The Impact of Topology on Byzantine Containment in Stabilization*

Swan Dubois[1], Toshimitsu Masuzawa[2], and Sébastien Tixeuil[1]

[1] LIP6 - UMR 7606 Université Pierre et Marie Curie - Paris 6 & INRIA, France
[2] Osaka University, Japan

Abstract. Self-stabilization is a 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 systems 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 proves 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* [1,2,3] is a versatile technique that permits forward recovery from any kind of *transient* faults, while *Byzantine Fault-tolerance* [4,5] 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 [6,7,8] 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* (any correct processes eventually satisfy its specification). In completely connected synchronous systems, one of the most studied problems in the context of self-stabilization with Byzantine faults is that of *clock*

* This work has been supported in part by ANR projects SHAMAN, ALADDIN, SPADES, by MEXT Global COE Program and by JSPS Grant-in-Aid for Scientific Research ((B) 22300009).

synchronization. In [9,6], probabilistic self-stabilizing protocols were proposed for up to one third of Byzantine processes, while in [10,11] deterministic solutions tolerate up to one fourth and one third of Byzantine processes, respectively.

- *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 [8,12,13,14]. 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. Note that a strictly stabilizing algorithm with a radius of 0 which runs on a completely connected system provides a masking approach.

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 which 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 [15,16]). 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 [8] 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* $S = (V, E)$ consists of a set $V = \{v_1, v_2, \ldots, v_n\}$ of processes and a set E 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 V and whose link set is E, so we use graph terminology to describe a distributed system S.

Processes u and v are called *neighbors* if $(u, v) \in E$. 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 $S = (V, E)$ is defined as $\Delta = \max\{\Delta_v \mid v \in V\}$. 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 not distinguishable.

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 are 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 $R \subseteq V$ 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 $R^i \subseteq V$ 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. A more constrained daemon is the *central* one which must choose only one enabled process at each step. Note that, as the central daemon allows executions that are also allowed under the distributed daemon, an impossibility result under the central daemon is stronger than one under the distributed one. In the same way, a possibility result under the distributed daemon is stronger than one under the central one.

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. In other words, a Byzantine process has always an enabled rule and the daemon arbitrarily chooses a new state for this process when this process is activated. If v is a Byzantine process, v can repeatedly change its variables arbitrarily. The only restriction we do on Byzantine processes is that the root process can never be Byzantine.

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 *local specification predicate* (shortly, specification), $spec(v)$, for each process v: a configuration is a desired one (with a solution) if every process $v \in V$ satisfies $spec(v)$ in this configuration. A specification $spec(v)$ is a boolean expression on variables of V_v ($\subseteq V$) where V_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 a local specification predicate.

Self-Stabilization. A *self-stabilizing protocol* ([1]) 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 [8] 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 [8]. The parameter f refers explicitly to the number of Byzantine processes, while [8] dealt with unbounded number of Byzantine faults (that is $f \in \{0 \ldots n\}$).

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 $e = \rho_0, \rho_1, \ldots$ contains a configuration ρ_i that is (c, f)-contained for spec.*

An important limitation of the model of [8] 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 any global problem, such as the spanning tree construction we consider in this paper, r can not be bounded to a constant. The results of [8] show us that there exists no $(o(n), 1)$-strictly stabilizing protocol for these problems and especially 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 Byzantine ones is not reduced to the union of c-neighborhood of Byzantine 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 a 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 does not belong 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 $e = \rho_0, \rho_1, \ldots$, 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 spanning trees as defined in [16] (note that [15] 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 [16], authors give a general definition of a routing metric and provide a characterization of *maximizable metrics*, that is metrics which always allow to construct a maximum (routing) metric spanning trees. In the following, we recall all definitions and notations introduced in [16].

Definition 8 (Routing metric). *A* routing metric *is a five-tuple* $(M, W, met, mr, \prec)$ *where:*
- M *is a set of metric values,*
- W *is a set of edge weights,*
- met *is a metric function whose domain is* $M \times W$ *and whose range is* M*,*
- mr *is the maximum metric value in* M *with respect to* $\prec$ *and is assigned to the root of the system,*
- $\prec$ *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:
  - *irreflexivity:* $m \not\prec m$,
  - *transitivity : if* $m \prec m'$ *and* $m' \prec m''$ *then* $m \prec m''$,
  - *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 exist* $w_0, \ldots, w_{k-1}$ *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 ($\mathcal{R}$).

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

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

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 $v_k, \ldots, v_0$ 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 S with respect to the assigned metric.*

The goal of the work of [16] is the study of metrics that always allow the construction of a maximum metric tree. More formally, the definition follows.

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 [15] provides a self-stabilizing protocol to construct a maximum metric tree with respect to any maximizable metric. Moreover, [16] provides a fully characterization of maximazable 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 $met(m,w) = m$*

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 [16]). *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 ($prnt_v \in N_v \cup \{\bot\}$), 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. If u and v are two neighor processes, we denote 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* $\mathcal{M}$-path *is a path* $(v_0, \ldots, v_k)$ $(k \geq 1)$ *such that: (i)* $prnt_{v_0} = \bot$, $level_{v_0} = mr$, $dist_{v_0} = 0$*, and* $v_0 \in B \cup \{r\}$*, (ii)* $\forall i \in \{1, \ldots, k\}, prnt_{v_i} = v_{i-1}$, $level_{v_i} = met(level_{v_{i-1}}, w_{v_i, v_{i-1}})$*, and* $dist_{v_i} = i$*, (iii)* $\forall i \in \{1, \ldots, k\}, met(level_{v_{i-1}}, w_{v_i, v_{i-1}}) = max_{\prec}\{met(level_u, w_{v_i,u}), u \in N_{v_i}\}$*, and (iv)* $level_{v_k} = \mu(v_k, v_0)$*.*

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

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

Following discussion of Section 3 and results from [8], 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 matches these optimal containment area. From now, S_B denotes this optimal containment area, *i.e.*:

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

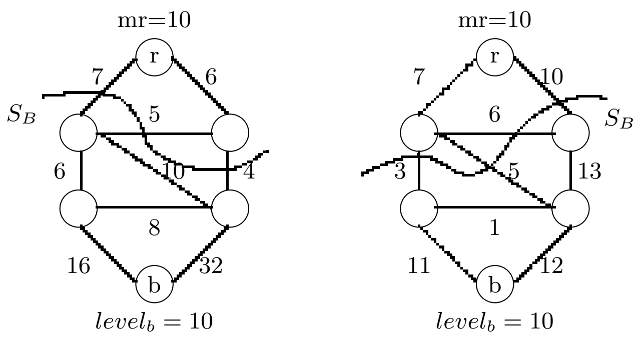

Fig. 1. Examples of containment areas for flow spanning tree construction

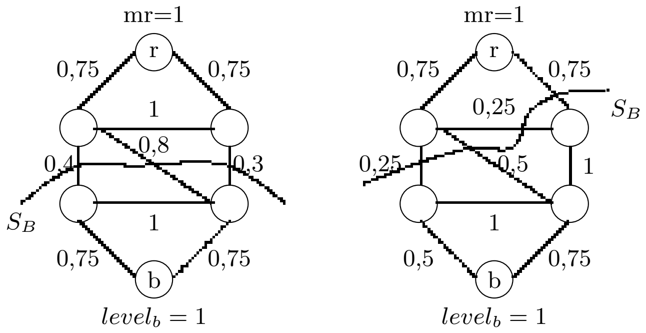

Fig. 2. Examples of containment areas for reliability spanning tree construction

Intuitively, Byzantine faults may disturb only processes that are (non strictly) closer from a Byzantine process than the root with respect to the metric. Figures 1 and 2 provide some examples of containment areas with respect to two 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 all value $w \in W$, we have: $met(m, w) = m$.*

4.1 Impossibility Result

In this section, we show that there exist some constraints on the containment area of any topology-aware strictly stabilizing protocol 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 $\mu(v, r) = min_{\prec}\{\mu(v, b), b \in B\} = m$. 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 $dist_u = 2$ (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 a self-stabilizing protocol), we can deduce that the system reaches in a finite time a configuration ρ_1^0 in which: $prnt_r = \bot$, $prnt_u = r$, $prnt_v = u$, $prnt_b = v$, $level_r = level_u = level_v = level_b = m$, $dist_r = 0$, $dist_u = 1$, $dist_v = 2$ 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 $dist_u = 2$ (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 a self-stabilizing protocol), we can deduce that the system reaches in a finite time a configuration ρ_1^1 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 $dist_u = 1$ (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 a self-stabilizing protocol), we can deduce that the system reaches in a finite time a configuration ρ_1^2 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 achieves 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 [15] (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 processes which have the same maximum metric. The main modifications we bring to this protocol follow. 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 ensure a fair selection along the set of its neighbor. We perform this fairness with a round-robin order along the set of neighbors. The second modification is to give priority to rules $(\boldsymbol{R_2})$ and $(\boldsymbol{R_3})$ over $(\boldsymbol{R_1})$ for any correct non root process (that is, such a process which has $(\boldsymbol{R_1})$ and another rule enabled in a given configuration always executes the other rule if it is activated). Our solution is presented as Algorithm 4.1.

In the following, we provide a sketch[1] of 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 $(V \setminus S_B) \setminus B$ is connected respectively by boundedness of the metric.

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

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

If we take a configuration $\rho \in \mathcal{C}$ such that $IM_m(\rho)$ holds for a given $m \in M$, then we can prove that the boundedness of $\mathcal{M}$ implies that, for any step $\rho \mapsto \rho'$ of $\mathcal{SSMAX}$, $IM_m(\rho')$ holds. Hence, we can deduce that:

Lemma 1. *For any metric value $m \in M$, the predicate IM_m is closed by actions of $\mathcal{SSMAX}$.*

[1] Due to the lack of place, formal proofs are omitted. A full version of this work is available in the companion technical report (see [17]).

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

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

$$\forall m_i \in M, P_{m_i} = \big\{v \in (V \setminus S_B) \setminus B \big| \mu(v, r) = m_i\big\}$$

$$\begin{aligned} \forall m_i \in M, \quad V_{m_i} &= \bigcup_{j=0}^{i} P_{m_j} \\ \forall m_i \in M, \quad I_{m_i} &= \big\{v \in V \big| max_{\prec}\{\mu(v,u), u \in B \cup \{r\}\} \prec m_i\big\} \\ \forall m_i \in M, \mathcal{LC}_{m_i} &= \big\{\rho \in \mathcal{C} \big| (\forall v \in V_{m_i}, spec(v)) \wedge (IM_{m_i}(\rho))\big\} \\ \mathcal{LC} &= \mathcal{LC}_{m_k} \end{aligned}$$

If we consider a configuration $\rho \in \mathcal{LC}_{m_i}$ for a given metric value m_i and a process $v \in V_{m_i}$, then we can show from the closure of IM_{m_i} (established in Lemma 1), the boundedness of $\mathcal{M}$ and the construction of the protocol that v is not enabled in ρ. Then, the closure of IM_{m_i} is sufficient to conclude that:

Lemma 2. *For any $m_i \in M$, the set $\mathcal{LC}_{m_i}$ is closed by actions of $\mathcal{SSMAX}$.*

Lemma 2 applied to $\mathcal{LC} = \mathcal{LC}_{m_k}$ gives us the following result:

Lemma 3. *Any configuration of $\mathcal{LC}$ is $(S_B, n-1)$-TA contained for spec.*

This lemma establishes the closure of $\mathcal{SSMAX}$. To prove the TA strict stabilization of $\mathcal{SSMAX}$, it remains to prove its convergence. In this goal, we prove that any execution starting from an arbitrary configuration of C converges to $\mathcal{LC}_{m_0} = \mathcal{LC}_{mr}$ and then to $\mathcal{LC}_{m_1}$ and so on until $\mathcal{LC}_{m_k} = \mathcal{LC}$.

Note that IM_{mr} is satisfied by any configuration of C and that if all processes of P_{mr} are not enabled in a configuration then this configuration belongs to $\mathcal{LC}_{mr}$. Then, we can prove that any process of P_{mr} takes only a finite number of steps in any execution. This implies the following result:

Lemma 4. *Starting from any configuration of $\mathcal{C}$, any execution of $\mathcal{SSMAX}$ reaches in a finite time a configuration of $\mathcal{LC}_{mr}$.*

Given a metric value $m_i \in M$ and a configuration $\rho_0 \in \mathcal{LC}_{m_i}$, assume that $e = \rho_0, \rho_1, \ldots$ is an execution of $\mathcal{SSMAX}$ starting from ρ_0. We define then 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 $f(\rho_j) = min\{dist_v, v \in A_j\}$. We can prove that there exists an integer k such that $f(\rho_k) = D$. This implies the following lemma:

Lemma 5. *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 $\forall v \in I_{m_i}, level_v = m_i \Rightarrow dist_v = D$.*

Given a metric value $m_i \in M$, consider a configuration $\rho_0 \in \mathcal{LC}_{m_i}$ such that $\forall v \in I_{m_i}, level_v = m_i \Rightarrow dist_v = D$. Assume that $e = \rho_0, \rho_1, \ldots$ is an execution of $\mathcal{SSMAX}$ starting from ρ_0. For any configuration ρ_i of e, we define the following set $E_{\rho_i} = \{v \in I_{m_i} | level_v = m_i\}$. First, we prove that there exists an integer k such that for any integer $j \geq k$, we have $E_{\rho_{j+1}} \subseteq E_{\rho_j}$. In other words, there exists a point of the execution afterwards the set E can not grow. Moreover, we prove that if a process of E_{ρ_j} $(j \geq k)$ is activated during the step $\rho_j \mapsto \rho_{j+1}$, then it satisfies $v \notin E_{\rho_{j+1}}$. Finally, we observe 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 $E_{\rho_j} = \emptyset$. In other words, we have:

Lemma 6. *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 $\forall v \in I_{m_i}, level_v \prec m_i$.*

A direct consequence of Lemmas 5 and 6 is the following.

Lemma 7. *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 $IM_{m_{i+1}}(\rho')$ holds.*

Given a metric value $m_i \in M$, consider a configuration $\rho \in \mathcal{LC}_{m_i}$. We know by Lemma 7 that any execution starting from ρ reaches in a finite time a configuration ρ' such that $IM_{m_{i+1}}(\rho')$ holds. Denote by e an execution starting from ρ'. Now, we can observe that, if all processes of $P_{m_{i+1}}$ are not enabled in a configuration of e, then this configuration belongs to $\mathcal{LC}_{m_{i+1}}$. Then, we can prove that any process of $P_{m_{i+1}}$ takes only a finite number of steps in any execution starting from ρ'. This implies the following 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 of $\mathcal{LC}_{m_{i+1}}$.*

Let ρ be an arbitrary configuration. We know by Lemma 4 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 8 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 following result:

Lemma 9. *Starting from any configuration, any execution of $\mathcal{SSMAX}$ reaches a configuration of $\mathcal{LC}$ in a finite time.*

Lemmas 3 and 9 imply respectively the closure and the convergence of $\mathcal{SSMAX}$. We can summarize our results with the following theorem.

Theorem 3. *$\mathcal{SSMAX}$ is a $(S_B, n-1)$-TA 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. Dijkstra, E.W.: Self-stabilizing systems in spite of distributed control. ACM Commun. 17(11), 643–644 (1974)
2. Dolev, S.: Self-stabilization. MIT Press, Cambridge (March 2000)
3. Tixeuil, S.: Self-stabilizing Algorithms. Chapman & Hall/CRC Applied Algorithms and Data Structures. In: Algorithms and Theory of Computation Handbook, 2nd edn., pp. 26.1–26.45. CRC Press, Taylor & Francis Group (November 2009)
4. Lamport, L., Shostak, R.E., Pease, M.C.: The byzantine generals problem. ACM Trans. Program. Lang. Syst. 4(3), 382–401 (1982)
5. Nesterenko, M., Tixeuil, S.: Discovering network topology in the presence of byzantine nodes. IEEE Trans. Parallel Distrib. Syst. (October 2009)

6. Dolev, S., Welch, J.L.: Self-stabilizing clock synchronization in the presence of byzantine faults. J. ACM 51(5), 780–799 (2004)
7. Daliot, A., Dolev, D.: Self-stabilization of byzantine protocols. In: Tixeuil, S., Herman, T. (eds.) SSS 2005. LNCS, vol. 3764, pp. 48–67. Springer, Heidelberg (2005)
8. Nesterenko, M., Arora, A.: Tolerance to unbounded byzantine faults. In: 21st Symposium on Reliable Distributed Systems, p. 22. IEEE Computer Society, Los Alamitos (2002)
9. Ben-Or, M., Dolev, D., Hoch, E.N.: Fast self-stabilizing byzantine tolerant digital clock synchronization. In: Bazzi, R.A., Patt-Shamir, B. (eds.) PODC, pp. 385–394. ACM, New York (2008)
10. Dolev, D., Hoch, E.N.: On self-stabilizing synchronous actions despite byzantine attacks. In: Pelc, A. (ed.) DISC 2007. LNCS, vol. 4731, pp. 193–207. Springer, Heidelberg (2007)
11. Hoch, E.N., Dolev, D., Daliot, A.: Self-stabilizing byzantine digital clock synchronization. In: Datta, A.K., Gradinariu, M. (eds.) SSS 2006. LNCS, vol. 4280, pp. 350–362. Springer, Heidelberg (2006)
12. Sakurai, Y., Ooshita, F., Masuzawa, T.: A self-stabilizing link-coloring protocol resilient to byzantine faults in tree networks. In: Higashino, T. (ed.) OPODIS 2004. LNCS, vol. 3544, pp. 283–298. Springer, Heidelberg (2005)
13. Masuzawa, T., Tixeuil, S.: 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 (2007)
14. Dubois, S., Potop-Butucaru, M.G., Nesterenko, M., Tixeuil, S.: Self-stabilizing byzantine asynchronous unison. CoRR abs/0912.0134 (2009)
15. Gouda, M.G., Schneider, M.: Stabilization of maximal metric trees. In: Arora, A. (ed.) WSS, pp. 10–17. IEEE Computer Society, Los Alamitos (1999)
16. Gouda, M.G., Schneider, M.: Maximizable routing metrics. IEEE/ACM Trans. Netw. 11(4), 663–675 (2003)
17. Dubois, S., Masuzawa, T., Tixeuil, S.: The Impact of Topology on Byzantine Containment in Stabilization. Research report inria-00481836 (May 2010), `http://hal.inria.fr/inria-00481836/en/`