The Topology of Asynchronous Byzantine Colorless Tasks

Hammurabi Mendes[a], Christine Tasson[b], and Maurice Herlihy*[a]

[a] Computer Science Dept., Brown University, Providence, RI, USA, {hmendes,mph}@cs.brown.edu

[b] Univ. Paris Diderot, Sorbonne Paris Cité, PPS, UMR 7126, CNRS, F-75205, Paris, France, Christine.Tasson@pps.univ-paris-diderot.fr

Contact:	Hammurabi Mendes <hmendes@cs.brown.edu>
Address:	Box 1910, Computer Science Department, Brown University
	115 Waterman Street
	Providence RI 02906, USA
Telephone:	+1 (401) 580 3458

Abstract

In this paper, we extend the topological model that characterizes task solvability in crash-failure systems to colorless tasks in Byzantine asynchronous systems. We give the first theorem with necessary and sufficient conditions to solve *arbitrary* colorless tasks in such model, capturing the relation between the total number of processes, the number of faulty processes, and the topological structure of the task's simplicial complexes. In the interim, we provide novel asynchronous protocols for k-set agreement and barycentric agreement with optimal Byzantine fault tolerance.

Regular submission.

Please consider for student paper award: Hammurabi Mendes is a full-time student.

*Supported by NSF 000830491.

1 Introduction

A *task* is a distributed coordination problem in which each process starts with a private input from a finite set, communicates with other processes, and eventually decides on a private output, also from a finite set.

One of the central questions in distributed computing is characterizing which tasks can be solved in which models of computation. Tools adapted from combinatorial topology have been successful in characterizing task solvability in synchronous and asynchronous *crash-failure* models [16], where faulty processes simply halt and fail silently. In this paper, we extend the approach to the asynchronous *Byzantine* model, where communication has unbounded delay and faulty processes can display arbitrary behavior.

We focus on an important class of tasks called *colorless* tasks [4], which includes well-studied problems such as consensus [9], k-set agreement [8], and approximate agreement [1]. As explained more formally below, a colorless task is one that can be defined entirely in terms of *sets* of input and output values assigned, without the need to specify which value is assigned to which process, or how many times an assigned value appears[1].

1.1 Our Contributions

Our first contribution is to **extend** the application of the topological model from [16], formerly used to characterize solvability in crash-failure systems, to colorless tasks in asynchronous Byzantine systems. This approach leads to novel conclusions, suggesting that the model is well-suited and robust among a multitude of communication and failure abstractions. Besides, it opens up new questions, such as how to further extend the application to colored tasks, as well as other timing models. Background information on the topological model in distributed computing is presented in Sec. 2.

Our second contribution is to give **new protocols** for k-set agreement and barycentric agreement in the Byzantine-failure model. It is well-known that asynchronous Byzantine consensus is impossible [9], as well as wait-free asynchronous set-agreement [4, 16, 21]. Our protocols limn the boundary between the possible and impossible in the presence of Byzantine failures. Using powerful algorithmic tools, presented in Sec. 3, our k-set agreement and barycentric agreement protocols, which are discussed in Sec. 4, will provide the appropriate support for our main theorem.

Finally, our principal contribution is to give the **first theorem** with necessary and sufficient conditions to solve *non-trivial colorless tasks* in the asynchronous Byzantine model. The task itself is defined in terms of a pair of combinatorial structures called *simplicial complexes* [20, 18]. Whether a task is solvable is equivalent to the existence of a certain structure-preserving map between the task's simplicial complexes. This equivalence captures the relation between $n+1$, the number of processes, t, the number of failures, and the topological structure of the task's simplicial complexes[2]. While an analogous characterization has long been known for crash failures [16], our solvability theorem, presented in Sec. 5, is the first such characterization for Byzantine failures.

[1] The *renaming* task [2] is an example of a task that is *not* colorless, as each process chooses a distinct name.
[2] Our results extend to the core/survivor-set model [17] (Appendix C).

2 Topological Model

We now overview some important notions from combinatorial topology, and describe how they are applied to model concurrent computation. For details, please refer to Munkres [20] or Kozlov [18].

2.1 Combinatorial Tools

A *simplicial complex* is a finite set V along with a collection of subsets $\mathcal{K}$ of V closed under containment. An element of V is a *vertex* of $\mathcal{K}$. Each set in $\mathcal{K}$ is called a *simplex*, usually denoted by lower-case Greek letters: σ, τ. A subset of a simplex is called a *face*. The *dimension* $\dim(\sigma)$ of a simplex σ is $|\sigma| - 1$. We use "k-simplex" as shorthand for "k-dimensional simplex", and similarly for "k-face". The dimension $\dim(\mathcal{K})$ of a complex is the maximal dimension of its simplices. The set of simplices of $\mathcal{K}$ of dimension at most ℓ is a subcomplex of $\mathcal{K}$, which is called *ℓ-skeleton* of $\mathcal{K}$, denoted by $\operatorname{skel}^\ell(\mathcal{K})$.

Let $\mathcal{K}$ and $\mathcal{L}$ be complexes. A *vertex map* f carries vertices of $\mathcal{K}$ to vertices of $\mathcal{L}$. If f additionally carries simplices of $\mathcal{K}$ to simplices of $\mathcal{L}$, it is called a *simplicial* map. A *carrier map* Φ from $\mathcal{K}$ to $\mathcal{L}$ takes each simplex $\sigma \in \mathcal{K}$ to a subcomplex $\Phi(\sigma) \subseteq \mathcal{L}$, such that for all $\sigma, \tau \in \mathcal{K}$, we have that $\Phi(\sigma \cap \tau) \subseteq \Phi(\sigma) \cap \Phi(\tau)$. A simplicial map $\phi : \mathcal{K} \to \mathcal{L}$ is *carried by the carrier map* $\Phi : \mathcal{K} \to 2^{\mathcal{L}}$ if, for every simplex $\sigma \in \mathcal{K}$, we have $\phi(\sigma) \subseteq \Phi(\sigma)$.

Although we defined simplices and complexes in a purely combinatorial way, they can also be interpreted geometrically. An n-simplex can be identified with the convex hull of $(n+1)$ affinely-independent points in the Euclidean space of appropriate dimension. This geometric interpretation can be extended to complexes. The point-set underlying such *geometric complex* $\mathcal{K}$ is called its *polyhedron*, and is denoted $|\mathcal{K}|$.

We can define simplicial and carrier maps in geometrical complexes. Given a simplicial map $\phi : \mathcal{K} \to \mathcal{L}$ (resp. carrier map $\Phi : \mathcal{K} \to 2^{\mathcal{L}}$), the polyhedrons of every simplex in $\mathcal{K}$ and $\mathcal{L}$ induce a continuous simplicial map $\phi_c : |\mathcal{K}| \to |\mathcal{L}|$ (resp. continuous carrier map $\Phi_c : |\mathcal{K}| \to |2^{\mathcal{L}}|$). Furthermore, we say that ϕ (resp. ϕ_c) is carried by Φ if, for every simplex $\sigma \in \mathcal{K}$, we have that $|\phi(\sigma)| \subseteq |\Phi(\sigma)|$ (resp. $\phi_c(|\sigma|) \subseteq \Phi_c(|\sigma|)$).

2.1.1 Simplicial Approximations

For any simplex σ, the *boundary* of σ, denoted $\partial\,\sigma$, is the simplicial complex of proper faces of σ. The *interior* of σ is formally defined as $\operatorname{Int}\sigma = |\sigma| \setminus |\,\partial\,\sigma|$. The *open star* of $\sigma \in \mathcal{A}$, denoted $\operatorname{Ost}\sigma$, is the union of the interiors of all simplices in $\mathcal{A}$ containing σ. See Fig. 1.

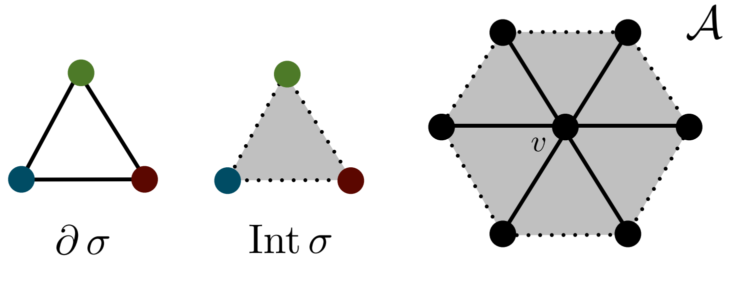

Figure 1: The interior and boundary of a 2-simplex σ, and the open star of a 0-simplex $\{v\} \subseteq \mathcal{A}$.

Intuitively speaking, given a continuous map, a simplicial approximation is a "sufficiently close" combinatorial counterpart. We define it formally below.

Definition 2.1. A simplicial map $\mu : \mathcal{K} \to \mathcal{L}$ is a *simplicial approximation* of $\Phi_c : |\mathcal{K}| \to |\mathcal{L}|$ if

$$\Phi_c(|\operatorname{Int}\sigma|) \subseteq \bigcap_{v\in\sigma} |\operatorname{Ost}\mu(v)| = |\operatorname{Ost}\mu(\sigma)|, \text{ for all } \sigma \in \mathcal{K}.$$

We introduce new concepts in order to define the conditions for the existence of a simplicial approximation. A subdivision of a complex $\mathcal{A}$ is a complex $\mathcal{B}$ such that: (i) for *any* $\tau \in \mathcal{B}$, $|\tau|$ is contained in the polyhedron of some $\sigma \in \mathcal{A}$. (ii) for *any* $\sigma \in \mathcal{A}$, $|\sigma|$ is the union of disjoint polyhedrons of simplices belonging to $\mathcal{B}$. We understand the subdivision as an operator Div over complexes. A *mesh-shrinking* subdivision $\operatorname{Div}\mathcal{A}$ of $\mathcal{A}$ is one such the longest polyhedron of an edge in $\operatorname{Div}\mathcal{A}$ (i.e., a 1-simplex in $\operatorname{skel}^1(\operatorname{Div}\mathcal{A})$) is strictly smaller than the corresponding one in $\mathcal{A}$ [3].

Theorem 2.2. (Simplicial Approximation Theorem [20]) Given a continuous map $\Phi_c : |\mathcal{K}| \to |\mathcal{L}|$, and any mesh-shrinking subdivision operator Div, there is an $N > 0$ such that Φ_c has a simplicial approximation $\mu : \operatorname{Div}^N \mathcal{K} \to \mathcal{L}$.

2.1.2 Connectivity

We say that a simplicial complex $\mathcal{K}$ is x-connected if every continuous map of a x-sphere in $|\mathcal{K}|$ can be extended into the continuous map of a $(x+1)$-disk in $|\mathcal{K}|$. In analogy, think of a pencil as a 1-disk, and its extremes as a 0-sphere; a coin as a 2-disk, and its border as a 1-sphere; a billiard ball as a 3-disk, and its outer layer as a 2-sphere. In addition, (-1)-connected is defined as non-empty.

Fact 2.3. [20] For any k-simplex σ, the boundary of σ is homeomorphic to a $(k-1)$-sphere, and σ is homeomorphic to a k-disk.

2.2 Model for Concurrent Computation

We now present a simplified version of the general combinatorial model of [10, 11, 13, 14, 16] that is specially suitable for colorless tasks. We initially describe our operational context.

We have $n+1$ sequential processes[4], $P_0, \ldots, P_n$, that communicate via message-passing through a fully-connected, asynchronous network of reliable FIFO channels. Every message sent is eventually delivered after a finite, but unbounded delay, with the message's sender reliably identified. Processes are asynchronous as well, having no bound on their relative speed. Up to t processes might be *faulty*, displaying arbitrary, "Byzantine" behavior. We abstract the asynchronous communication in discrete rounds of related messages, in a full-information protocol.

2.2.1 Colorless Tasks

An *initial configuration* is an assignment of input values to processes, and a *final configuration* is the analogous for output values. A *task specification* consists of a set of legal initial configurations, and, for each of them, its corresponding set of legal final configurations.

A *colorless task* [14, 5] is a triple $(\mathcal{I}, \mathcal{O}, \Delta)$, where $\mathcal{I}$ is the *input complex*, $\mathcal{O}$ is the *output complex*, and $\Delta : \mathcal{I} \to 2^{\mathcal{O}}$ is a carrier map. Each vertex in $\mathcal{I}$ is an input value, and each simplex is an initial configuration with possible initial configurations closed under inclusion. The output

[3] Intuitively, we are effectively shrinking the borders of the simplexes.

[4] Choosing $n+1$ processes rather than n simplifies the topological notation but slightly complicates the computing notation. Choosing n processes makes the opposite trade-off. We choose $n+1$ for compatibility with prior work.

complex is analogous in regard to final configurations. Given an initial configuration, the carrier map Δ specifies which final configurations are legal.

With colorless tasks, input and output simplexes represent *sets* of input and output values, held by the processes in the appropriate initial/final configuration. Such sets are closed under inclusion, that is, if σ represents a valid initial or final configuration, so is any face $\tau \subseteq \sigma$. The admissible *sets* of output values depend solely on the starting *set* of input values picked by processes.

Indeed, in k-set agreement [8], which is colorless, a process can adopt the input of any other process, or two processes can exchange their inputs, and the initial configuration remains valid, and similarly for outputs. Conversely, in *renaming* [2], which is colored, the specific input/output assignments cannot be manipulated in such ways[5].

To illustrate the relation between the carrier map and the task specification, in binary consensus the input complex $\mathcal{I}$ has two vertices, labeled 0 and 1, joined by an edge, and the output complex $\mathcal{O}$ has two isolated vertices, labeled 0 and 1. The carrier map Δ sends each vertex of $\mathcal{I}$ to the output vertex with the same label, and the edge of $\mathcal{I}$ to both vertices of $\mathcal{O}$.

2.2.2 Byzantine Colorless Tasks

In our model for Byzantine colorless tasks, we require that outputs of non-faulty processes depend solely on inputs of non-faulty processes. We formalize the concept below.

For any non-faulty process P_i, its input is denoted I_i and its output is denoted O_i. The set of *non-faulty inputs* is defined as $\sigma_I = \{I_i : P_i \text{ is non-faulty}\}$, and the set of *non-faulty outputs* is defined as $\sigma_O = \{O_i : P_i \text{ is non-faulty}\}$. A Byzantine colorless task requires that

$$\sigma_O \in \Delta(\sigma_I). \tag{1}$$

In other words, the admissible non-faulty output *sets* depend solely on the starting non-faulty input *set* held by non-faulty processes. So, we only care about inputs and outputs of non-Byzantine processes, and Byzantine inputs do not "influence" outputs of non-Byzantine processes.

We model such requirement by making our task specification $(\mathcal{I}, \mathcal{O}, \Delta)$ constrain the behavior of non-faulty processes *only*. Now, initial and final configurations refer to non-faulty processes, and inputs and output simplexes represent non-faulty input and output sets, as defined before. Conceptually, non-faulty processes start on vertices of a simplex σ in $\mathcal{I}$, and halt on vertices of a simplex $\tau \in \Delta(\sigma)$. Multiple non-faulty processes may start on the same input vertex and halt on the same output vertex.

Therefore, the *same* task definition triple $(\mathcal{I}, \mathcal{O}, \Delta)$ is used both for crash and Byzantine failure models, with the nature of the failures affecting only *protocols* (algorithms), but not *specifications* (models) – failures only alter the specification *scope*, but not its *definition*. Perhaps contrary to intuition, we *extend* the model to Byzantine tasks by *constraining* the specification semantics to non-faulty processes *only*.

2.2.3 Protocols and Complexes

Informally, we abstract protocols as continuous exchanges of internal states, in a full-information fashion [15]. Given a model for communication and failures, we can define a *protocol complex* $\mathcal{P}(\mathcal{I})$ for any task $(\mathcal{I}, \mathcal{O}, \Delta)$. A vertex in $v \in \mathcal{P}(\mathcal{I})$ is a tuple with a non-faulty process identifier and

[5] If one process adopts the output of another process, the output configuration becomes forbidden.

its final state. A simplex $\sigma = \{(Q_1, s_1), \ldots, (Q_x, s_x)\}$ in $\mathcal{P}(\mathcal{I})$ indicates that, in some execution, non-faulty processes $Q_1, \ldots, Q_x$ finish with states $s_1, \ldots, s_x$, respectively. The formal definition of *protocol* is identical to [14], here presented for completeness:

Definition 2.4. A protocol for $(\mathcal{I}, \mathcal{O}, \Delta)$ is a carrier map $\mathcal{P}$ taking $\sigma \in \mathcal{I}$ to a protocol complex denoted $\mathcal{P}(\sigma) \subseteq \mathcal{P}(\mathcal{I})$. For any $\sigma, \tau \in \mathcal{I}$, we have that $\mathcal{P}(\sigma \cap \tau) = \mathcal{P}(\sigma) \cap \mathcal{P}(\tau)$.

Definition 2.5. A protocol $\mathcal{P}$ solves $(\mathcal{I}, \mathcal{O}, \Delta)$ if there exists a simplicial map $\delta : \mathcal{P}(\mathcal{I}) \to \mathcal{O}$ carried by Δ.

3 Basic Algorithmic Tools

In this work, we use two well-known constructions from prior work: *reliable broadcast* [6, 22, 3, 7] and *stable vectors* [2]. In the following sections, we overview these primitives, and define our concept of *quorum* that underlies our algorithms for Byzantine colorless tasks.

3.1 Reliable Broadcast

The *reliable broadcast* technique forces Byzantine processes to communicate consistently with other processes. Communication is organized in asynchronous rounds, where a round may involve several message exchanges. Messages have the form (P, r, c), where P is the sending process, r is the current round, and c is the actual content. Messages not conforming to this structure can safely be discarded. The technique, which works as long as $n > 3t$, guarantees the following [6, 22, 3, 7]:

Non-Faulty Integrity: If a non-faulty P never reliably broadcasts (P, r, c), no non-faulty process ever reliable receives (P, r, c).

Non-Faulty Liveness: If a non-faulty P does reliably broadcast (P, r, c), all non-faulty processes will reliably receive (P, r, c) eventually.

Global Uniqueness: If two non-faulty processes Q and R reliably receive, respectively, (P, r, c) and (P, r, c'), then the messages are equal ($c = c'$), even if the sender P is Byzantine.

Global Liveness: For two non-faulty processes Q and R, if Q reliably receives (P, r, c), then R will reliably receive (P, r, c) eventually, even if the sender P is Byzantine.

For completeness, we provide the algorithms in Appendix A. Here, $P.\texttt{RBSend}(M)$ denotes the reliable broadcast of message M by process P, and $P.\texttt{RBRecv}(M)$ the reliable receipt of M by P.

3.2 Quorums

Let $\mathcal{M}^r$ be the set of all messages reliably broadcast by all processes during a discrete round r, and let $M_i^r \subseteq \mathcal{M}^r$ be the subset reliably received by a non-faulty process P_i. By the global uniqueness of the reliable broadcast, if (P, r, c) and (P', r', c') are distinct messages in M_i^r then the senders are distinct: $P \neq P'$. Furthermore, by definition of M_i^r, we have that $r = r'$. Let $Good(\mathcal{M}^r)$ denote the set of distinct message contents in $\mathcal{M}^r$ that were reliably broadcast by the non-faulty processes.

Definition 3.1. We say that a content c has a *quorum* in M_i^r, written $c \in Quorum(M_i^r)$, if c was reliably received in M_i^r from $t + 1$ or more different processes.

For any $M \subseteq \mathcal{M}^r$, we know that $\mathit{Quorum}(M) \subseteq \mathit{Good}(\mathcal{M}^r)$: if P_i obtained a quorum for c, P_i becomes aware that c was sent by a non-faulty process. Conversely, any content received from less than $t+1$ processes cannot be "trusted" – recall that non-faulty process outputs must depend only on non-faulty process inputs.

Algorithm 1 shows the procedure by which non-faulty processes get a set of messages containing a quorum in a communication round r. It works as long as $n+1 > t \cdot \max\{3, |\mathit{Good}(\mathcal{M}^r)|+1\}$. We denote by M_i^r the set M received by process P_i at round r.

Algorithm 1 P.`RecvQuorum`(r)

```
Require: $n+1 > t \cdot \max\{3, |\mathit{Good}(\mathcal{M}^r)|+1\}$
1: $M \leftarrow \emptyset$
2: while $|M| < (n+1) - t$ or $\mathit{Quorum}(M) = \emptyset$ do
3:     upon RBRecv($(Q, r, c)$) do
4:         $M \leftarrow M \cup \{(Q, r, c)\}$
5: return $M$
```

Informally, the procedure eventually finishes since all $(n+1)-t$ non-faulty processes eventually show up, and, since $(n+1)-t > t \cdot |\mathit{Good}(\mathcal{M}^r)|$, some value will appear $t+1$ or more times. The following properties, formally proved in in Appendix B, present the guarantees of the algorithm.

Property 3.2. If $n+1 > t \cdot \max\{3, |\mathit{Good}(\mathcal{M}^r)|+1\}$, then `RecvQuorum`$(r)$ eventually returns, and, for any non-faulty process P_i,

$$|M_i^r| \geq (n+1) - t \text{ and } \mathit{Quorum}(M_i^r) \neq \emptyset.$$

Property 3.3. After `RecvQuorum`(r), any two non-faulty processes P_i and P_j have $|M_i^r \setminus M_j^r| \leq t$.

3.3 Stable Vectors

Algorithm 1 ensures that any two non-faulty processes P_i and P_j collect at least $n+1-2t$ messages in common. Indeed, $|M_i^r| \geq (n+1)-t$ by Property 3.2 and $|M_i^r \setminus M_j^r| \leq t$ by Property 3.3, therefore $|M_i^r \cap M_j^r| \geq (n+1) - 2t$.

In Algorithm 2, we adapt the *stable vectors* technique presented in [2] to ensure that the two non-faulty processes P_i and P_j have $(n+1)-t$ messages in common in M_i^r and M_j^r, with a common value in $\mathit{Quorum}(M_i^r \cap M_j^r)$ and with sets of messages totally ordered by containment. Since we use the `RecvQuorum`() procedure, it requires that $n+1 > t \cdot \max\{3, |\mathit{Good}(\mathcal{M}^r)|+1\}$,

The technique works as follows. (1) P uses `RecvQuorum`() to obtain a set of messages M with $\mathit{Quorum}(M) \neq \emptyset$. (2) P reliably transmits its *report*, containing those messages. (3) Any further message reliably received in M causes P to reliably broadcast an updated report of M. (4) P keeps reliably receiving reports, stored in a vector R, until it identifies $(n+1)-t$ *buddies* in B, where a buddy is a process P_j whose last report R_j is M. (5) P decides, but keeps updating M and sending updated reports in the background (Algorithm 3).

The following lemmas (proved in Appendix B) show that the procedure terminates, and give the stable vector properties. Intuitively, any two non-faulty processes identify a common non-faulty buddy, which reliably broadcasts monotonically increasing reports, guaranteeing the properties.

Lemma 3.4. For any non-faulty process P_i, the sequence of transmitted reports is monotonically increasing. All other non-faulty processes receive those reports in such order.

Algorithm 2 P.`RecvStable`(r)

```
Require: $n + 1 > t \cdot \max\{3, |Good(\mathcal{M}^r)| + 1\}$
1: $M, B \leftarrow \emptyset$
2: $R[x] \leftarrow \emptyset$ for all $0 \leq x \leq n$
3: $M \leftarrow$ RecvQuorum$(r)$
4: RBSend$((P, r\{$report$\}, M))$
5: while $|B| < (n + 1) - t$ do
6:     upon RBRecv$((P_j, r, c))$ do
7:         $M \leftarrow M \cup \{(P_j, r, c)\}$
8:         RBSend$((P, r\{$report$\}, M))$
9:     upon RBRecv$((P_j, r\{$report$\}, R_j))$ do
10:        $R[j] \leftarrow R_j$
11:    $B \leftarrow \{P_x : R[x] = M, 0 \leq x \leq n\}$
12: return $M$                                  ▷ while activating RSEcho$(M, r)$
```

Algorithm 3 P.`RSEcho`(M, r)

```
1: upon RBRecv$((Q, r, c))$ do
2:     $M \leftarrow M \cup \{(Q, r, c)\}$
3:     RBSend$((P, r\{$report$\}, M))$
```

Lemma 3.5. `RecvStable`(r) eventually returns.

Lemma 3.6. After `RecvStable`(r), any two non-faulty processes P_i and P_j satisfy the following: (i) $|M_i^r \cap M_j^r| \geq (n + 1) - t$; (ii) $Quorum(M_i^r \cap M_j^r) \neq \emptyset$; and (iii) $M_i^r \subseteq M_j^r$ or $M_j^r \subseteq M_i^r$.

4 k-Set Agreement and Barycentric Agreement

Our solvability theorem for colorless tasks builds on two novel asynchronous Byzantine protocols: k-set agreement [8], and *barycentric agreement* [16], which depend, in turn, on our previous communication primitives. We describe the protocols in the following sections.

4.1 k-Set Agreement

In the k-set agreement task [8], processes decide up to k values originally input. In our model, non-faulty processes start on vertices of a simplex σ and halt on vertices of a simplex of $\text{skel}^{k-1}(\sigma)$. Informally, they decide on up to k values originally input by non-faulty processes.

Algorithm 4 P.`KSetAgree`(v)

```
Require: $k > t$ and $n + 1 > t(d + 2)$, with $\dim(\mathcal{I}) = d > 0$.
1: RBSend$((P, 1, v))$
2: $M \leftarrow$ RecvQuorum$(1)$
3: return least-ranked element in $Quorum(M)$
```

Algorithm 4 shows a k-set agreement protocol assuming that $k > t$ and $n + 1 > t(d + 2)$, with

$\dim(\mathcal{I}) = d > 0$. Since at most $d+1$ distinct values are reliably broadcast by non-faulty processes, $|Good(\mathcal{M}^1)| \leq d+1$, and Algorithm 1 becomes applicable as $n+1 > t \cdot \max\{3, |Good(\mathcal{M}^1)|+1\}$.

Lemma 4.1. Algorithm 4 solves the k-set agreement problem for Byzantine colorless tasks.

Proof. Non-faulty processes decide values in $Quorum(M) \subseteq Quorum(\mathcal{M}^1) \subseteq Good(\mathcal{M}^1)$. For any non-faulty process P_i, $|M| \geq (n+1)-t$ and $Quorum(M) \neq \emptyset$ (Property 3.2), so

$$|\mathcal{M}^1 \setminus M| \leq t \Rightarrow |Quorum(\mathcal{M}^1) \setminus Quorum(M)| \leq t.$$

Therefore, P_i misses up to t decidable values in $Quorum(\mathcal{M}^1)$, so P_i's decision is among the $(t+1)$ least-ranked elements. Since P_i was taken arbitrarily, all non-faulty processes decide similarly, and the claim follows because $k \geq t+1$. ☐

4.2 Barycentric Agreement

The barycentric subdivision of simplex σ is constructed, informally speaking, by subdividing σ along the barycenters of its faces, as shown in Figure 2. Formally, the barycentric subdivision of σ is a simplicial complex $\operatorname{Bary}\sigma$ whose vertices are faces of σ, and whose simplices are chains of distinct faces totally ordered by containment. Every m-simplex $\tau \in \operatorname{Bary}\sigma$ might be written as $\{\sigma_0, \ldots, \sigma_m\}$, where $\sigma_0 \subset \cdots \subset \sigma_m \subseteq \sigma$.

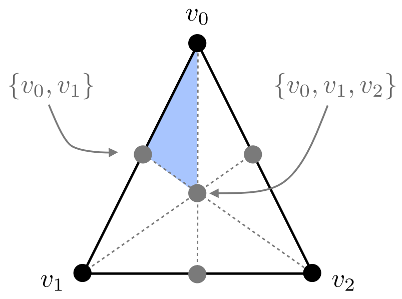

Figure 2: Barycentric subdivision of $\sigma = \{v_0, v_1, v_2\}$, and one of its simplices $\{\{v_0\}, \{v_0, v_1\}, \{v_0, v_1, v_2\}\}$.

In the barycentric agreement task, non-faulty processes start on vertices of a simplex σ and halt on vertices of a simplex in $\operatorname{Bary}\sigma$. Formally, for each $\sigma \in \mathcal{I}$, we have $\Delta(\sigma) = \operatorname{Bary}\sigma$.

Algorithm 5 shows a barycentric agreement protocol assuming that $n+1 > t(d+2)$, with $\dim(\mathcal{I}) = d > 0$. Since at most $d+1$ distinct values are reliably broadcast by non-faulty processes, $|Good(\mathcal{M}^1)| \leq d+1$, and Algorithm 2 becomes applicable as $n+1 > t \cdot \max\{3, |Good(\mathcal{M}^1)|+1\}$.

Algorithm 5 $P.$`BaryAgree`(v)

```
Require: $n+1 > t(d+2)$, with $\dim(\mathcal{I}) = d > 0$.
1: RBSend($(P, 1, \{v\})$)
2: $M \leftarrow$ RecvStable($1$)
3: return $Quorum(M)$
```

Lemma 4.2. Algorithm 5 solves the barycentric agreement problem for Byzantine colorless tasks.

Proof. By Lemma 3.6, for any two non-faulty processes P_i and P_j, we have that $M_i \subseteq M_j$ or $M_j \subseteq M_i$, and also that $Quorum(M_i \cap M_j) \neq \emptyset$. Therefore, $Quorum(M_i) \subseteq Quorum(M_j)$ or $Quorum(M_j) \subseteq Quorum(M_i)$, implicating that the decided values, which are faces of σ, are totally ordered by containment. ☐

5 The Solvability Theorem

In this section we present our main theorem. We first clarify which tasks are deemed trivial, and later characterize the solvability of non-trivial Byzantine colorless tasks.

Informally, a task is trivial if all non-faulty processes can choose an output directly from their own inputs, without communicating with any other processes. Formally, $(\mathcal{I}, \mathcal{O}, \Delta)$ is trivial if there is a *simplicial* map $\delta : \mathcal{I} \to \mathcal{O}$ carried by Δ. In non-trivial tasks, non-faulty processes cannot decide solely based on their own inputs: some non-faulty process must become aware of inputs from other non-faulty processes. Our Byzantine model requires $t + 1$ processes granting the existence of the input before it is "learned":

Definition 5.1. A process P *learns* a value v if it receives $t+1$ messages $m_1, \ldots, m_{t+1}$ from different processes, with m_i granting p_i as having input v ($p_i \neq p_j$ when $i \neq j$).

We now proceed with our main solvability theorem.

Theorem 5.2. A non-trivial colorless task $(\mathcal{I}, \mathcal{O}, \Delta)$ has a t-resilient protocol in the asynchronous Byzantine model if and only if

1. $n + 1 > t(\dim(\mathcal{I}) + 2)$ and
2. there is a continuous map $f : |\operatorname{skel}^t(\mathcal{I})| \to |\mathcal{O}|$ carried by Δ.

Proof. **Conditions imply protocol.** Say that the map f exists. By the simplicial approximation theorem [20, 18] (shown in Theorem 2.2), f has a *simplicial approximation* $\phi : \operatorname{Bary}^N \operatorname{skel}^t(\mathcal{I}) \to \mathcal{O}$, for some $N > 0$, also carried by Δ. The protocol for non-faulty processes is shown below, presuming that $n + 1 > t(\dim(\mathcal{I}) + 2)$ with $\dim(\mathcal{I}) > 0$.

1. Execute the Byzantine k-set agreement protocol, for $k = t+1$, choosing vertices on a simplex in $\operatorname{skel}^t(\mathcal{I})$.
2. Execute the Byzantine barycentric agreement protocol N times, choosing vertices on a simplex in $\operatorname{Bary}^N \operatorname{skel}^t(\mathcal{I})$.
3. Apply $\phi : \operatorname{Bary}^N \operatorname{skel}^t(\mathcal{I}) \to \mathcal{O}$ to choose vertices on a simplex in $\mathcal{O}$.

Since ϕ and f are carried by Δ, non-faulty processes starting on vertices of $\sigma_I \in \mathcal{I}$ finish on vertices of $\sigma_O \in \Delta(\sigma)$. Also, since $1 \leq \dim(\sigma_I) \leq \dim(\mathcal{I})$, by definition, the preconditions are satisfied for calling the protocols in steps (1) and (2).

Protocol implies conditions. Say that the protocol exists.

Condition (1). Assume, for the sake of contradiction, that $n+1 \leq t(\dim(\mathcal{I})+2)$. Consider an execution where Byzantine processes never send messages, and the $(n+1) - t$ non-faulty processes start on vertices of an arbitrary $\sigma \in \mathcal{I}$, with $\dim(\sigma) = \dim(\mathcal{I})$. Furthermore, say that less than $t+1$ non-faulty processes input any particular vertex of σ, since $(n+1) - t \leq t(\dim(\mathcal{I})+1) = t|\sigma|$. However, no process can learn any value in this case, by Definition 5.1, and since we presumed a protocol solving $(\mathcal{I}, \mathcal{O}, \Delta)$, we must have a *simplicial* map $\delta : \mathcal{I} \to \mathcal{O}$ carried by Δ. The map contradicts the fact that $(\mathcal{I}, \mathcal{O}, \Delta)$ is non-trivial, so we must have $n + 1 > t(\dim(\mathcal{I}) + 2)$.

Condition (2). We argue by reduction to the crash-failure case. Note that any t-resilient Byzantine protocol is also a t-resilient crash-failure protocol for colorless tasks. From [12, 15][6], for

[6] These papers characterize connectivity in terms of c, the minimum core size, as defined by Junqueira and Marzullo [17]. For t-resilient tasks in the crash-failure model, $t = c + 1$. See also Appendix C.

any $\sigma \in \mathcal{I}$, the protocol complex $\mathcal{P}(\sigma)$ is $(t-1)$-connected in the crash-failure model, so, in light of the previous observation, it is also $(t-1)$ connected in the Byzantine-failure model. This implicates that $\mathrm{skel}^x(\mathcal{P}(\sigma))$ is $(x-1)$-connected for $0 \leq x \leq t$. We will inductively construct a sequence of continuous maps $g_x : |\,\mathrm{skel}^x(\mathcal{I})| \to |\mathcal{P}(\mathrm{skel}^x(\mathcal{I}))|$, considering $0 \leq x \leq t$, mapping skeletons of $\mathcal{I}$ to skeletons of $\mathcal{P}(\mathcal{I})$.

Base. Let g_0 map any vertex $v \in \sigma$ to any vertex $v' \in \mathcal{P}(v)$, which exists because $\mathrm{skel}^x(\mathcal{P}(v))$ is (-1)-connected by hypothesis. We just constructed $g_0 : |\,\mathrm{skel}^0(\mathcal{I})| \to |\mathcal{P}(\mathrm{skel}^0(\mathcal{I}))|$.

Induction Hypothesis. Assume

$$g_{x-1} : |\,\mathrm{skel}^{x-1}(\mathcal{I})| \to |\mathcal{P}(\mathrm{skel}^{x-1}(\mathcal{I}))|,$$

with $x \leq t$, sending the geometrical boundary of a x-simplex σ^x in $\mathrm{skel}^x(\mathcal{I})$ to $|\mathcal{P}(\mathrm{skel}^{x-1}(\sigma^x))|$. Formally, $g_{x-1}(|\,\partial\,\sigma^x|) \subseteq |\mathcal{P}(\mathrm{skel}^{x-1}(\sigma^x))|$. By hypothesis, $\mathcal{P}(\sigma^x)$ is x-connected, so the continuous image of the $(x-1)$-sphere $|\,\partial\,\sigma^x|$ could be extended to a continuous x-disk $|\sigma^x|$, defining g_x such that $g_x(|\sigma^x|) \subseteq |\mathcal{P}(\mathrm{skel}^x(\mathcal{I}))|$. As all such maps agree on their intersections, in light of definition 2.4, we just constructed

$$g_x : |\,\mathrm{skel}^x(\mathcal{I})| \to |\mathcal{P}(\mathrm{skel}^x(\mathcal{I}))| \subseteq |\mathcal{P}(\mathcal{I})|.$$

As we assumed a protocol solving $(\mathcal{I}, \mathcal{O}, \Delta)$, we have a simplicial map $\delta : \mathcal{P}(I) \to \mathcal{O}$ carried by Δ (given by definition 2.5). Our map is induced by the composition $g_t \circ \delta$. For details, on the induced composition, see [20, 18]. $\square$

We note that the barycentric agreement, which we employed in the "conditions imply protocol" part of the previous proof, can be similarly accomplished using a multidimensional approximate agreement protocol such as [19]. For the interested reader, we discuss such approach in Appendix D.

6 Conclusion

In this work, we give the first necessary and sufficient conditions for the solvability of non-trivial colorless tasks in asynchronous Byzantine systems. This particular characterization is interesting *per se*, as previous, analogous results exist for crash-failure systems [16]. However, this work also evidences (i) the power and suitability of the combinatorial topology tools, and (ii) novel, yet simple algorithms for the fundamental k-set agreement and barycentric agreement problems in asynchronous Byzantine systems.

We saw how combinatorial topology tools can produce novel results in distributed computing by virtue of being capable to support existential arguments without complicated, model-specific constructive arguments. Indeed, we used or slightly adapted basic communication primitives (reliable broadcast, quorums, stable vectors) and straightforward algorithmic tools (k-set agreement and barycentric agreement) to demonstrate our solvability theorem, relying on powerful, model-independent observations from combinatorial topology to complement the proof.

Our necessary and sufficient conditions capture the relation between the number of processes, the number of failures, and the topological structure of the task's simplicial complexes. This differs from the case of crash-failure systems, and opens similar questions for other scenarios, such as synchronous systems and colored tasks, which we intend to carry as future work.

A Algorithms for Reliable Broadcast

Algorithms 6 to 8 show the reliable broadcast protocol for sender P, round r, and content c. The symbol "$\cdot$" represents a wildcard, matching any value.

Algorithm 6 P.`RBSend`$((P, r, c))$

```
send$(P, r, c)$ to all processes
```

Algorithm 7 P.`RBEcho`$()$

```
upon recv$(Q, r, c)$ from $Q$ do
    if never sent $(Q, r\{\texttt{echo}\}, \cdot)$ then
        send$(Q, r\{\texttt{echo}\}, c)$ to all processes
upon recv$(\cdot, r\{\texttt{echo}\}, c)$ from $\geq (n+1) - t$ processes do
    if never sent $(P, r\{\texttt{ready}\}, \cdot)$ then
        send$(P, r\{\texttt{ready}\}, c)$ to all processes
upon recv$(\cdot, r\{\texttt{ready}\}, c)$ from $\geq t+1$ processes do
    if never sent $(P, r\{\texttt{ready}\}, \cdot)$ then
        send$(P, r\{\texttt{ready}\}, c)$ to all processes
```

Algorithm 8 P.`RBRecv`$((P, r, c))$

```
recv$(\cdot, r\{\texttt{ready}\}, c)$ from $(n+1) - t$ processes
return $(P, r, c)$
```

B Proofs for Communication Primitives

In this section, we prove lemmas related to quorum and stable vectors. The proofs on stable vector guarantees are variations from those in [2].

B.1 Proof of Property 3.2

Proof. Since $n+1 > 3t$, the processes can perform reliable broadcast. Note that the $(n+1)-t$ messages sent by the non-faulty processes can be grouped by their contents:

$$(n+1) - t = \sum_{c \in Good(\mathcal{M}^r)} |\{(P, r, c) : (P, r, c) \in \mathcal{M}^r, P \text{ is non-faulty}\}|.$$

If every content c in $Good(\mathcal{M}^r)$ was reliably broadcast by at most t non-faulty processes, we would have $(n+1) - t \leq |Good(\mathcal{M}^r)| \cdot t$, which contradicts the hypothesis. Hence, at least one content in $Good(\mathcal{M}^r)$ was reliably broadcast by more than $t+1$ non-faulty processes. By the non-faulty liveness of the reliable broadcast, such content will eventually be reliably received by all non-faulty processes. $\square$

B.2 Proof of Property 3.3

Proof. If $|M_i^r \setminus M_j^r| > t$, then M_j^r missed more than t messages in $\mathcal{M}^r$, the messages reliably broadcast in round r. However, this contradicts the fact that $|M_j^r| \geq (n+1) - t$ with M_j^r being obtained by reliable broadcast. $\square$

B.3 Proof of Lemma 3.4

Proof. First, note that the set M of P_i is monotonically increasing, so the sequence of transmitted reports is also monotonically increasing. As we assume FIFO channels, any other non-faulty process P_j receives those reports in the same order. $\square$

B.4 Proof of Lemma 3.5

Proof. Consider $\mathcal{S}$, the set of all messages reliably received by at least one non-faulty process. By the non-faulty liveness of the reliable broadcast, all non-faulty processes will eventually obtain $M = \mathcal{S}$, which is never expanded, or we would contradict the definition of $\mathcal{S}$. All non-faulty processes then send reports $\mathcal{R} = \mathcal{S}$, which are final, for the same reason as before.

Again, by the non-faulty liveness property of the reliable broadcast, all these processes will eventually receive $(n+1) - t$ reports $\mathcal{R}$. By the monotonicity of received reports (Lemma 3.4) and FIFO message delivery, those reports are not overwritten, matching the local $M = \mathcal{S}$. All non-faulty processes then return from `RecvStable`(). $\square$

B.5 Proof of Lemma 3.6

Proof. Call B_i^r the set of buddies whose reports are stored in R, for process P_i and round r, right before it decides. Since all reports are transmitted via reliable broadcast, and every non-faulty process collects $(n+1) - t$ reports, $|B_i^r \setminus B_j^r| \leq t$ with $|B_i^r| \geq (n+1) - t$, which implies that $|B_i^r \cap B_j^r| \geq n + 1 - 2t$. In other words, any two non-faulty processes identify $n + 1 - 2t > t + 1$ buddies in common, including a non-faulty process P_k. Therefore, $M_i^r = R_k'$ and $M_j^r = R_k''$, where R_k' and R_k'' are reports sent by P_k at possibly different occasions.

Since the set M_k^r is monotonically increasing, either $R_k' \subseteq R_k''$ or $R_k'' \subseteq R_k'$, guaranteeing property (iii). Both R_k' and R_k'' contain R_k, the first report sent by P_k, by Lemma 3.4. Finally, $|R_k| \geq (n+1) - t$ and $\mathit{Quorum}(R_k) \neq \emptyset$, by Lemma 3.2, which guarantee properties (i) and (ii). $\square$

C Solvability for Non-Independent Faults

The t-resilient model presumes independent, identically distributed process failures, however in practice they correlate to the same processor, machine, etc. The core/survivor-set model from Junqueira and Marzullo [17] assumes an adversarial scheduler that defines sets of possibly faulty/correct processes. Although we provided our results in the more well-known uniform model, our algorithms and theorem **do apply** to such extended failure model, as we discuss below.

In the model, a *core* is a minimal set of processes that will never fail together in any admissible execution. The minimum core size is denoted as c, and processes can always wait for $(n+1)-(c-1)$ messages. Note that t-resilient tasks have $c = t + 1$.

In the non-uniform failure model, we redefine the *variable* t to be $c - 1$, one less than the minimum core size, in the reliable broadcast, quorum, and stable vector protocols, As any process can always wait for $(n+1)-(c-1)=(n+1)-t$ messages, our correctness arguments for these protocols remain identical, and therefore algorithms 1 and 2 allow us to solve c-set agreement and barycentric agreement, respectively.

We here overview the changes in the proof of our solvability theorem between its principal formulation, and in the one following the model of Junqueira and Marzullo [17]. The overall proof structure is identical to the one for Theorem 5.2 – we highlight only changes to make it work in the non-uniform failure model, which are trivial and short in nature.

Theorem C.1. A non-trivial colorless task $(\mathcal{I},\mathcal{O},\Delta)$ has a protocol in the asynchronous Byzantine model with core size c if and only if $n+1>(c-1)\cdot(\dim(\mathcal{I})+2)$ and there is a continuous map $f:|\operatorname{skel}^{c-1}(\mathcal{I})|\to|\mathcal{O}|$ carried by Δ.

Proof. **Conditions imply protocol.** Assuming the map f exists, proceed as before, but use the c-set agreement protocol to jump to $\operatorname{skel}^{c-1}(\mathcal{I})$. The application of the barycentric agreement protocol and of the simplicial approximation theorem remain identical.

Protocol implies conditions. For the first condition, the argument remains identical, as we still presume non-trivial tasks. The second condition also preserves our argument, as [12, 15] actually define the connectivity of the protocol complexes in terms of c: for any $\sigma\in\mathcal{I}$, we have that $\mathcal{P}(\sigma)$ is $(c-2)$-connected. As we defined $c=t+1$, the argument remains identical. □

D Barycentric Agreement via Approximate Agreement

In this section, we show how to transform the multi-dimensional approximate agreement protocol of [19] into barycentric agreement. The point-set occupied by $\mathcal{I}$ is compact, and the open stars of the vertices of $\operatorname{Bary}\mathcal{I}$ form an open cover of $\mathcal{I}$. Any such cover has a *Lebesgue* number $\lambda>0$ [20], such that every set of diameter less than λ is contained in some member of the cover.

Here is a Byzantine barycentric agreement protocol. Suppose the non-faulty processes start at the vertices of an input simplex σ. Using $(\lambda/2)$-approximate agreement, each non-faulty process p_i chooses a point inside σ, the convex hull of the inputs, such that the distance between any pair of points is less than $\lambda/2$. Equivalently, each open ball of radius $\lambda/2$ around v_i contains all values chosen by the approximate agreement protocol. Because the diameter of this set is less than the Lebesgue number λ, there is at least one vertex u_i in $\operatorname{Bary}\mathcal{I}$ such that $B(v_i,\lambda/2)$ lies in the open star around u_i. Let each P_i choose any such u_i.

We must still show that the vertices u_i chosen by the processes P_i lie on a single simplex of $\operatorname{Bary}\sigma$. Note that u_i,u_j are vertices of a common simplex if and only if the open star around u_i intersects the open star around u_j. By construction, $v_j\in B(v_i,\lambda/2)$, which is in the open star around u_i, and v_j is in the open star around u_j, hence u_i,u_j are vertices of a single simplex.

References

[1] I. Abraham, Y. Amit, and D. Dolev. Optimal resilience asynchronous approximate agreement. In *Proceedings of the 8th international conference on Principles of Distributed Systems*, OPODIS'04, pages 229–239, Berlin, Heidelberg, 2005. Springer-Verlag.

[2] H. Attiya, A. Bar-Noy, D. Dolev, D. Peleg, and R. Reischuk. Renaming in an Asynchronous Environment. *Journal of the ACM*, July 1990.

[3] H. Attiya and J. Welch. *Distributed Computing Fundamentals, Simulations, and Advanced TopicsSecond Edition.* John Wiley and Sons, Inc., 2004.

[4] E. Borowsky and E. Gafni. Generalized FLP impossibility result for t-resilient asynchronous computations. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 91–100, New York, NY, USA, 1993. ACM.

[5] E. Borowsky, E. Gafni, N. Lynch, and S. Rajsbaum. The BG distributed simulation algorithm. *Distributed Computing*, 14(3):127–146, 2001.

[6] G. Bracha. Asynchronous byzantine agreement protocols. *Information and Computation*, 75(2), 1987.

[7] C. Cachin, R. Guerraoui, and L. Rodrigues. *Introduction to Reliable and Secure Distributed Programming.* Springer, 2nd ed. 2011 edition, Feb. 2011.

[8] S. Chaudhuri. More choices allow more faults: Set consensus problems in totally asynchronous systems. *Information and Computation*, 105(1):132–158, July 1993.

[9] M. Fischer, N. A. Lynch, and M. S. Paterson. Impossibility Of Distributed Commit With One Faulty Process. *Journal of the ACM*, 32(2), Apr. 1985.

[10] M. Herlihy and S. Rajsbaum. Algebraic spans. *Mathematical Structures in Computer Science*, 10(4):549–573, 2000.

[11] M. Herlihy and S. Rajsbaum. A classification of wait-free loop agreement tasks. *Theor. Comput. Sci.*, 291(1):55–77, 2003.

[12] M. Herlihy and S. Rajsbaum. Concurrent computing and shellable complexes. In N. Lynch and A. Shvartsman, editors, *Distributed Computing*, volume 6343 of *Lecture Notes in Computer Science*, pages 109–123. Springer Berlin Heidelberg, 2010.

[13] M. Herlihy and S. Rajsbaum. The topology of shared-memory adversaries. In *Proceeding of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, PODC '10, pages 105–113, New York, NY, USA, 2010. ACM.

[14] M. Herlihy and S. Rajsbaum. Simulations and reductions for colorless tasks. In *Proceedings of the 2012 ACM symposium on Principles of distributed computing*, PODC '12, pages 253–260, New York, NY, USA, 2012. ACM.

[15] M. Herlihy, S. Rajsbaum, and M. Tuttle. An Axiomatic Approach to Computing the Connectivity of Synchronous and Asynchronous Systems. *Electron. Notes Theor. Comput. Sci.*, 230:79–102, 2009.

[16] M. Herlihy and N. Shavit. The topological structure of asynchronous computability. *J. ACM*, 46(6):858–923, 1999.

[17] F. P. Junqueira and K. Marzullo. Designing Algorithms for Dependent Process Failures. Technical report, 2003.

[18] D. N. Kozlov. *Combinatorial Algebraic Topology*, volume 21 of *Algorithms and Computation in Mathematics*. Springer, 1 edition, Oct. 2007.

[19] H. Mendes and M. Herlihy. Multidimensional approximate agreement in byzantine asynchronous systems. In *STOC '13: Proceedings of the fourty-fifth annual ACM symposium on Theory of computing*, to appear, June 2013.

[20] J. Munkres. *Elements of Algebraic Topology.* Prentice Hall, 2 edition, Jan. 1984.

[21] M. Saks and F. Zaharoglou. Wait-Free k-Set Agreement is Impossible: The Topology of Public Knowledge. *SIAM J. Comput.*, 29(5):1449–1483, 2000.

[22] T. Srikanth and S. Toueg. Simulating authenticated broadcasts to derive simple fault-tolerant algorithms. *Distributed Computing*, 2, 1987.