition is simple: the value "0" in LocalByzCon$_1$ propagates throughout the network until it reaches $g_2$. Once it reaches $g_2$, the value "0" of LocalByzCon$_2$ propagates throughout the network, causing all nodes to terminate. Notice that before $g_2$ updates its input value of LocalByzCon$_2$, no correct node will have a value of "0" for LocalByzCon$_2$. Lastly, notice that at least $\frac{1}{2}\mathcal{D}_{\text{TIME}}$ rounds must pass before $g_2$ changes the input value to the third LocalByzCon (see Remark 3). Thus, if the second and third LocalByzCon are executed "at half speed" (every round, the nodes wait one round), then all correct nodes terminate not before $\mathcal{D}_{\text{TIME}}$ rounds pass, and no later than $2\mathcal{D}_{\text{TIME}}$ rounds.

The above schema requires the same memory space as the "original" LocalByzCon (i.e. independent of $n$), up to a constant factor, while providing termination detection, as required.

The decision groups $g_1, g_2$ must be selected prior to LocalByzCon's execution. An alternative option is to select $g_1$ using some leader election algorithm, and then use an MST algorithm to find $g_2$. However, in addition to *Byzantine* tolerance, these algorithms' memory requirements must not depend on $n$, which means that global identifiers cannot be used. There is a plethora of research regarding leader election / spanning trees and their relation to memory space (see [15], [3], [4], [1]). However, as far as we know, there are no lower or upper bounds regarding the exact model this work operates in (e.g. local identities, but no global identities). Thus, it is an open question whether it is required to choose $g_1, g_2$ a priori, or if they can be chosen at runtime.

### 4.3 Complexity Analysis

Consider graph $G$ with maximal degree $d_{max}$, and an $\ell$-lightweight entwined structure $\mathcal{G}$ (with diameter $\mathcal{D}$) that covers $G$. Denote $g_{max} := argmax_{g_i \in \mathcal{G}}\{|g_i|\}$, since $\mathcal{G}$ is $\ell$-lightweight, $g_{max}$ is polynomial in $d_{max}$, and $|\mathcal{G}| = n \cdot poly(d_{max})$. Therefore, by Section 3.5, LocalByzCon's time complexity is $O(\mathcal{D}) \cdot poly(d_{max})$, and its message complexity is $O(\mathcal{D}) \cdot n \cdot poly(d_{max})$.

From the above, if $G$'s maximal degree is independent of $n$, and if $\mathcal{D} = O(\log n)$, then the running time of LocalByzCon is $O(\log n)$ and its message complexity is $O(n \log n)$, while using $O(1)$ space. (Section 5 shows entwined structures that reach these values). This achieves an exponential improvement on the time and message complexity of *Byzantine* consensus in the "global" model, which are $O(n)$ and $O(n^2)$ respectively. [3]

## 5 Constructing 1-lightweight Entwined Structures

In this section we present a family of graphs for which 1-lightweight entwined structures exist. The family of graphs is given in a constructive way: for each

[3] In fact, known algorithms that use the transformation in [6] to operate in not-fully-connected graphs might even require $O(n^3)$ messages.

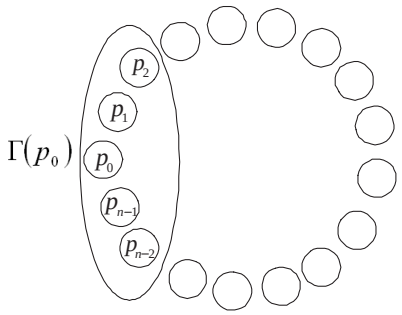


**Fig. 2.** $p_0$'s neighbors in an extended ring topology of order 2.

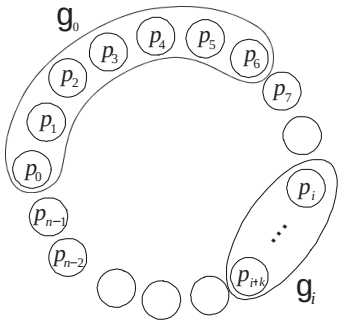


**Fig. 3.** $g_0$ for $f_\ell = 1$ and $k = 3(f_\ell + 1) = 6$.

graph $G' = (V', E')$ (where $|V'| = n'$) and for any value of $f_\ell$ ($f_\ell << n'$) we construct a graph $G = (V, E)$ ($|V| = n$) with a 1-lightweight $f_\ell$-entwined structure $\mathcal{G}$. Our construction achieves $|\mathcal{G}| = O(n)$ and $|g|$ is small for all $g \in \mathcal{G}$, thus ensuring that $\mathcal{G}$ is indeed 1-lightweight. Furthermore, $G$'s diameter and maximal degree are a function of those of $G'$s, thus ensuring that for $G'$ with bounded degree and logarithmic diameter, $G$ has similar properties.

First we show how to construct an entwined structure, given a graph with "ring topology". Then we show how "to combine" two graphs with ring topologies. Lastly, for every graph $G'$, we create a graph $G$ that "blows up" each node $p$ in $G'$ to be a ring (containing $O(|\Gamma(p)|)$ nodes), and then combines the different rings of $G'$.

### 5.1 A Simple "Ring" Entwined Structure

This section presents a simple construction of an entwined structure $\mathcal{G}$ for a ring topology network. The constructed $\mathcal{G}$ has $|g_{max}|$ constant (independent of $n$), but a diameter of $O(n)$.

**Definition 16.** *A graph $G = (V, E)$ is said to have an* **extended ring topology** *of order $k$, if $E = \bigcup_{p_i \in V} \bigcup_{1 \le j \le k} \{(p_i, p_{i+j})\}$ (addition is done modulo $n$).*

Informally, an extended ring topology of order $k$ means that each node is connected to the $k$ neighbors "ahead" of it in the ring; since $G$ is bi-directional, each node is also connected to the $k$ neighbors "behind" it (see Figure 2 for an example). Notice that a "regular" ring topology is actually an extended ring topology of order 1.

Consider a graph $G = (V, E)$ with extended ring topology of order $k = 3 \cdot (f_\ell + 1)$. Define $\mathcal{G}$ as follows: $g_i := \{p_i, p_{i+1}, p_{i+2}, \ldots, p_{i+k}\}$ (see Figure 3).

*Claim.* Any $g \in \mathcal{G}$ is a $f_\ell$-decision group.

*Claim.* For any $i$, $g_i$ and $g_{i+1}$ are $f_\ell$-connected.

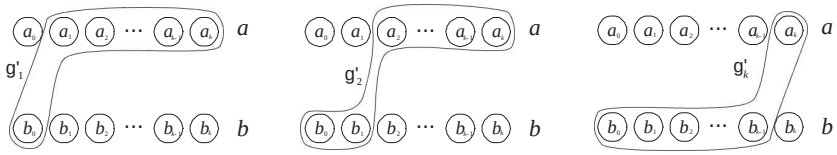


**Fig. 4.** The original decision groups $a, b$ and the constructed decision groups $g'_1, g'_2$ and $g'_k$.

**Lemma 5.** *$\mathcal{G}$ is an $f_\ell$-entwined structure with diameter at most $n$, and $|g| = 3 \cdot f_\ell + 4$ for any $g \in \mathcal{G}$.*

*Proof.* By definition of $g_i \in \mathcal{G}$, $|g_i| = k + 1 = 3 \cdot f_\ell + 4$. Let $g_i, g_j \in \mathcal{G}$ be some decision groups. $0 \leq i, j \leq n - 1$ and either $i \geq j$ or $j \geq i$. Assume w.l.o.g. that $j \geq i$. Consider the list $C := g_i, g_{i+1}, \ldots, g_j$. By the previous claim, $g_i$ and $g_{i+1}$ are $f_\ell$-connected, and therefore $C$ is a value chain. Thus, there exists a $f_\ell$-value chain between any two decision groups in $\mathcal{G}$ of length at most $n$. Thus, $\mathcal{G}$ is an $f_\ell$-entwined structure with diameter at most $n$. □

### 5.2 Connecting Rings Together

Consider two extended ring topology graphs $G_1, G_2$ and their respective $f_\ell$-entwined structures $\mathcal{G}_1, \mathcal{G}_2$ (as built in the previous subsection). Let $a \in \mathcal{G}_1$ and $b \in \mathcal{G}_2$ be 2 decision groups that are fully connected and are of the same size, e.g. $|a| = |b| = k + 1$.

Mark by $a_i$ the nodes in $a$ and by $b_i$ the nodes in $b$ (for $0 \leq i \leq k$). Define $g'_j := \{a_j, a_{j+1}, \ldots, a_k, b_0, \ldots, b_{j-1}\}$ for $1 \leq j \leq k$; see Figure 4 for an example. Add edges such that each $g'_j$ is fully connected. Notice that $|g'_j| = k+1$ and that for $1 \leq j < k$, it holds that $g'_j$ and $g'_{j+1}$ are $f_\ell$-connected. In addition $a$ and $g'_0$ are $f_\ell$-connected, as well as $b$ and $g'_k$.

Take $\mathcal{G}' = \mathcal{G}_1 \bigcup \mathcal{G}_2 \bigcup \{g'_j\}$. $\mathcal{G}'$ is a $f_\ell$-entwined structure over the graph that is the union of $G_1, G_2$ with edges that are induced by the different $g'_j$s. Notice that the diameter of $\mathcal{G}'$ is the diameter of $\mathcal{G}_1$ plus the diameter of $\mathcal{G}_2$ plus $k + 1$ (the length of the distance between $a$ and $b$). Furthermore, $|\mathcal{G}'| = |\mathcal{G}_1| + |\mathcal{G}_2| + k$. In addition every node $p \in a \bigcup b$ participates in no more than $k$ additional decision groups.

### 5.3 General Entwined Structure Construction

Let $G'$ be any graph on $n'$ nodes, and let $d_v$ be the degree of node $v$; denote $d_{max} := max_{v \in G'} d_v$. Consider a graph $G_v$ per node $v$ with $d_v \cdot k$ nodes, such that $G_v$ has an extended ring topology of order $k$. Consider the "ring" entwined structure constructed in the previous sections $\mathcal{G}_v$ with decision groups denoted as $g_1(v), g_2(v), \cdots$. Notice that $g_i(v) \cap g_{i+k}(v) = \phi$ (see Figure 5). For each node $v \in G'$, consider the list of its neighbors $\Gamma(v) = v_1, v_2, \ldots, v_{d_v}$, and define a function $N(u, v)$ that returns the index of $v$ as a neighbor of $u$; *i.e.,* $N(v, v_i) = i$.

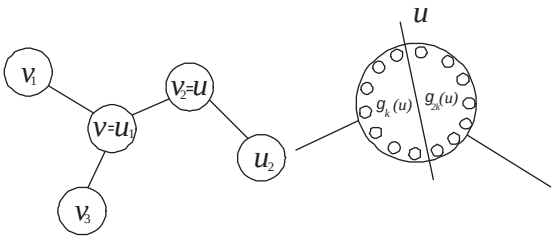


**Fig. 5.** A graph with 5 nodes, and node $u$'s "ring" $\mathcal{G}_u$ and 2 decision groups $g_k(u), g_{2k}(u)$.

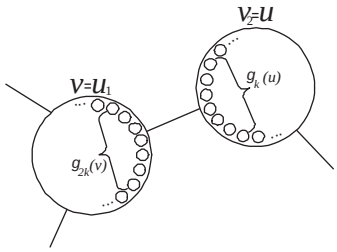


**Fig. 6.** $v, u$ connecting, combining $g_{2k}(v), g_k(u)$.

Using the operation defined in the previous subsection: for any edge $(u, v)$ in $G'$, "connect" the rings in the following way: $g_{N(u,v)*k}(u)$ with $g_{N(v,u)*k}(v)$. That is, let $v$ be the $i$th neighbor of $u$ ($N(u, v) = i$ or $u_i = v$), and $u$ be the $j$th neighbor of $v$ ($N(v, u) = j$, $v_j = u$) (see Figure 6); the following two decision groups are to be combined: $g_{i*k}(u)$ with $g_{j*k}(v)$. Denote the new decision groups created due to each such "connection" as $\mathcal{G}_{u,v}$.

Consider $G$ to be the union of all $G_v$ (both nodes and edges), and let $\mathcal{G}$ be the union of the entwined structures. That is, $\mathcal{G} = (\bigcup_{v \in G'} \mathcal{G}_v) \cup (\bigcup_{u,v \in G'} \mathcal{G}_{u,v})$.

*Claim.* Let $g \in \mathcal{G}_v$ and $g' \in \mathcal{G}_u$ such that $(v, u)$ is an edge in $G'$, then there is a $f_\ell$-value-chain between $g$ and $g'$ of length $\leq (d_{max} + 1) \cdot k$.

*Claim.* $\mathcal{G}$ is a $f_\ell$-entwined structure with diameter $\mathcal{D} \leq D_{G'} \cdot (d_{max} + 1) \cdot k + 2k$, where $D_{G'}$ is the diameter of $G'$.

**Theorem 3.** *$\mathcal{G}$ is a 1-lightweight $f_\ell$-entwined structure.*

### 5.4 Analysis

The above construction shows that for any graph $G'$ there is a graph $G$ and an $f_\ell$-entwined structure $\mathcal{G}$ (covering $G$) s.t. the diameter of $\mathcal{G}$ is linear in the diameter of $G'$, the maximal degree of $G'$, and in $k$. Thus, by taking $G'$ to be any graph with constant degree and logarithmic diameter, we have that the diameter of $\mathcal{G}$ is $O(D_{G'} \cdot k)$. In other words, $\mathcal{D} = O(k \cdot logn)$; by taking $k$ to be constant as well (this means that the graph $G$ has a constant degree), we have that $\mathcal{D} = O(logn)$. Therefore, the above construction produces graphs and their respective entwined structures, such that the diameter is logarithmic and the degree is constant, fulfilling the promise of Section 4.3.

## 6 Conclusion and Future Work

A sufficient condition for solving *Byzantine* consensus in the presence of a localized *Byzantine* adversary was given. Furthermore, for a family of graphs it was

shown how to solve the *Byzantine* consensus problem using memory space that is constant (independent of the size of the network).

We consider few points for future research: first, find tighter bounds for when *Byzantine* consensus can be solved on a graph $G$. Second, allow for dynamic changes in the system, such as nodes joining or leaving or even constructing the entwined structure on-the-fly. Third, adapt the entwined structure-construction such that if some portion of the network does not uphold the required *Byzantine*-to-correct threshold, then the rest of the network may still reach consensus. Lastly, it would be interesting to find other constructions of entwined structures and see what additional types of graphs can solve *Byzantine* consensus.

## Acknowledgements

We would like to thank the anonymous reviewers for their helpful comments.

## References

1. Y. Afek and G. Stupp. Optimal time-space tradeoff for shared memory leader election. *J. Algorithms*, 25(1):95–117, 1997.
2. H. Attiya and J. Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, 2004.
3. J. Beauquier, M. Gradinariu, and C. Johnen. Memory space requirements for self-stabilizing leader election protocols. In *PODC '99: Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing*, pages 199–207, New York, NY, USA, 1999. ACM.
4. J. Beauquier, M. Gradinariu, and C. Johnen. Randomized self-stabilizing and space optimal leader election under arbitrary scheduler on rings. *Distributed Computing*, 20(1):75–93, July 2007.
5. Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.*, 20(4):398–461, 2002.
6. D. Dolev. The byzantine generals strike again. *Journal of Algorithms*, 3:14–30, 1982.
7. D. Dolev and R. Reischuk. Bounds on information exchange for byzantine agreement. *J. ACM*, 32(1):191–204, 1985.
8. S. Dolev. *Self-Stabilization*. The MIT Press, 2000.
9. Shlomi Dolev, Mohamed G. Gouda, and Marco Schneider. Memory requirements for silent stabilization. In *PODC '96: Proceedings of the fifteenth annual ACM symposium on Principles of distributed computing*, pages 27–34, New York, NY, USA, 1996. ACM.
10. Chiu-Yuen Koo. Broadcast in radio networks tolerating byzantine adversarial behavior. In *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of distributed computing*, pages 275–282, New York, NY, USA, 2004. ACM.
11. L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, 4(3):382–301, 1982.
12. N. Lynch. *Distributed Algorithms*. Morgan Kaufmann, 1996.
13. Andrzej Pelc and David Peleg. Broadcasting with locally bounded byzantine faults. *Inf. Process. Lett.*, 93(3):109–115, 2005.

14. S. Toueg, K. J. Perry, and T. K. Srikanth. Fast distributed agreement. *SIAM Journal on Computing*, 16(3):445–457, Jun 1987.
15. M. Yamashita and T. Kameda. Computing on anonymous networks. i. characterizing the solvable cases. *Parallel and Distributed Systems, IEEE Transactions*, 7(1):69–89, Jan 1996.