The Impact of Topology on Byzantine Containment in Stabilization

Swan Dubois[*][†] Toshimitsu Masuzawa[‡] Sébastien Tixeuil[§]

Abstract

Self-stabilization is an versatile approach to fault-tolerance since it permits a distributed system to recover from any transient fault that arbitrarily corrupts the contents of all memories in the system. Byzantine tolerance is an attractive feature of distributed system that permits to cope with arbitrary malicious behaviors.

We consider the well known problem of constructing a maximum metric tree in this context. Combining these two properties prove difficult: we demonstrate that it is impossible to contain the impact of Byzantine nodes in a self-stabilizing context for maximum metric tree construction (strict stabilization). We propose a weaker containment scheme called *topology-aware strict stabilization*, and present a protocol for computing maximum metric trees that is optimal for this scheme with respect to impossibility result.

Keywords Byzantine fault, Distributed protocol, Fault tolerance, Stabilization, Spanning tree construction

1 Introduction

The advent of ubiquitous large-scale distributed systems advocates that tolerance to various kinds of faults and hazards must be included from the very early design of such systems. *Self-stabilization* [3, 5, 14] is a versatile technique that permits forward recovery from any kind of *transient* faults, while *Byzantine Fault-tolerance* [10] is traditionally used to mask the effect of a limited number of *malicious* faults. Making distributed systems tolerant to both transient and malicious faults is appealing yet proved difficult [6, 2, 12] as impossibility results are expected in many cases.

Two main paths have been followed to study the impact of Byzantine faults in the context of self-stabilization:

- *Byzantine fault masking.* In completely connected synchronous systems, one of the most studied problems in the context of self-stabilization with Byzantine faults is that of *clock synchronization*. In [1, 6], probabilistic self-stabilizing protocols were proposed for up to one third of Byzantine processes, while in [4, 9] deterministic solutions tolerate up to one fourth and one third of Byzantine processes, respectively.

[*]Université Pierre et Marie Curie & INRIA, France, swan.dubois@lip6.fr

[†]Contact author, Telephone: 33 1 44 27 87 67, Postal address: LIP6, Case 26/00-225, Campus Jussieu, 4 place Jussieu, 75252 Paris Cedex 5, France

[‡]Osaka University, Japan, masuzawa@ist.osaka-u.ac.jp

[§]Université Pierre et Marie Curie & INRIA, France, sebastien.tixeuil@lip6.fr

- *Byzantine containment.* For *local* tasks (*i.e.* tasks whose correctness can be checked locally, such as vertex coloring, link coloring, or dining philosophers), the notion of *strict stabilization* was proposed [12, 13, 11]. Strict stabilization guarantees that there exists a *containment radius* outside which the effect of permanent faults is masked, provided that the problem specification makes it possible to break the causality chain that is caused by the faults. As many problems are not local, it turns out that it is impossible to provide strict stabilization for those.

Our Contribution. In this paper, we investigate the possibility of Byzantine containment in a self-stabilizing setting for tasks that are global (*i.e.* for with there exists a causality chain of size r, where r depends on n the size of the network), and focus on a global problem, namely maximum metric tree construction (see [7, 8]). As strict stabilization is impossible with such global tasks, we weaken the containment constraint by relaxing the notion of containment radius to containment area, that is Byzantine processes may disturb infinitely often a set of processes which depends on the topology of the system and on the location of Byzantine processes.

The main contribution of this paper is to present new possibility results for containing the influence of unbounded Byzantine behaviors. In more details, we define the notion of *topology-aware strict stabilization* as the novel form of the containment and introduce *containment area* to quantify the quality of the containment. The notion of topology-aware strict stabilization is weaker than the strict stabilization but is stronger than the classical notion of self-stabilization (*i.e.* every topology-aware strictly stabilizing protocol is self-stabilizing, but not necessarily strictly stabilizing).

To demonstrate the possibility and effectiveness of our notion of topology-aware strict stabilization, we consider *maximum metric tree construction*. It is shown in [12] that there exists no strictly stabilizing protocol with a constant containment radius for this problem. In this paper, we provide a topology-aware strictly stabilizing protocol for maximum metric tree construction and we prove that the containment area of this protocol is optimal.

2 Distributed System

A *distributed system* $S = (P, L)$ consists of a set $P = \{v_1, v_2, \ldots, v_n\}$ of processes and a set L of bidirectional communication links (simply called links). A link is an unordered pair of distinct processes. A distributed system S can be regarded as a graph whose vertex set is P and whose link set is L, so we use graph terminology to describe a distributed system S.

Processes u and v are called *neighbors* if $(u, v) \in L$. The set of neighbors of a process v is denoted by N_v, and its cardinality (the *degree* of v) is denoted by $\Delta_v (= |N_v|)$. The degree Δ of a distributed system $S = (P, L)$ is defined as $\Delta = \max\{\Delta_v \mid v \in P\}$. We do not assume existence of a unique identifier for each process. Instead we assume each process can distinguish its neighbors from each other by locally arranging them in some arbitrary order: the k-th neighbor of a process v is denoted by $N_v(k)$ $(1 \leq k \leq \Delta_v)$. The *distance* between two processes u and v is the length of the shortest path between u and v.

In this paper, we consider distributed systems of arbitrary topology. We assume that a single process is distinguished as a *root*, and all the other processes are identical.

We adopt the *shared state model* as a communication model in this paper, where each process can directly read the states of its neighbors.

The variables that are maintained by processes denote process states. A process may take actions during the execution of the system. An action is simply a function that is executed in an atomic manner by the process. The actions executed by each process is described by a finite set of guarded actions of the form $\langle \text{guard} \rangle \longrightarrow \langle \text{statement} \rangle$. Each guard of process u is a boolean expression involving the variables of u and its neighbors.

A global state of a distributed system is called a *configuration* and is specified by a product of states of all processes. We define C to be the set of all possible configurations of a distributed system S. For a process set $R \subseteq P$ and two configurations ρ and ρ', we denote $\rho \overset{R}{\mapsto} \rho'$ when ρ changes to ρ' by executing an action of each process in R simultaneously. Notice that ρ and ρ' can be different only in the states of processes in R. For completeness of execution semantics, we should clarify the configuration resulting from simultaneous actions of neighboring processes. The action of a process depends only on its state at ρ and the states of its neighbors at ρ, and the result of the action reflects on the state of the process at ρ'.

A *schedule* of a distributed system is an infinite sequence of process sets. Let $Q = R^1, R^2, \ldots$ be a schedule, where $R^i \subseteq P$ holds for each i ($i \geq 1$). An infinite sequence of configurations $e = \rho_0, \rho_1, \ldots$ is called an *execution* from an initial configuration ρ_0 by a schedule Q, if e satisfies $\rho_{i-1} \overset{R^i}{\mapsto} \rho_i$ for each i ($i \geq 1$). Process actions are executed atomically, and we also assume that a *distributed daemon* schedules the actions of processes, i.e. any subset of processes can simultaneously execute their actions.

The set of all possible executions from $\rho_0 \in C$ is denoted by E_{ρ_0}. The set of all possible executions is denoted by E, that is, $E = \bigcup_{\rho \in C} E_\rho$. We consider *asynchronous* distributed systems where we can make no assumption on schedules except that any schedule is *weakly fair*: every process is contained in infinite number of subsets appearing in any schedule.

In this paper, we consider (permanent) *Byzantine faults*: a Byzantine process (i.e. a Byzantine-faulty process) can make arbitrary behavior independently from its actions. If v is a Byzantine process, v can repeatedly change its variables arbitrarily.

3 Self-Stabilizing Protocol Resilient to Byzantine Faults

Problems considered in this paper are so-called *static problems*, i.e. they require the system to find static solutions. For example, the spanning-tree construction problem is a static problem, while the mutual exclusion problem is not. Some static problems can be defined by a *specification predicate* (shortly, specification), $spec(v)$, for each process v: a configuration is a desired one (with a solution) if every process satisfies $spec(v)$. A specification $spec(v)$ is a boolean expression on variables of P_v ($\subseteq P$) where P_v is the set of processes whose variables appear in $spec(v)$. The variables appearing in the specification are called *output variables* (shortly, *O-variables*). In what follows, we consider a static problem defined by specification $spec(v)$.

Self-Stabilization. A *self-stabilizing protocol* ([3]) is a protocol that eventually reaches a *legitimate configuration*, where $spec(v)$ holds at every process v, regardless of the initial configuration. Once it reaches a legitimate configuration, every process never changes its O-variables and always satisfies $spec(v)$. From this definition, a self-stabilizing protocol is expected to tolerate any number and any type of transient faults since it can eventually recover from any configuration affected by the transient faults. However, the recovery from any configuration is guaranteed only when every process correctly executes its action from the configuration, i.e., we do not consider existence of permanently faulty processes.

Strict stabilization. When (permanent) Byzantine processes exist, Byzantine processes may not satisfy *spec*(v). In addition, correct processes near the Byzantine processes can be influenced and may be unable to satisfy *spec*(v). Nesterenko and Arora [12] define a *strictly stabilizing protocol* as a self-stabilizing protocol resilient to unbounded number of Byzantine processes.

Given an integer c, a *c-correct process* is a process defined as follows.

Definition 1 (c-correct process) *A process is c-correct if it is correct (*i.e. *not Byzantine) and located at distance more than c from any Byzantine process.*

Definition 2 ((c, f)-containment) *A configuration ρ is* (c, f)-contained *for specification spec if, given at most f Byzantine processes, in any execution starting from ρ, every c-correct process v always satisfies spec(v) and never changes its O-variables.*

The parameter c of Definition 2 refers to the *containment radius* defined in [12]. The parameter f refers explicitly to the number of Byzantine processes, while [12] dealt with unbounded number of Byzantine faults (that is $f \in \{0 \dots n\}$).

Definition 3 ((c, f)-strict stabilization) *A protocol is* (c, f)-strictly stabilizing *for specification spec if, given at most f Byzantine processes, any execution $e = \rho_0, \rho_1, \dots$ contains a configuration ρ_i that is (c, f)-contained for spec.*

An important limitation of the model of [12] is the notion of *r-restrictive* specifications. Intuitively, a specification is r-restrictive if it prevents combinations of states that belong to two processes u and v that are at least r hops away. An important consequence related to Byzantine tolerance is that the containment radius of protocols solving those specifications is at least r. For some problems, such as the spanning tree construction we consider in this paper, r can not be bounded to a constant. We can show that there exists no $(o(n), 1)$-strictly stabilizing protocol for the spanning tree construction.

Topology-aware strict stabilization. In the former paragraph, we saw that there exist a number of impossibility results on strict stabilization due to the notion of r-restrictive specifications. To circumvent this impossibility result, we define here a new notion, which is weaker than the strict stabilization: the *topology-aware strict stabilization* (denoted by TA-strict stabilization for short). Here, the requirement to the containment radius is relaxed, *i.e.* the set of processes which may be disturbed by Byzantines ones is not reduced to the union of c-neighborhood of Byzantines processes but can be defined depending on the topology of the system and on Byzantine processes location.

In the following, we give formal definition of this new kind of Byzantine containment. From now, B denotes the set of Byzantine processes and S_B (which is function of B) denotes a subset of V (intuitively, this set gathers all processes which may be disturbed by Byzantine processes).

Definition 4 (S_B-correct node) *A node is* S_B-correct *if it is a correct node (*i.e. *not Byzantine) which not belongs to S_B.*

Definition 5 (S_B-legitimate configuration) *A configuration ρ is* S_B-legitimate *for spec if every S_B-correct node v is legitimate for spec (*i.e. *if spec(v) holds).*

Definition 6 ((S_B, f)-topology-aware containment) *A configuration ρ_0 is* (S_B, f)-topology-aware contained *for specification spec if, given at most f Byzantine processes, in any execution $e = \rho_0, \rho_1, \dots$, every configuration is S_B-legitimate and every S_B-correct process never changes its O-variables.*

The parameter S_B of Definition 6 refers to the *containment area*. Any process which belongs to this set may be infinitely disturbed by Byzantine processes. The parameter f refers explicitly to the number of Byzantine processes.

Definition 7 ((S_B, f)-topology-aware strict stabilization) *A protocol is* (S_B, f)-topology-aware strictly stabilizing *for specification spec if, given at most f Byzantine processes, any execution* $e = \rho_0, \rho_1, \ldots$ *contains a configuration* ρ_i *that is* (S_B, f)*-topology-aware contained for spec.*

Note that, if B denotes the set of Byzantine processes and $S_B = \{v \in V | min\{d(v, b), b \in B\} \leq c\}$, then a (S_B, f)-topology-aware strictly stabilizing protocol is a (c, f)-strictly stabilizing protocol. Then, a TA-strictly stabilizing protocol is generally weaker than a strictly stabilizing one, but stronger than a classical self-stabilizing protocol (that may never meet its specification in the presence of Byzantine processes).

The parameter S_B is introduced to quantify the strength of fault containment, we do not require each process to know the actual definition of the set. Actually, the protocol proposed in this paper assumes no knowledge on this parameter.

4 Maximum Metric Tree Construction

In this work, we deal with maximum (routing) metric trees as defined in [8] (note that [7] provides a self-stabilizing solution to this problem). Informally, the goal of a routing protocol is to construct a tree that simultaneously maximizes the metric values of all of the nodes with respect to some total ordering $\prec$. In the following, we recall all definitions and notations introduced in [8].

Definition 8 (Routing metric) *A* routing metric *(or just* metric*) is a five-tuple* $(M, W, met, mr, \prec)$ *where:*

1. *M is a set of metric values,*
2. *W is a set of edge weights,*
3. *met is a metric function whose domain is $M \times W$ and whose range is M,*
4. *mr is the maximum metric value in M with respect to $\prec$ and is assigned to the root of the system,*
5. *$\prec$ is a less-than total order relation over M that satisfies the following three conditions for arbitrary metric values m, m', and m'' in M:*
   (a) *irreflexivity: $m \not\prec m$,*
   (b) *transitivity : if $m \prec m'$ and $m' \prec m''$ then $m \prec m''$,*
   (c) *totality: $m \prec m'$ or $m' \prec m$ or $m = m'$.*

Any metric value $m \in M \setminus \{mr\}$ satisfies the utility condition *(that is, there exists $w_0, \ldots, w_{k-1}$ in W and $m_0 = mr, m_1, \ldots, m_{k-1}, m_k = m$ in M such that $\forall i \in \{1, \ldots, k\}, m_i = met(m_{i-1}, w_{i-1})$).*

For instance, we provide the definition of three classical metrics with this model: the shortest path metric ($\mathcal{SP}$), the flow metric ($\mathcal{F}$), and the reliability metric ($\mathcal{R}$).

$$\begin{array}{rcl} \mathcal{SP} & = & (M_1, W_1, met_1, mr_1, \prec_1) \\ \text{where} & & M_1 = \mathbb{N} \\ & & W_1 = \mathbb{N} \\ & & met_1(m,w) = m + w \\ & & mr_1 = 0 \\ & & \prec_1 \text{ is the classical } > \text{ relation} \end{array} \qquad \begin{array}{rcl} \mathcal{F} & = & (M_2, W_2, met_2, mr_2, \prec_2) \\ \text{where} & & mr_2 \in \mathbb{N} \\ & & M_2 = \{0, \dots, mr_2\} \\ & & W_2 = \{0, \dots, mr_2\} \\ & & met_2(m,w) = min\{m,w\} \\ & & \prec_2 \text{ is the classical } < \text{ relation} \end{array}$$

$$\begin{array}{rcl} \mathcal{R} & = & (M_3, W_3, met_3, mr_3, \prec_3) \\ \text{where} & & M_3 = [0,1] \\ & & W_3 = [0,1] \\ & & met_3(m,w) = m * w \\ & & mr_3 = 1 \\ & & \prec_3 \text{ is the classical } < \text{ relation} \end{array}$$

Definition 9 (Assigned metric) *An* assigned metric *over a system S is a six-tuple $(M, W, met, mr, \prec, wf)$ where $(M, W, met, mr, \prec)$ is a metric and wf is a function that assigns to each edge of S a weight in W.*

Let a rooted path (from v) be a simple path from a process v to the root r. The next set of definitions are with respect to an assigned metric $(M, W, met, mr, \prec, wf)$ over a given system S.

Definition 10 (Metric of a rooted path) *The* metric of a rooted path *in S is the prefix sum of met over the edge weights in the path and mr.*

For example, if a rooted path p in S is $v_k, \dots, v_0$ with $v_0 = r$, then the metric of p is $m_k = met(m_{k-1}, wf(\{v_k, v_{k-1}\})$ with $\forall i \in \{1, k-1\}, m_i = met(m_{i-1}, wf(\{v_i, v_{i-1}\})$ and $m_0 = mr$.

Definition 11 (Maximum metric path) *A rooted path p from v in S is called a* maximum metric path *with respect to an assigned metric if and only if for every other rooted path q from v in S, the metric of p is greater than or equal to the metric of q with respect to the total order $\prec$.*

Definition 12 (Maximum metric of a node) *The* maximum metric of a node $v \neq r$ *(or simply* metric value *of v) in S is defined by the metric of a maximum metric path from v. The maximum metric of r is mr.*

Definition 13 (Maximum metric tree) *A spanning tree T of S is a* maximum metric tree *with respect to an assigned metric over S if and only if every rooted path in T is a maximum metric path in N with respect to the assigned metric.*

The goal of the work of [8] is the study of metrics that always allow the construction of a maximum metric tree. More formally, the definition follow.

Definition 14 (Maximizable metric) *A metric is* maximizable *if and only if for any assignment of this metric over any system S, there is a maximum metric tree for S with respect to the assigned metric.*

Note that [7] provides a self-stabilizing protocol to construct a maximum metric tree with respect to any maximizable metric. Moreover, [8] provides a fully characterization of maximizable metrics as follow.

Definition 15 (Boundedness) *A metric $(M, W, met, mr, \prec)$ is* bounded *if and only if: $\forall m \in M, \forall w \in W, met(m,w) \prec m$ or $met(m,w) = m$*

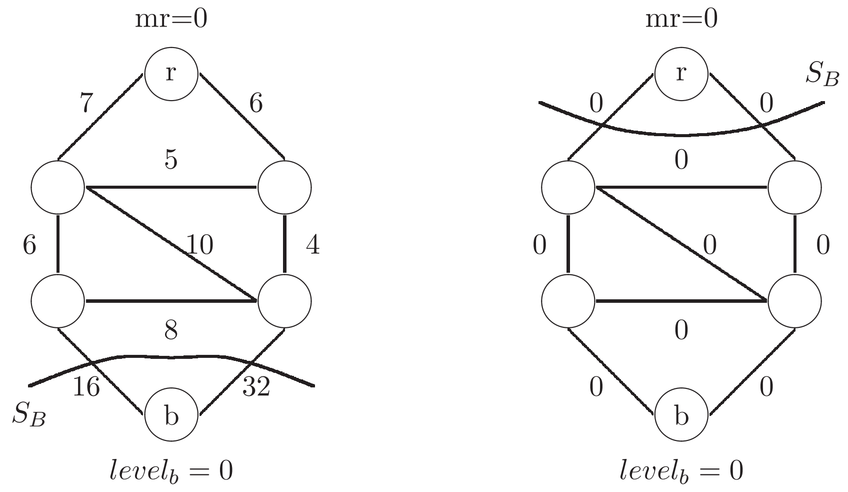


Figure 1: Examples of containment areas for SP spanning tree construction.

Definition 16 (Monotonicity) *A metric* $(M, W, met, mr, \prec)$ *is* monotonic *if and only if:* $\forall (m, m') \in M^2, \forall w \in W, m \prec m' \Rightarrow (met(m, w) \prec met(m', w)$ *or* $met(m, w) = met(m', w))$

Theorem 1 (Characterization of maximizable metrics [8]) *A metric is maximizable if and only if this metric is bounded and monotonic.*

Given a maximizable metric $\mathcal{M} = (M, W, mr, met, \prec)$, the aim of this work is to construct a maximum metric tree with respect to $\mathcal{M}$ which spans the system in a self-stabilizing way in a system subject to permanent Byzantine failures. It is obvious that these Byzantine processes may disturb some correct processes. It is why, we relax the problem in the following way: we want to construct a maximum metric forest with respect to $\mathcal{M}$. The root of any tree of this forest must be either the real root or a Byzantine process.

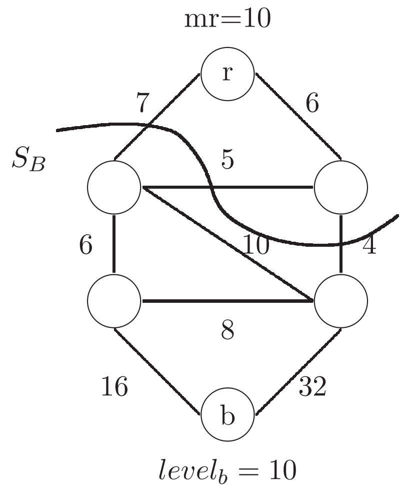
Each process v has three O-variables: a pointer to its parent in its tree ($prnt_v \in N_v \cup \{\perp\}$), a level which stores its current metric value ($level_v \in M$), and a variable which stores its distance to the root of its tree ($dist_v \in \{0, \ldots, D\}$). Obviously, Byzantine process may disturb (at least) their neighbors. We use the following specification of the problem.

We introduce new notations as follows. Given an assigned metric $(M, W, met, mr, \prec, wf)$ over the system S and two processes u and v, we denote by $\mu(u, v)$ the maximum metric of node u when v plays the role of the root of the system and by $w_{u,v}$ the weight of the edge $\{u, v\}$ (that is, the value of $wf(\{u, v\})$).

Definition 17 ($\mathcal{M}$-path) *Given an assigned metric* $\mathcal{M} = (M, W, mr, met, \prec, wf)$ *over a system* S*, a path* $(v_0, \ldots, v_k)$ $(k \geq 1)$ *of* S *is a* $\mathcal{M}$-path *if and only if:*

1. $prnt_{v_0} = \perp$, $level_{v_0} = 0$, $dist_{v_0} = 0$, *and* $v_0 \in B \cup \{r\}$,

2. $\forall i \in \{1, \ldots, k\}, prnt_{v_i} = v_{i-1}$, $level_{v_i} = met(level_{v_{i-1}}, w_{v_i, v_{i-1}})$, *and* $dist_{v_i} = i$,



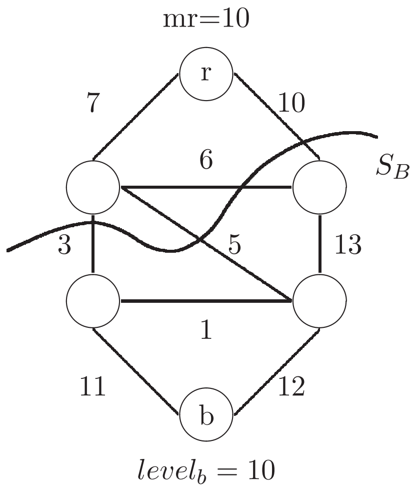


Figure 2: Examples of containment areas for flow spanning tree construction.

3. $\forall i \in \{1, \dots, k\}, met(level_{v_{i-1}}, w_{v_i,v_{i-1}}) = \underset{u \in N_v}{max_{\prec}}\{met(level_u, w_{v_i,u})\}$, *and*

4. $level_{v_k} = \mu(v_k, v_0)$.

We define the specification predicate $spec(v)$ of the maximum metric tree construction with respect to a maximizable metric $\mathcal{M}$ as follows.

$$spec(v) : \begin{cases} prnt_v = \bot, level_v = 0, \text{ and } dist_v = 0 \text{ if } v \text{ is the root } r \\ \text{there exists a } \mathcal{M}\text{-path } (v_0, \dots, v_k) \text{ such that } v_k = v \text{ otherwise} \end{cases}$$

Following discussion of Section 3, it is obvious that there exists no strictly stabilizing protocol for this problem. It is why we consider the weaker notion of topology-aware strict stabilization. First, we show an impossibility result in order to define the best possible containment area. Then, we provide a maximum metric tree construction protocol which is (S_B, f)-TA-strictly stabilizing where $f \leq n-1$ which match these optimal containment area, namely:

$$S_B = \{v \in V \setminus B \,|\mu(v,r) \preceq max_{\prec}\{\mu(v,b), b \in B\}\} \setminus \{r\}$$

Figures from 1 to 3 provide some examples of containment areas with respect to several maximizable metrics.

We introduce here a new definition that is used in the following.

Definition 18 (Fixed point) *A metric value m is a* fixed point *of a metric $\mathcal{M} = (M, W, mr, met, \prec)$ if $m \in M$ and if for any value $w \in W$, we have: $met(m, w) = m$.*

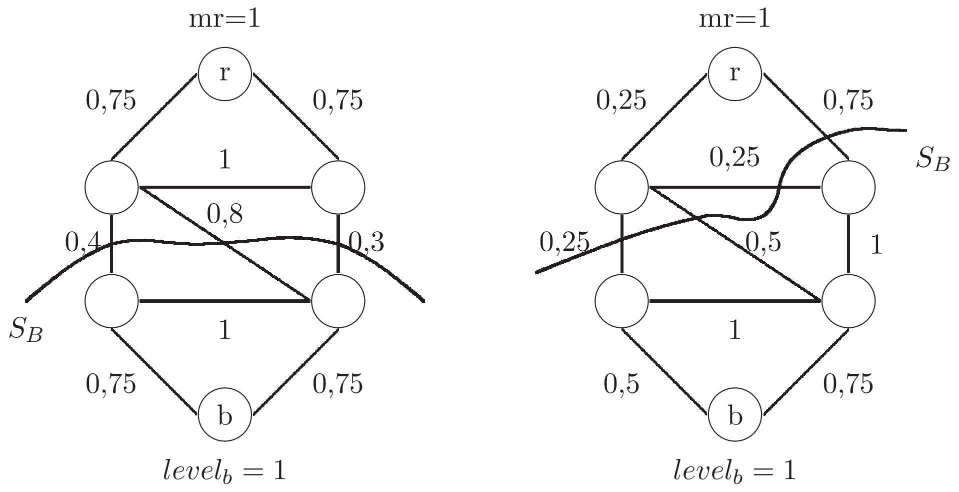


Figure 3: Examples of containment areas for reliability spanning tree construction.

4.1 Impossibility Result

In this section, we show that there exists some constraints on the containment area of any topology-aware strictly stabilizing for the maximum metric tree construction depending on the metric.

Theorem 2 *Given a maximizable metric $\mathcal{M} = (M, W, mr, met, \prec)$, even under the central daemon, there exists no $(A_B, 1)$-TA-strictly stabilizing protocol for maximum metric spanning tree construction with respect to $\mathcal{M}$ where $A_B \subsetneq S_B$.*

Proof Let $\mathcal{M} = (M, W, mr, met, \prec)$ be a maximizable metric and $\mathcal{P}$ be a $(A_B, 1)$-TA-strictly stabilizing protocol for maximum metric spanning tree construction protocol with respect to $\mathcal{M}$ where $A_B \subsetneq S_B$. We must distinguish the following cases:

Case 1: $|M| = 1$.

Denote by m the metric value such that $M = \{m\}$. For any system and for any process $v \neq r$, we have $\mu(v, r) = \underset{b \in B}{min_{\prec}}\{\mu(v, b)\} = m$. Consequently, $S_B = V \setminus (B \cup \{r\})$ for any system.

Consider the following system: $V = \{r, u, v, b\}$ and $E = \{\{r, u\}, \{u, v\}, \{v, b\}\}$ (b is a Byzantine process). As $S_B = \{u, v\}$ and $A_B \subsetneq S_B$, we have: $u \notin A_B$ or $v \notin A_B$. Consider now the following configuration ρ_0^0: $prnt_r = prnt_b = \bot$, $prnt_v = b$, $prnt_u = v$, $level_r = level_u = level_v = level_b = m$, $dist_r = dist_b = 0$, $dist_v = 1$ and $dist_u = 2$ (see Figure 4, other variables may have arbitrary values). Note that ρ_0^0 is A_B-legitimate for *spec* (whatever A_B is).

Assume now that b behaves as a correct process with respect to $\mathcal{P}$. Then, by convergence of $\mathcal{P}$ in a fault-free system starting from ρ_0^0 which is not legitimate (remember that a strictly-stabilizing protocol is a special case of self-stabilizing protocol), we can deduce that the system reaches in a finite time a configuration ρ_1^0 (see Figure 4) in which: $prnt_r = \bot$, $prnt_u = r$,

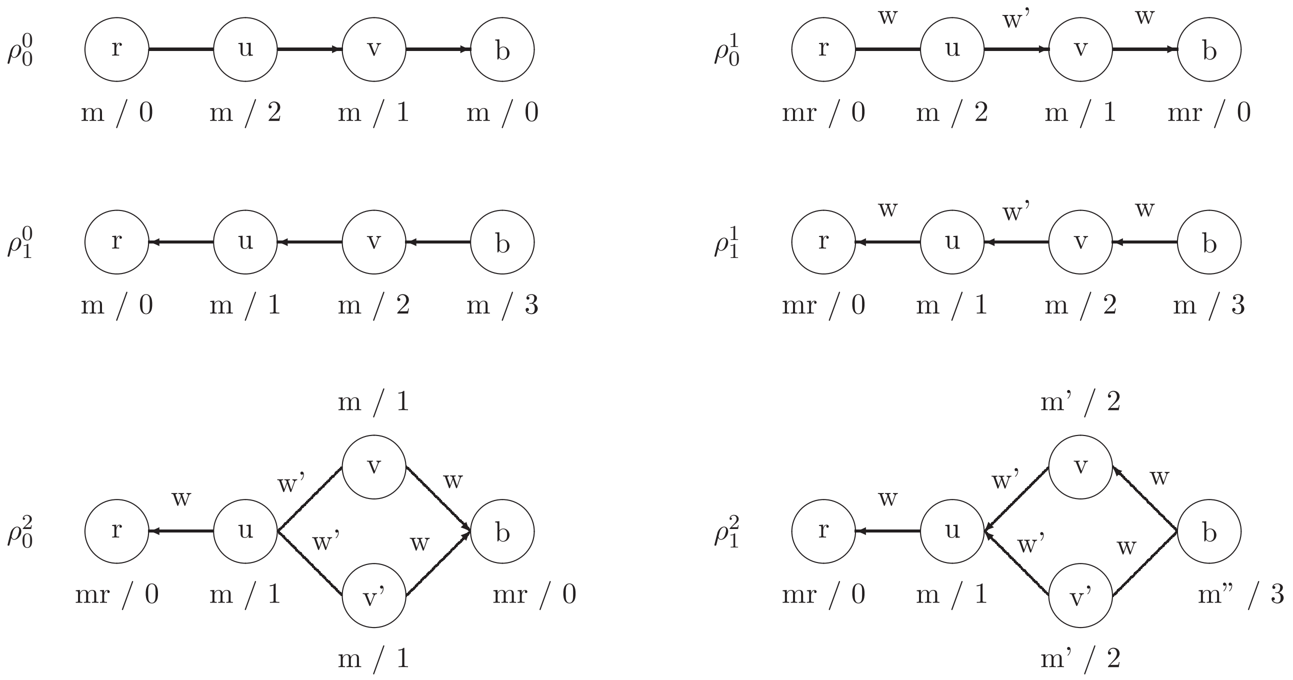


Figure 4: Configurations used in proof of Theorem 2.

$prnt_v = u$, $prnt_b = v$, $level_r = level_u = level_v = level_b = m$, $dist_r = 0$, $dist_u = 1$, $dist_v = 2$ and $dist_b = 3$. Note that processes u and v modify their O-variables in this execution. This contradicts the $(A_B, 1)$-TA-strict stabilization of $\mathcal{P}$ (whatever A_B is).

Case 2: $|M| \geq 2$.
By definition of a bounded metric, we can deduce that there exist $m \in M$ and $w \in W$ such that $m = met(mr, w) \prec mr$. Then, we must distinguish the following cases:

Case 2.1: m is a fixed point of $\mathcal{M}$.
Consider the following system: $V = \{r, u, v, b\}$, $E = \{\{r, u\}, \{u, v\}, \{v, b\}\}$, $w_{r,u} = w_{v,b} = w$, and $w_{u,v} = w'$ (b is a Byzantine process). As for any $w' \in W$, $met(m, w') = m$ (by definition of a fixed point), we have: $S_B = \{u, v\}$. Since $A_B \subsetneq S_B$, we have: $u \notin A_B$ or $v \notin A_B$. Consider now the following configuration ρ_0^1: $prnt_r = prnt_b = \bot$, $prnt_v = b$, $prnt_u = v$, $level_r = level_b = mr$, $level_u = level_v = m$, $dist_r = dist_b = 0$, $dist_v = 1$ and $dist_u = 2$ (see Figure 4, other variables may have arbitrary values). Note that ρ_0^1 is A_B-legitimate for *spec* (whatever A_B is).

Assume now that b behaves as a correct process with respect to $\mathcal{P}$. Then, by convergence of $\mathcal{P}$ in a fault-free system starting from ρ_0^1 which is not legitimate (remember that a strictly-stabilizing protocol is a special case of self-stabilizing protocol), we can deduce that the system reaches in a finite time a configuration ρ_1^1 (see Figure 4) in which: $prnt_r = \bot$, $prnt_u = r$, $prnt_v = u$, $prnt_b = v$, $level_r = mr$, $level_u = level_v = level_b = m$ (since m is a fixed point), $dist_r = 0$, $dist_u = 1$, $dist_v = 2$ and $dist_b = 3$. Note that processes u and v modify their O-variables in this execution. This contradicts the $(A_B, 1)$-TA-strict stabilization of $\mathcal{P}$ (whatever A_B is).

Case 2.2: m is not a fixed point of $\mathcal{M}$.

This implies that there exists $w' \in W$ such that: $met(m, w') \prec m$ (remember that $\mathcal{M}$ is bounded). Consider the following system: $V = \{r, u, v, v', b\}$, $E = \{\{r, u\}, \{u, v\}, \{u, v'\}, \{v, b\}, \{v', b\}\}$, $w_{r,u} = w_{v,b} = w_{v',b} = w$, and $w_{u,v} = w_{u,v'} = w'$ (b is a Byzantine process). We can see that $S_B = \{v, v'\}$. Since $A_B \subsetneq S_B$, we have: $v \notin A_B$ or $v' \notin A_B$. Consider now the following configuration ρ_0^2: $prnt_r = prnt_b = \bot$, $prnt_v = prnt_{v'} = b$, $prnt_u = r$, $level_r = level_b = mr$, $level_u = level_v = level_{v'} = m$, $dist_r = dist_b = 0$, $dist_v = dist_{v'} = 1$ and $dist_u = 1$ (see Figure 4, other variables may have arbitrary values). Note that ρ_0^2 is A_B-legitimate for *spec* (whatever A_B is).

Assume now that b behaves as a correct process with respect to $\mathcal{P}$. Then, by convergence of $\mathcal{P}$ in a fault-free system starting from ρ_0^2 which is not legitimate (remember that a strictly-stabilizing protocol is a special case of self-stabilizing protocol), we can deduce that the system reaches in a finite time a configuration ρ_1^2 (see Figure 4) in which: $prnt_r = \bot$, $prnt_u = r$, $prnt_v = prnt_{v'} = u$, $prnt_b = v$ (or $prnt_b = v'$), $level_r = mr$, $level_u = m$ $level_v = level_{v'} = met(m, w') = m'$, $level_b = met(m', w) = m''$, $dist_r = 0$, $dist_u = 1$, $dist_v = dist_{v'} = 2$ and $dist_b = 3$. Note that processes v and v' modify their O-variables in this execution. This contradicts the $(A_B, 1)$-TA-strict stabilization of $\mathcal{P}$ (whatever A_B is).

□

4.2 Topology-Aware Strict Stabilizing Protocol

In this section, we provide our self-stabilizing protocol that achieve optimal containment areas to permanent Byzantine failures for constructing a maximum metric tree for any maximizable metric $\mathcal{M} = (M, W, met, mr, \prec)$. More formally, our protocol is (S_B, f)-strictly stabilizing, that is optimal with respect to the result of Theorem 2. Our protocol is borrowed from the one of [7] (which is self-stabilizing). The key idea of this protocol is to use the distance variable (upper bounded by a given constant D) to detect and break cycles of process which has the same maximum metric. The main modification we bring to this protocol follows. In the initial protocol, when a process modifies its parent, it chooses arbitrarily one of the "better" neighbors (with respect to the metric). To achieve the (S_B, f)-TA-strict stabilization, we must ensures a fair selection along the set of its neighbor. We perform this fairness with a round-robin order along the set of neighbors. Our solution is presented as Algorithm 4.1.

In the following, we provide the proof of the TA-strict stabilization of $\mathcal{SSMAX}$. Remember that the real root r can not be a Byzantine process by hypothesis. Note that the subsystem whose set of nodes is $V \setminus S_B$ is connected respectively by boundedness of the metric.

Lemma 1 *For any process $v \in V$, we have:*

$$\forall u \in N_v, met\left(\underset{p \in B \cup \{r\}}{max_\prec}\{\mu(u, p)\}, w_{u,v}\right) \preceq \underset{p \in B \cup \{r\}}{max_\prec}\{\mu(v, p)\}$$

algorithm 4.1 $\mathcal{SSMAX}$: A TA-strictly stabilizing protocol for maximum metric tree construction.

```
Data:
    $N_v$: totally ordered set of neighbors of $v$.
    $D$: upper bound of the number of processes in a simple path.

Variables:
    $prnt_v \begin{cases} = \bot \text{ if } v = r \\ \in N_v \text{ if } v \neq r \end{cases}$ : pointer on the parent of $v$ in the tree.
    $level_v \in \{m \in M | m \preceq mr\}$: metric of the node.
    $dist_v \in \{0, \dots, D\}$: distance to the root.

Macro:
    For any subset $A \subseteq N_v$, $choose(A)$ returns the first element of $A$ which is bigger than $prnt_v$ (in a round-robin fashion).

Rules:
    $(\boldsymbol{R_r})$ :: $(v = r) \wedge ((level_v \neq mr) \vee (dist_v \neq 0)) \longrightarrow level_v := mr;\ dist_v := 0$
    $(\boldsymbol{R_1})$ :: $(v \neq r) \wedge (prnt_v \in N_v) \wedge ((dist_v \neq min(dist_{prnt_v} + 1, D)) \vee (level_v \neq met(level_{prnt_v}, w_{v,prnt_v})))$
            $\longrightarrow dist_v := min(dist_{prnt_v} + 1, D); level_v := met(level_{prnt_v}, w_{v,prnt_v})$
    $(\boldsymbol{R_2})$ :: $(v \neq r) \wedge (dist_v = D) \wedge (\exists u \in N_v, dist_u < D - 1)$
            $\longrightarrow prnt_v := choose(\{u \in N_v | dist_v < D - 1\});\ dist_v := dist_{prnt_v} + 1;\ level_v := met(level_{prnt_v}, w_{v,prnt_v})$
    $(\boldsymbol{R_3})$ :: $(v \neq r) \wedge (\exists u \in N_v, (dist_u < D - 1) \wedge (level_v \prec met(level_u, w_{u,v})))$
            $\longrightarrow prnt_v := choose\left(\left\{u \in N_v \middle| (level_u < D-1) \wedge (met(level_u, w_{u,v}) = \underset{q \in N_v / level_q < D-1}{max_\prec} \{met(level_q, w_{q,v})\})\right\}\right);$
            $level_v := met(level_{prnt_v}, w_{prnt_v,v});\ dist_v := dist_{prnt_v} + 1$
```

Proof Let $v \in V$ be a process. By contradiction, assume that there exists a neighbor u of v such that:

$$\underset{p \in B \cup \{r\}}{max_\prec} \{\mu(v,p)\} \prec met\left(\underset{p \in B \cup \{r\}}{max_\prec} \{\mu(u,p)\}, w_{u,v}\right)$$

Let $q \in B \cup \{r\}$ one of the process such that $\underset{p \in B \cup \{r\}}{max_\prec} \{\mu(u,p)\} = \mu(u,q)$. Then, we have:

$$\begin{array}{rcll} \underset{p \in B \cup \{r\}}{max_\prec} \{\mu(v,p)\} & \prec & met(\mu(u,q), w_{u,v}) & \text{by construction of } q \\ & \prec & \mu(v,q) & \text{since } met(\mu(u,q), w_{u,v}) \preceq \mu(v,q) \end{array}$$

This contradicts the fact that $q \in B \cup \{r\}$ and shows us the result. □

Given a configuration $\rho \in C$ and a metric value $m \in M$, let us define the following predicate:

$$IM_m(\rho) \equiv \forall v \in V, level_v \preceq max_\prec \left\{ m, \underset{u \in B \cup \{r\}}{max_\prec} \{\mu(v,u)\} \right\}$$

Lemma 2 *For any metric value $m \in M$, the predicate IM_m is closed by actions of $\mathcal{SSMAX}$.*

Proof Let m be a metric value ($m \in M$). Let $\rho \in C$ be a configuration such that $IM_m(\rho) = true$ and $\rho' \in C$ be a configuration such that $\rho \overset{R}{\mapsto} \rho'$ is a step of $\mathcal{SSMAX}$.

If the root process $r \in R$ (respectively a Byzantine process $b \in R$), then we have $level_r = mr$ (respectively $level_b \preceq mr$) in ρ' by construction of $(\boldsymbol{R_r})$ (respectively by definition of $level_b$). Hence,

$level_r \preceq max_{\prec}\left\{m,\ \underset{u\in B\cup\{r\}}{max_{\prec}}\{\mu(r,u)\}\right\} = mr$ (respectively $level_b \preceq max_{\prec}\left\{m,\ \underset{u\in B\cup\{r\}}{max_{\prec}}\{\mu(b,u)\}\right\} \preceq mr$).

If a correct process $v \in R$ with $v \neq r$, then there exists a neighbor p of v such that $level_p \preceq max_{\prec}\left\{m,\ \underset{u\in B\cup\{r\}}{max_{\prec}}\{\mu(p,u)\}\right\}$ in ρ (since $IM(\rho) = true$) and $prnt_v = p$ and $level_v = met(level_p, w_{v,p})$ in ρ' (since v is activated during this step).

If we apply the Lemma 1 to met and to neighbor p, we obtain the following property:

$$met\left(\underset{u\in B\cup\{r\}}{max_{\prec}}\{\mu(p,u)\}, w_{v,p}\right) \preceq \underset{u\in B\cup\{r\}}{max_{\prec}}\{\mu(v,u)\}$$

Consequently, we obtain that, in ρ':

$$\begin{array}{rcll} level_v & = & met(level_p, w_{v,p}) & \\ & \preceq & met\left(max_{\prec}\left\{m,\ \underset{u\in B\cup\{r\}}{max_{\prec}}\{\mu(p,u)\}\right\}, w_{v,p}\right) & \text{by boundedness of } \mathcal{M} \\ & \preceq & max_{\prec}\left\{met(m, w_{v,p}), met\left(\underset{u\in B\cup\{r\}}{max_{\prec}}\{\mu(p,u)\}, w_{v,p}\right)\right\} & \\ & \preceq & max_{\prec}\left\{m,\ \underset{u\in B\cup\{r\}}{max_{\prec}}\{\mu(v,u)\}\right\} & \text{since } met(m, w_{v,p}) \preceq m \end{array}$$

We can deduce that $IM_d(\rho') = true$, that concludes the proof. □

Given an assigned metric to a system G, we can observe that the set of metrics value M is finite and that we can label elements of M by $m_0 = mr, m_1, \ldots, m_k$ in a way such that $\forall i \in \{0, \ldots, k-1\}, m_{i+1} \prec m_i$.

We introduce the following notations:

$$\begin{array}{rrcl} \forall m_i \in M, & P_{m_i} & = & \left\{v \in V \setminus S_B \middle| \mu(v,r) = m_i\right\} \\ \forall m_i \in M, & V_{m_i} & = & \bigcup_{j=0}^{i} P_{m_j} \\ \forall m_i \in M, & I_{m_i} & = & \left\{v \in V \middle| \underset{u\in B\cup\{r\}}{max_{\prec}}\{\mu(v,u)\} \prec m_i\right\} \\ \forall m_i \in M, & \mathcal{LC}_{m_i} & = & \left\{\rho \in \mathcal{C} \middle| (\forall v \in V_{m_i}, spec(v)) \wedge (IM_{m_i}(\rho))\right\} \\ & \mathcal{LC} & = & \mathcal{LC}_{m_k} \end{array}$$

Lemma 3 *For any $m_i \in M$, the set $\mathcal{LC}_{m_i}$ is closed by actions of $\mathcal{SSMAX}$.*

Proof Let m_i be a metric value from M and ρ be a configuration of $\mathcal{LC}_{m_i}$. By construction, any process $v \in V_{m_i}$ satisfies $spec(v)$ in ρ.

In particular, the root process satisfies: $prnt_r = \perp$, $level_r = mr$, and $dist_r = 0$. By construction of $\mathcal{SSMAX}$, r is not enabled and then never modifies its O-variables (since the guard of the rule of r does not involve the state of its neighbors).

In the same way, any process $v \in V_{m_i}$ satisfies: $prnt_v \in N_v$, $level_v = met(level_{prnt_v}, w_{prnt_v,v})$, $dist_v = dist_{prnt_v} + 1$, and $level_v = \underset{u\in N_v}{max_{\prec}}\{met(level_u, w_{u,v})\}$. Note that, as $v \in V_{m_i}$ and $spec(v)$

holds in ρ, we have: $level_v = \mu(v,r) = \max_{\prec\, p\in B\cup\{r\}} \{\mu(v,p)\}$ and $dist_v \leq D-1$ by construction of D. Hence, process v is not enabled in ρ.

Assume that there exists a process $v \in V_{m_i}$ that take a step $\rho' \overset{R}{\mapsto} \rho''$ in an execution starting from ρ (without loss of generality, assume that v is the first process of $v \in V_{m_i}$ that takes a step in this execution). Then, we know that $v \neq r$. This activation implies that a neighbor $u \notin V_{m_i}$ (since v is the first process of V_{m_i} to take a step) of v modified its $level_u$ variable to a metric value $m \in M$ such that $level_v \prec met(m, w_{u,v})$ in ρ' (note that O-variables of v and $prnt_v$ remain consistent since v is the first process to take a step in this execution).

Hence, we have $level_v = \max_{\prec\, p\in B\cup\{r\}} \{\mu(v,p)\} \prec met(m, w_{u,v})$. Moreover, the closure of IM_B (established in Lemma 2) ensures us that $m \preceq \max_{\prec\, p\in B\cup\{r\}} \{\mu(u,p)\}$. By boundedness of $\mathcal{M}$, we can deduce that $met(m, w_{u,v}) \preceq met(\max_{\prec\, p\in B\cup\{r\}} \{\mu(u,p)\}, w_{u,v})$. Consequently, we obtain that $\max_{\prec\, p\in B\cup\{r\}} \{\mu(v,p)\} \prec met(\max_{\prec\, p\in B\cup\{r\}} \{\mu(u,p)\}, w_{u,v})$. This is contradictory with the result of Lemma 1.

In conclusion, any process $v \in V_{m_i}$ takes no step in any execution starting from ρ and then always satisfies $spec(v)$. Then, the closure of IM_B (established in Lemma 2) concludes the proof. □

Lemma 4 *Any configuration of $\mathcal{LC}$ is $(S_B, n-1)$-TA contained for spec.*

Proof This is a direct application of the Lemma 3 to $\mathcal{LC} = \mathcal{LC}_{m_i}$. □

Lemma 5 *Starting from any configuration of $\mathcal{C}$, any execution of $\mathcal{SSMAX}$ reaches in a finite time a configuration of $\mathcal{LC}_{mr}$.*

Proof Let ρ be an arbitrary configuration. Then, it is obvious that $IM_{mr}(\rho)$ is satisfied. By closure of IM_{mr} (proved in Lemma 2), we know that IM_{mr} remains satisfied in any execution starting from ρ.

If r does not satisfy $spec(r)$ in ρ, then r is continuously enabled. Since the scheduling is weakly fair, r is activated in a finite time and then r satisfies $spec(r)$ in a finite time. Denote by ρ' the first configuration in which $spec(r)$ holds. Note that r takes no step in any execution starting from ρ'.

The boundedness of $\mathcal{M}$ implies that P_{mr} induces a connected subsystem. If $P_{mr} = \{r\}$, then we proved that $\rho' \in \mathcal{LC}_{mr}$ and we have the result.

Otherwise, observe that, for any configuration of an execution starting from ρ', if all processes of P_{mr} are not enabled, then all processes v of P_{mr} satisfy $spec(v)$. Assume now that there exists an execution e starting from ρ' in which some processes of P_{mr} takes infinitely many steps. By construction, at least one of these processes (note it v) has a neighbor u which takes only a finite number of steps in e (recall that P_{mr} induces a connected subsystem and that r takes no step in e). After u takes its last step of e, we can observe that $level_u = mr$ and $dist_u < D-1$ (otherwise, u is activated in a finite time that contradicts its construction).

As v can execute consequently $\boldsymbol{(R_1)}$ only a finite number of times (since the incrementation of $dist_v$ is bounded by D), we can deduce that v executes $\boldsymbol{(R_2)}$ or $\boldsymbol{(R_3)}$ infinitely often. In both cases, u belongs to the set which is the parameter of function *choose*. By the fairness of this function, we can deduce that $prnt_v = u$ in a finite time in e. Then, the construction of u implies that v is never enabled in the sequel of e. This is contradictory with the construction of e.

Consequently, any execution starting from ρ' reaches in a finite time a configuration such that all processes of P_{mr} are not enabled. We can deduce that this configuration belongs to $\mathcal{LC}_{mr}$, that ends the proof. □

Lemma 6 *For any $m_i \in M$ and for any configuration $\rho \in \mathcal{LC}_{m_i}$, any execution of $\mathcal{SSMAX}$ starting from ρ reaches in a finite time a configuration such that:*

$$\forall v \in I_{m_i}, level_v = m_i \Rightarrow dist_v = D$$

Proof Let m_i be an arbitrary metric value of M and ρ_0 be an arbitrary configuration of $\mathcal{LC}_{m_i}$. Let $e = \rho_0, \rho_1, \dots$ be an execution starting from ρ_0.

Note that ρ_0 satisfies IM_{m_i} by construction. Hence, we have $\forall v \in I_{m_i}, level_v \preceq m_i$. The closure of IM_{m_i} (proved in Lemma 2) ensures us that this property is satisfied in any configuration of e.

If any process $v \in I_{m_i}$ satisfies $level_v \prec m_i$ in ρ_0, then the result is obvious. Otherwise, we define the following variant function. For any configuration ρ_j of e, we denote by A_j the set of processes v of I_{m_i} such that $level_v = m_i$ in ρ_j. Then, we define $f(\rho_j) = \underset{v \in A_j}{min}\{dist_v\}$. We will prove the result by showing that there exists an integer k such that $f(\rho_k) = D$.

First, if a process v joins A_j (that is, $v \notin A_{j-1}$ but $v \in A_j$), then it takes a distance value greater or equals to $f(\rho_{j+1})$ by construction of the protocol. We can deduce that the fact that some processes join A_j does not decrease f. Moreover, the construction of the protocol implies that a process v such that $v \in A_j$ and $v \in A_{j+1}$ can not decrease its distance value in the step $\rho_j \mapsto \rho_{j+1}$.

Then, consider for a given configuration ρ_j a process $v \in A_j$ such that $dist_v = f(\rho_j) < D$. We distinguish the following cases:

Case 1: $level_v = met(level_{prnt_v}, w_{v,prnt_v})$

The fact that $v \in I_{m_i}$, the boundedness of $\mathcal{M}$ and the closure of IM_{m_i} imply that $prnt_v \in A_j$ (and, hence that $level_{prnt_v} = m_i$). Then, by construction of $f(\rho_j)$, we know that $dist_v \neq dist_{prnt_v} + 1$ (otherwise, we do not have $dist_v = f(\rho_j)$ since $prnt_v$ has a smaller distance value). Consequently, v is enabled by $(\boldsymbol{R_1})$ in ρ_j and $dist_v$ increase of at least 1 during the step $\rho_j \mapsto \rho_{j+1}$ if this rule is executed.

Case 2: $level_v \neq met(level_{prnt_v}, w_{v,prnt_v})$

The rule $(\boldsymbol{R_1})$ is then enabled for v. If this rule is executed during the step $\rho_j \mapsto \rho_{j+1}$, one of the two following sub cases appears.

Case 2.1: $met(level_{prnt_v}, w_{v,prnt_v}) \prec m_i$

Then, v does not belong to A_{j+1} by definition.

Case 2.2: $met(level_{prnt_v}, w_{v,prnt_v}) = m_i$

Remind that the closure of IM_{m_i} implies then that $level_{prnt_v} = m_i$. By construction of $f(\rho_j)$, we have $dist_{prnt_v} \geq f(\rho_j)$ in ρ_j. Then, we can see that $dist_v$ increases of at least 1 during the step $\rho_j \mapsto \rho_{j+1}$.

In all cases, v is enabled by $(\boldsymbol{R_1})$ in ρ_j and the execution of this rule either increases strictly $dist_v$ or removes v from A_{j+1}.

As I_{m_i} is finite and the scheduling is weakly fair, we can deduce that f increases in a finite time in any execution starting from ρ_j. By repeating the argument at most D times, we can deduce that e contains a configuration ρ_k such that $f(\rho_k) = D$, that shows the result. □

Lemma 7 *For any $m_i \in M$ and for any configuration $\rho \in \mathcal{LC}_{m_i}$ such that $\forall v \in I_{m_i}, level_v = m_i \Rightarrow dist_v = D$, any execution of $\mathcal{SSMAX}$ starting from ρ reaches in a finite time a configuration such that:*

$$\forall v \in I_{m_i}, level_v \prec m_i$$

Proof Let $m_i \in M$ be an arbitrary metric value and ρ_0 be a configuration of $\mathcal{LC}_{m_i}$ such that $\forall v \in I_{m_i}, level_v = m_i \Rightarrow dist_v = D$. Let $e = \rho_0, \rho_1, \ldots$ be an arbitrary execution starting from ρ_0.

For any configuration ρ_j of e, let us denote $E_{\rho_j} = \{v \in I_{m_i} | level_v = m_i\}$. By the closure of IM_{m_i} (which holds by definition in ρ_0) established in Lemma 2, we obtain the result if there exists a configuration ρ_j of e such that $E_{\rho_j} = \emptyset$.

If there exists some processes $v \in I_{m_i} \setminus E_{\rho_0}$ (and hence $level_v \prec m_i$) such that $prnt_v \in E_{\rho_0}$ and $met(level_{prnt_v}, w_{v,prnt_v}) = m_i$ in ρ_0, then we can observe that these processes are continuously enabled by $(\boldsymbol{R_1})$. As the scheduling is weakly fair, v activates this rule in a finite time and then, $level_v = m_i$ and $dist_v = D$. In other words, v joins E_{ρ_l} for a given integer l. We can conclude that there exists an integer k such that for any $v \in I_{m_i} \setminus E_{\rho_0}$, either $prnt_v \notin E_{\rho_k}$ or $met(level_{prnt_v}, w_{v,prnt_v}) \prec m_i$.

Then, we prove that, for any integer $j \geq k$, we have $E_{\rho_{j+1}} \subseteq E_{\rho_j}$. For the sake of contradiction, assume that there exists an integer $j \geq k$ and a process $v \in I_{m_i}$ such that $v \in E_{\rho_{j+1}}$ and $v \notin E_{\rho_j}$. Without loss of generality, assume that j is the smallest integer which performs these properties. Let us study the following cases:

Case 1: If v activates $(\boldsymbol{R_1})$ during the step $\rho_j \mapsto \rho_{j+1}$, then we know that $prnt_v \notin E_{\rho_j}$ in ρ_j (otherwise, we have a contradiction with the fact that $v \in E_{\rho_{j+1}}$). But in this case, we have: $level_{prnt_v} \prec m_i$. The boundedness of $\mathcal{M}$ implies that $level_v \prec m_i$ in ρ_{j+1} that contradicts the fact that $v \in E_{\rho_{j+1}}$.

Case 2: If v activates either $(\boldsymbol{R_2})$ or $(\boldsymbol{R_3})$ during the step $\rho_j \mapsto \rho_{j+1}$, then v chooses a new parent which has a distance smaller than $D-1$ in ρ_j. This implies that this new parent does not belongs to E_{ρ_j}. Then, we have $level_{prnt_v} \prec m_i$. The boundedness of $\mathcal{M}$ implies that $level_v \prec m_i$ in ρ_{j+1} that contradicts the fact that $v \in E_{\rho_{j+1}}$.

In the two cases, our claim is satisfied. In other words, there exists a point of the execution afterwards the set E can not grow (this implies that, if a process leave the set E, it is a definitive leaving).

Assume now that there exists a step $\rho_j \mapsto \rho_{j+1}$ (with $j \geq k$) such that a process $v \in E_{\rho_j}$ is activated. Observe that the closure of IM_{m_i} implies that v can not be activated by the rule $(\boldsymbol{R_3})$. If v activates $(\boldsymbol{R_1})$ during this step, then v modifies its level during this step (otherwise, we have a contradiction with the fact that $level_{prnt_v} = m_i \Rightarrow dist_v = D$). The closure of IM_{m_i} implies that v leaves the set E during this step. If v activates $(\boldsymbol{R_2})$ during this step, then v chooses a new parent which has a distance smaller than $D-1$ in ρ_j. This implies that this new parent does not belongs to E_{ρ_j}. Then, we have $level_{prnt_v} \prec m_i$. The boundedness of $\mathcal{M}$ implies that $level_v \prec m_i$ in ρ_{j+1}. In other words, if a process of E_{ρ_j} is activated during the step $\rho_j \mapsto \rho_{j+1}$, then it satisfies $v \notin E_{\rho_{j+1}}$.

Finally, observe that the construction of the protocol and the construction of the bound D ensures us that any process $v \in I_{m_i}$ such that $dist_v = D$ is activated in a finite time. In conclusion, we obtain that there exists an integer j such that $E_{\rho_j} = \emptyset$, that implies the result. □

Lemma 8 *For any $m_i \in M$ and for any configuration $\rho \in \mathcal{LC}_{m_i}$, any execution of $\mathcal{SSMAX}$ starting from ρ reaches in a finite time a configuration ρ' such that $IM_{m_{i+1}}$ holds.*

Proof This result is a direct consequence of Lemmas 6 and 7. □

Lemma 9 *For any $m_i \in M$ and for any configuration $\rho \in \mathcal{LC}_{m_i}$, any execution of $\mathcal{SSMAX}$ starting from ρ reaches in a finite time a configuration of $\mathcal{LC}_{m_{i+1}}$.*

Proof Let m_i be a metric value of M and ρ be an arbitrary configuration of $\mathcal{LC}_{m_i}$. We know by Lemma 8 that any execution starting from ρ reaches in a finite time a configuration ρ' such that $IM_{m_{i+1}}$ holds. By closure of IM and of $\mathcal{LC}_{m_i}$ (established respectively in Lemma 2 and 3), we know that any configuration of any execution starting from ρ' belongs to $\mathcal{LC}_{m_i}$ and satisfies $IM_{m_{i+1}}$.

We know that $V_{m_i} \neq \emptyset$ since $r \in V_{m_i}$ for any $i \geq 0$. Remind that $V_{m_{i+1}}$ is connected by the boundedness of $\mathcal{M}$. Then, we know that there exists at least one process p of $P_{m_{i+1}}$ which has a neighbor q in V_{m_i} such that $\mu(p,r) = met(\mu(q,r), w_{p,q})$. Moreover, Lemma 3 ensures us that any process of V_{m_i} takes no step in any executions tarting from ρ'.

Observe that, for any configuration of an execution starting from ρ', if all processes of $P_{m_{i+1}}$ are not enabled, then all processes v of $P_{m_{i+1}}$ satisfy *spec*(v). Assume now that there exists an execution e starting from ρ' in which some processes of $P_{m_{i+1}}$ take infinitely many steps. By construction, at least one of these processes (note it v) has a neighbor u such that $\mu(v,r) = met(\mu(u,r), w_{v,u})$ which takes only a finite number of steps in e (recall the construction of p). After u takes its last step of e, we can observe that $level_u = \mu(u,r)$ and $dist_u < D-1$ (otherwise, u is activated in a finite time that contradicts its construction).

As v can execute consequently $\boldsymbol{(R_1)}$ only a finite number of times (since the incrementation of $dist_v$ is bounded by D), we can deduce that v executes $\boldsymbol{(R_2)}$ or $\boldsymbol{(R_3)}$ infinitely often. In both cases, u belongs to the set which is the parameter of function *choose* (remind that $IM_{m_{i+1}}$ is satisfied and that u has the better possible metric along v's neighbors). By the construction of this function, we can deduce that $prnt_v = u$ in a finite time in e. Then, the construction of u implies that v is never enabled in the sequel of e. This is contradictory with the construction of e.

Consequently, any execution starting from ρ' reaches in a finite time a configuration such that all processes of $P_{m_{i+1}}$ are not enabled. We can deduce that this configuration belongs to $\mathcal{LC}_{m_{i+1}}$, that ends the proof. □

Lemma 10 *Starting from any configuration, any execution of $\mathcal{SSMAX}$ reaches a configuration of $\mathcal{LC}$ in a finite time.*

Proof Let ρ be an arbitrary configuration. We know by Lemma 5 that any execution starting from ρ reaches in a finite time a configuration of $\mathcal{LC}_{mr} = \mathcal{LC}_{m_0}$. Then, we can apply at most k times the result of Lemma 9 to obtain that any execution starting from ρ reaches in a finite time a configuration of $\mathcal{LC}_{m_k} = \mathcal{LC}$, that proves the result. □

Theorem 3 *$\mathcal{SSMAX}$ is a $(S_B, n-1)$-TA-strictly stabilizing protocol for spec.*

Proof This result is a direct consequence of Lemmas 4 and 10. □

Note that Theorem 2 ensures us that S_B is the optimal containment area for a topology-aware strictly stabilizing protocol for *spec*.

5 Conclusion

We introduced a new notion of Byzantine containment in self-stabilization: the topology-aware strict stabilization. This notion relaxes the constraint on the containment radius of the strict stabilization to a containment area. In other words, the set of correct processes which may be infinitely often disturbed by Byzantine processes is a function depending on the topology of the system and on the actual location of Byzantine processes. We illustrated the relevance of this notion by providing a topology-aware strictly stabilizing protocol for the maximum metric tree construction problem which does not admit strictly stabilizing solution. Moreover, our protocol performs the optimal containment area with respect to the topology-aware strict stabilization.

Our work raises some opening questions. Number of problems do not accept strictly stabilizing solution. Does any of them admit a topology-aware strictly stabilizing solution ? Is it possible to give a necessary and/or sufficient condition for a problem to admit a topology-aware strictly stabilizing solution ? What happens if we consider only bounded Byzantine behavior ?

References

[1] Michael Ben-Or, Danny Dolev, and Ezra N. Hoch. Fast self-stabilizing byzantine tolerant digital clock synchronization. In Rida A. Bazzi and Boaz Patt-Shamir, editors, *PODC*, pages 385–394. ACM, 2008.

[2] Ariel Daliot and Danny Dolev. Self-stabilization of byzantine protocols. In Ted Herman and Sébastien Tixeuil, editors, *Self-Stabilizing Systems*, volume 3764 of *Lecture Notes in Computer Science*, pages 48–67. Springer, 2005.

[3] Edsger W. Dijkstra. Self-stabilizing systems in spite of distributed control. *Commun. ACM*, 17(11):643–644, 1974.

[4] Danny Dolev and Ezra N. Hoch. On self-stabilizing synchronous actions despite byzantine attacks. In Andrzej Pelc, editor, *DISC*, volume 4731 of *Lecture Notes in Computer Science*, pages 193–207. Springer, 2007.

[5] S. Dolev. *Self-stabilization.* MIT Press, March 2000.

[6] Shlomi Dolev and Jennifer L. Welch. Self-stabilizing clock synchronization in the presence of byzantine faults. *J. ACM*, 51(5):780–799, 2004.

[7] Mohamed G. Gouda and Marco Schneider. Stabilization of maximal metric trees. In Anish Arora, editor, *WSS*, pages 10–17. IEEE Computer Society, 1999.

[8] Mohamed G. Gouda and Marco Schneider. Maximizable routing metrics. *IEEE/ACM Trans. Netw.*, 11(4):663–675, 2003.

[9] Ezra N. Hoch, Danny Dolev, and Ariel Daliot. Self-stabilizing byzantine digital clock synchronization. In Ajoy Kumar Datta and Maria Gradinariu, editors, *SSS*, volume 4280 of *Lecture Notes in Computer Science*, pages 350–362. Springer, 2006.

[10] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[11] Toshimitsu Masuzawa and Sébastien Tixeuil. Stabilizing link-coloration of arbitrary networks with unbounded byzantine faults. *International Journal of Principles and Applications of Information Science and Technology (PAIST)*, 1(1):1–13, December 2007.

[12] Mikhail Nesterenko and Anish Arora. Tolerance to unbounded byzantine faults. In *21st Symposium on Reliable Distributed Systems (SRDS 2002)*, page 22. IEEE Computer Society, 2002.

[13] Yusuke Sakurai, Fukuhito Ooshita, and Toshimitsu Masuzawa. A self-stabilizing link-coloring protocol resilient to byzantine faults in tree networks. In *Principles of Distributed Systems, 8th International Conference, OPODIS 2004*, volume 3544 of *Lecture Notes in Computer Science*, pages 283–298. Springer, 2005.

[14] Sébastien Tixeuil. *Algorithms and Theory of Computation Handbook, Second Edition*, chapter Self-stabilizing Algorithms, pages 26.1–26.45. Chapman & Hall/CRC Applied Algorithms and Data Structures. CRC Press, Taylor & Francis Group, November 2009.