Maximum Metric Spanning Tree Made Byzantine Tolerant*

Swan Dubois[1], Toshimitsu Masuzawa[2], and Sébastien Tixeuil[3]

[1] UPMC Sorbonne Universités & INRIA, France
[2] Osaka University, Japan
[3] UPMC Sorbonne Universités & Institut Universitaire de France, France

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. This paper focuses on systems that are both self-stabilizing and Byzantine tolerant.

We consider the well known problem of constructing a maximum metric tree in this context. Combining these two properties is known to induce many impossibility results. In this paper, we first provide two new impossibility results about the construction of a maximum metric tree in presence of transient and (permanent) Byzantine faults. Then, we propose a new self-stabilizing protocol that provides optimal containment to an arbitrary number of Byzantine faults.

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] 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 [5,6,7] as impossibility results are expected in many cases.

Related Works. A promising path towards multi-tolerance to both transient and Byzantine faults is *Byzantine containment*. For *local* tasks (*i.e.* tasks whose correctness can be checked locally, such as vertex coloring, link coloring, or dining

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

D. Peleg (Ed.): DISC 2011, LNCS 6950, pp. 150–164, 2011.

philosophers), the notion of *strict stabilization* was proposed [7,8]. 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. To circumvent impossibility results, the weaker notion of *strong stabilization* was proposed [9,10]: here, correct nodes outside the containment radius may be perturbed by the actions of Byzantine nodes, but only a finite number of times.

Recently, the idea of generalizing the containment area in strict and strong stabilization to an area that depends both on the graph topology and the problem to be solved rather than an arbitrary fixed containment radius was proposed [11,12] and denoted by *topology aware* strict (and strong) stabilization. When maximizable metric trees are considered, [11] proposed an optimal (with respect to impossibility results) protocol for topology-aware strict stabilization, and for the simpler case of breath-first-search trees, [12] presented a protocol that is optimal both with respect to strict and strong variants of topology-aware stabilization. Up to this paper, the case of optimality for topology-aware strong stabilization in the general maximal metric case remained open.

Our Contribution. In this paper, we investigate the possibility of topology-aware strong stabilization for tasks that are global (*i.e.* there exists a causality chain of size r, where r depends on n the size of the network), and focus on the maximum metric tree problem. Our contribution in this paper is threefold. First, we provide two impossibility results for self-stabilizing maximum metric tree construction in presence of Byzantine faults. In more details, we characterize a specific class of maximizable metrics (that includes breath-first-search and shortest path metrics) that prevents the existence of strong stabilizing solutions and we provide a lower bound on the containment area for topology-aware strong stabilization (Section 3). Second, we provide a topology-aware strongly stabilizing protocol that matches this lower bound on the containment area (Section 4). Finally, we provide a necessary and sufficient condition for the existence of a strongly stabilizing solution (Section 5).

2 Model, Definitions and Previous Results

2.1 State Model

A *distributed system* $S = (V, L)$ consists of a set $V = \{v_1, v_2, \ldots, v_n\}$ 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 V and whose link set is L, so we use graph terminology to describe a distributed system S. We use the following notations: $n = |V|$, $m = |L|$ and $d(u, v)$ denotes the distance between two processes u and v (*i.e* the length of the shortest path between u and v). 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. We assume each process can distinguish its neighbors from each other by locally labeling them. We denote the maximal degre of the system by Δ.

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 action 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 $R \subseteq V$ 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. 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'.

We say that a process is *enabled* in a configuration ρ if the guard of at least one of its actions is evaluated as true in 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 distinguish some properties on the scheduler (or daemon). A *distributed daemon* schedules the actions of processes such that any subset of processes can simultaneously execute their actions. We say that the daemon is *central* if it schedules action of only one process at any step. 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 but we add the following assumption on schedules: any schedule is strongly fair (that is, it is impossible for any process to be infinitely often enabled without executing its action infinitely often in an execution) and k-bounded (that is, it is impossible for any process to execute more than k actions between two consecutive actions of any other process).

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. For a given execution, the number of faulty processes is arbitrary but we assume that the root process is never faulty.

2.2 Self-Stabilizing Protocols 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 ($\subseteq V$) 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 $spec(v)$.

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, *i.e.* when 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 [7] define a *strictly stabilizing protocol* as a self-stabilizing protocol resilient to unbounded number of Byzantine processes. More formally, given an integer c, a process is c-correct if it is correct (*i.e.* not Byzantine) and located at distance more than c from any Byzantine process. 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 refers to the *containment radius* defined in [7]. The parameter f refers explicitly to the number of Byzantine processes, while [7] dealt with unbounded number of Byzantine faults (that is $f \in \{0 \dots n\}$).

Definition 1 ((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.*

Strong stabilization. To circumvent impossibility results related to strict stabilization, [10] defines a weaker notion. Here, the requirement to the containment radius is relaxed, *i.e.* there may exist processes outside the containment radius that invalidate the specification predicate, due to Byzantine actions. However, the impact of Byzantine triggered action is limited in times: the set of Byzantine processes may only impact processes outside the containment radius a bounded number of times, even if Byzantine processes execute an infinite number of actions.

More formally, [10] defines strong stabilization as follows. From the states of c-correct processes, *c-legitimate configurations* and *c-stable configurations* are defined as follows. A configuration ρ is c-legitimate for *spec* if every c-correct process v satisfies $spec(v)$. A configuration ρ is c-stable if every c-correct process never changes the values of its O-variables as long as Byzantine processes make no action. Roughly speaking, the aim of self-stabilization is to guarantee that a distributed system eventually reaches a c-legitimate and c-stable configuration. However, a self-stabilizing system can be disturbed by Byzantine processes after reaching a c-legitimate and c-stable configuration. The *c-disruption* represents the period where c-correct processes are disturbed by Byzantine processes and is defined as follows. A portion of execution $e = \rho_0, \rho_1, \ldots, \rho_t$ $(t > 1)$ is a c-disruption if and only if the following holds: (i) e is finite, (ii) e contains at least one action of a c-correct process for changing the value of an O-variable, (iii) ρ_0 is c-legitimate for *spec* and c-stable, and (iv) ρ_t is the first configuration after ρ_0 such that ρ_t is c-legitimate for *spec* and c-stable. A configuration ρ_0 is (t, k, c, f)-time contained for *spec* if given at most f Byzantine processes, the following properties are satisfied: (i) ρ_0 is c-legitimate for *spec* and c-stable, (ii) every execution starting from ρ_0 contains a c-legitimate configuration for *spec* after which the values of all the O-variables of c-correct processes remain unchanged (even when Byzantine processes make actions repeatedly and forever), (iii) every execution starting from ρ_0 contains at most t c-disruptions, and (iv) every execution starting from ρ_0 contains at most k actions of changing the values of O-variables for each c-correct process.

Definition 2 ((t, c, f)-strongly stabilizing protocol). *A protocol A is (t, c, f)-strongly stabilizing if and only if starting from any arbitrary configuration, every execution involving at most f Byzantine processes contains a (t, k, c, f)-time contained configuration. Parameter k is the (t, c, f)-process-disruption times of A.*

Topology-aware Byzantine resilience. We describe here another weaker notion than the strict stabilization: the *topology-aware strict stabilization* (denoted by TA strict stabilization for short) introduced by [11]. Here, the requirement to the containment radius is relaxed, *i.e.* the set of processes that may be disturbed by Byzantine ones is not reduced to the union of c-neighborhood of Byzantine processes (*i.e.* the set of processes at distance at most c from a Byzantine process) but can be defined depending on the graph topology and Byzantine processes' locations.

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 (that is function of B) denotes a subset of V (intuitively, this set gathers all processes that may be disturbed by Byzantine processes). A process is *S_B-correct* if it is a correct process (*i.e.* not Byzantine) that does not belongs to S_B. A configuration ρ is *S_B-legitimate* for *spec* if every S_B-correct process v is legitimate for *spec* (*i.e.* if $spec(v)$ holds). A configuration ρ_0 is *(S_B, f)-TA 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 refers to the *containment area*. Any process that belongs to this set may be infinitely disturbed by Byzantine processes. The parameter f refers explicitly to the number of Byzantine processes.

Definition 3 ((S_B, f)-TA strict stabilization). *A protocol is* (S_B, f)-TA 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)*-TA contained for spec.*

Similarly to topology-aware strict stabilization, we can weaken the notion of strong stabilization using the notion of containment area. This idea was introduced by [12]. We recall in the following the formal definition of this concept. A configuration ρ is S_B-stable if every S_B-correct process never changes the values of its O-variables as long as Byzantine processes make no action. A portion of execution $e = \rho_0, \rho_1, \ldots, \rho_t$ $(t > 1)$ is a S_B-TA disruption if and only if the followings hold: (i) e is finite, (ii) e contains at least one action of a S_B-correct process for changing the value of an O-variable, (iii) ρ_0 is S_B-legitimate for *spec* and S_B-stable, and (iv) ρ_t is the first configuration after ρ_0 such that ρ_t is S_B-legitimate for *spec* and S_B-stable. A configuration ρ_0 is $(t, k, S_B,\ f)$-TA time contained for *spec* if given at most f Byzantine processes, the following properties are satisfied: (i) ρ_0 is S_B-legitimate for *spec* and S_B-stable, (ii) every execution starting from ρ_0 contains a S_B-legitimate configuration for *spec* after which the values of all the O-variables of S_B-correct processes remain unchanged (even when Byzantine processes make actions repeatedly and forever), (iii) every execution starting from ρ_0 contains at most t S_B-TA disruptions, and (iv) every execution starting from ρ_0 contains at most k actions of changing the values of O-variables for each S_B-correct process.

Definition 4 ((t, S_B, f)-TA strongly stabilizing protocol). *A protocol A is* (t, S_B, f)*-TA strongly stabilizing if and only if starting from any arbitrary configuration, every execution involving at most f Byzantine processes contains a* (t, k, S_B, f)*-TA time contained configuration that is reached after at most l rounds. Parameters l and k are respectively the* (t, S_B, f)*-stabilization time and the* (t, S_B, f)*-process disruption times of A.*

2.3 Maximum Metric Tree Construction

In this work, we deal with maximum (routing) metric trees. 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 $\prec$. We recall all definitions and notations introduced in [13].

A *routing metric* (or just *metric*) is a five-tuple $(M, W, met, mr, \prec)$ where: (i) M is a set of metric values, (ii) W is a set of edge weights, (iii) met is a metric function whose domain is $M \times W$ and whose range is M, (iv) mr is the maximum metric value in M with respect to $\prec$ and is assigned to the root of the system, and (v) $\prec$ is a less-than total order relation over M. The $\prec$ relation must satisfy the following three conditions for arbitrary metric values

m, m', and m'' in M: (i) irreflexivity ($m \not\prec m$), (ii) transitivity (if $m \prec m'$ and $m' \prec m''$ then $m \prec m''$), and (iii) totality ($m \prec m'$ or $m' \prec m$ or $m = m'$). Any metric value $m \in M \setminus \{mr\}$ must satisfy 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, this model allows to modelize the following metrics: the shortest path metric ($\mathcal{SP}$), the flow metric ($\mathcal{F}$), and the reliability metric ($\mathcal{R}$) as pointed in [13]. Note also that we can modelize the construction of a spanning tree with no particular constraints or a BFS spanning tree using a routing metric (respectively denoted by $\mathcal{NC}$ and by $\mathcal{BFS}$).

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. 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 $\forall i \in \{1, \ldots, k-1\}, m_i = met(m_{i-1}, wf(\{v_i, v_{i-1}\}))$ and $m_0 = mr$. 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 $\prec$. 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. 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 [13] is the study of metrics that always allow the construction of a maximum metric tree. More formally, the definition follows. 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.

Given a maximizable metric $\mathcal{M} = (M, W, met, mr, \prec)$, the aim of this work is to study the construction of a maximum metric tree with respect to $\mathcal{M}$ that spans the system in a self-stabilizing way in a system subject to permanent Byzantine faults (but we assume that the root process is never a Byzantine one). 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 that stores its current metric value ($level_v \in M$) and an integer that stores a distance ($dist_v \in \mathbb{N}$). 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 neighbors, we denote by $w_{u,v}$ the weight of the edge $\{u, v\}$ (that is, the value of $wf(\{u, v\})$).

Definition 5 ($\mathcal{M}$-path). *Given an assigned metric $\mathcal{M} = (M, W, met, mr, \prec, wf)$ 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: (i) $prnt_{v_0} = \perp$, $level_{v_0} = mr$, $dist_{v_0} = 0$, and $v_0 \in B \cup \{r\}$, (ii) $\forall i \in \{1, \ldots, k\}, prnt_{v_i} = v_{i-1}$ and $level_{v_i} = met(level_{v_{i-1}}, w_{v_i,v_{i-1}})$, (iii) $\forall i \in \{1, \ldots, k\}, met(level_{v_{i-1}}, w_{v_i,v_{i-1}}) = max_{\prec}\{met(level_u, w_{v_i,u}) | u \in N_{v_i}\}$, (iv) $\forall i \in \{1, \ldots, k\}, dist_{v_i} = legal\_dist_{v_{i-1}}$ (with $\forall u \in N_v, legal\_dist_u = dist_u + 1$ if $level_v = level_u$ and $legal\_dist_u = 0$ otherwise), and (v) $level_{v_k} = \mu(v_k, v_0)$.*

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

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

2.4 Previous Results

The first interesting result is due to [13] that provides a full characterization of maximizable metrics as follow. 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 $met(m, w) = m$. 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 $met(m, w) = met(m', w))$. Then, [13] proves that a metric is maximizable if and only if this metric is bounded and monotonic. Secondly, [14] provides a self-stabilizing protocol to construct a maximum metric tree with respect to any maximizable metric.

Now, we focus on self-stabilizing solutions resilient to Byzantine faults. Following [7], it is obvious that there exists no strictly stabilizing protocol for this problem. If we consider the weaker notion of topology-aware strict stabilization, [11] defines the best containment area as: $S_B = \{v \in V \setminus B | \mu(v, r) \preceq max_{\prec} \{\mu(v, b) | b \in B\}\} \setminus \{r\}$. Intuitively, S_B gathers correct processes that are closer (or at equal distance) from a Byzantine process than the root according to the metric. Moreover, [11] proves that the algorithm introduced for the maximum metric spanning tree construction in [14] achieves this optimal containment area. In other words, [11] proves the following results: (i) given a maximizable metric $\mathcal{M} = (M, W, met, mr, \prec)$, 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 $A_B \subsetneq S_B$ and (ii) given a maximizable metric $\mathcal{M} = (M, W, met, mr, \prec)$, the protocol of [14] is a $(S_B, n-1)$-TA strictly stabilizing protocol for maximum metric spanning tree construction with respect to $\mathcal{M}$.

Some other works try to circumvent the impossibility result of strict stabilization using the concept of strong stabilization but do not provide results for

all maximizable metric. Indeed, [10] proves the following result about spanning trees: there exists a $(t, 0, n-1)$-strongly stabilizing protocol for maximum metric spanning tree construction with respect to $\mathcal{NC}$ (that is, for a spanning tree with no particular constraints) with a finite t. On the other hand, regarding BFS spanning tree construction, [12] proved the following impossibility result: even under the central daemon, there exists no $(t, c, 1)$-strongly stabilizing protocol for maximum metric spanning tree construction with respect to $\mathcal{BFS}$ where t and c are two finite integers.

Now, if we focus on topology-aware strong stabilization, [12] introduced the following containment area: $S_B^* = \{v \in V | min\{d(v,b) | b \in B\} < d(r,v)\}$. We proved then the following results: (i) even under the central daemon, there exists no $(t, A_B^*, 1)$-TA strongly stabilizing protocol for maximum metric spanning tree construction with respect to $\mathcal{BFS}$ where $A_B^* \subsetneq S_B^*$ and t is a finite integer and (ii) the protocol of [15] is a $(t, S_B^*, n-1)$-TA strongly stabilizing protocol for maximum metric spanning tree construction with respect to $\mathcal{BFS}$ where t is a finite integer.

The main motivation of this work is to fill the gap between results about TA strong and strong stabilization in the general case (that is, for any maximizable metric). Mainly, we define the best possible containment area for TA strong stabilization, we propose a protocol that provides this containment area and we characterize the set of metrics that allow strong stabilization.

3 Impossibility Results

We provide here our impossibility results about containment radius (resp. area) of any strongly stabilizing (resp. TA strongly stabilizing) protocol for the maximum metric tree construction.

Strong Stabilization. We introduce here new definitions to characterize some important properties of maximizable metrics that are used in the following. A metric $\mathcal{M} = (M, W, met, mr, \prec)$ is *strictly decreasing* if, for any metric value $m \in M$, the following property holds: either $\forall w \in W, met(m,w) \prec m$ or $\forall w \in W, met(m,w) = m$. A metric value m is a *fixed point* of a metric $\mathcal{M} = (M, W, met, mr, \prec)$ if $m \in M$ and if for any value $w \in W$, we have: $met(m,w) = m$. Then, we define a specific class of maximizable metrics and we prove that it is impossible to construct a maximum metric tree in a strongly-stabilizing way if the considered metric is not in the class.

Definition 6 (Strongly maximizable metric). *A maximizable metric $\mathcal{M} = (M, W, met, mr, \prec)$ is strongly maximizable if and only if $|M| = 1$ or if the following properties hold: (i) $|M| \geq 2$, (ii) $\mathcal{M}$ is strictly decreasing, and (iii) $\mathcal{M}$ has one and only one fixed point.*

Note that $\mathcal{NC}$ is a strongly maximizable metric (since $|M| = 1$) whereas $\mathcal{BFS}$ or $\mathcal{F}$ are not (since the first one has no fixed point, the second is not strictly decreasing). Now, we can state our first impossibility result (the proof is available in [16]).

Theorem 1. *Given a maximizable metric $\mathcal{M} = (M, W, met, mr, \prec)$, even under the central daemon, there exists no $(t, c, 1)$-strongly stabilizing protocol for maximum metric spanning tree construction with respect to $\mathcal{M}$ for any finite integer t if: (i) $\mathcal{M}$ is not a strongly maximizable metric, or (ii) $c < |M| - 2$.*

Topology Aware Strong Stabilization. First, we generalize the set S_B^* previously defined for the $\mathcal{BFS}$ metric in [12] to any maximizable metric $\mathcal{M} = (M, W, met, mr, \prec)$. From now, S_B^* denotes the following set:

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

Intuitively, S_B^* gathers the set of correct processes that are strictly closer (according to $\mathcal{M}$) to a Byzantine process than the root. Note that we assume for the sake of clarity that $V \setminus S_B^*$ induces a connected subsystem. If it is not the case, then S_B^* is extended to include all processes belonging to connected subsystems of $V \setminus S_B^*$ that do not contain r. Now, we can state our generalization of impossibility result of [12] (the proof is available in [16]).

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

4 Topology-Aware Strongly Stabilizing Protocol

The goal of this section is to provide a $(t, S_B^*, n-1)$-TA strongly stabilizing protocol in order to match the lower bound on containment area provided by Theorem 2. If we focus on the protocol provided by [11] (that is $(S_B, n-1)$-TA strictly stabilizing), we can prove that this protocol does not satisfy our constraints since it is not $(t, S_B^*, 2)$-TA strongly stabilizing (see [16]).

4.1 Presentation of the Protocol

Our protocol needs a supplementary assumption. We introduce the following definition.

Definition 7 (Set of used metric values). *Given an assigned metric $\mathcal{AM} = (M, W, met, mr, \prec, wf)$ over a system S, the* set of used metric values *of $\mathcal{AM}$ is defined as $M(S) = \{m \in M | \exists v \in V, (\mu(v, r) = m) \vee (\exists b \in B, \mu(v, b) = m)\}$.*

We assume that we always have $|M(S)| \geq 2$ (the necessity of this assumption is explained below). Nevertheless, note that the contrary case ($|M(S)| = 1$) is possible if and only if the assigned metric is equivalent to $\mathcal{NC}$. As the protocol of [10] performs $(t, 0, n-1)$- strong stabilization with a finite t for this metric, we can achieve the $(t, S_B^*, n-1)$-TA strong stabilization when $|M(S)| = 1$ (since this implies that $S_B^* = \emptyset$). In this way, this assumption does not weaken the possibility result.

Although the protocol of [11] is not $(t, S_B^*, n-1)$-TA strongly stabilizing, our protocol borrows fundamental strategy from it. In this protocol, any process tries to maximize its *level* in the tree by choosing as its parent the neighbor that provides the best metric value. The key idea of this protocol is to use the *dist* variable (upper bounded by a given constant D) to detect and break cycles of processes that have the same maximum metric. To achieve $(S_B, n-1)$-TA strict stabilization, the protocol ensures a fair selection along the set of its neighbors with a round-robin order.

The possibility of infinite number of S_B^*-TA disruptions of the protocol of [11] mainly comes from the following fact: a Byzantine process can independently lie about its *level* and its *dist* variables. For example, a Byzantine process can provide a *level* equal to *mr* and a *dist* arbitrarily large. In this way, it may lead a correct process of $S_B \setminus S_B^*$ to have a *dist* variable equal to $D-1$ such that no other correct process can choose it as its parent (this rule is necessary to break cycle) but it cannot modify its state (this rule is only enabled when *dist* is equal to D). Then, this process may always prevent some of its neighbors to join a $\mathcal{M}$-path connected to the root and hence allow another Byzantine process to perform an infinite number of S_B^*-TA disruptions.

It is why we modified the management of the *dist* variable (note that other variables are managed exactly in the same way as in the protocol of [11]). In order to contain the effect of Byzantine processes on *dist* variables, each process that has a *level* different from the one of its parent in the tree sets its *dist* variable to 0. In this way, a Byzantine process modifying its *dist* variable can only affect correct processes that have the same *level*. Consequently, in the case where $|M(S)| \geq 2$, we are ensured that correct processes of $S_B \setminus S_B^*$ cannot keep a *dist* variable equal or greater than $D-1$ infinitely. Hence, a correct process of $S_B \setminus S_B^*$ cannot be disturbed infinitely often without joining a $\mathcal{M}$-path connected to the root.

We can see that the assumption $|M(S)| \geq 2$ is essential to perform the TA strong stabilization. Indeed, in the case where $|M(S)| = 1$, Byzantine processes can play exactly the scenario described above (in this case, our protocol is equivalent to the one of [11]).

The second modification we bring to the protocol of [11] follows. When a process has an inconsistent *dist* variable with its parent, we allow it only to increase its *dist* variable. If the process needs to decrease its *dist* variable (when it has a strictly greater distance than its parent), then it must change its parent. This rule allows us to bound the maximal number of steps of any process between two modifications of its parent (a Byzantine process cannot lead a correct one to infinitely often increase and decrease its distance without modifying its parent).

Our protocol is formally described in Algorithm 4.1.

4.2 Proof of the $(t, S_B^*, n-1)$-TA Strong Stabilization for *spec*

Due to the lack of space, only key ideas of this proof are provided but complete proofs are available in [16]. First, we prove the following result with a proof very similar to the one of [11].

Algorithm 4.1. $\mathcal{SSMAX}$, TA strongly 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:
$prnt_v \in \begin{cases} \{\perp\} \text{ if } v = r \\ N_v \text{ if } v \neq r \end{cases}$: pointer on the parent of v in the tree.
$level_v \in M$: metric of the node.
$dist_v \in \{0, \ldots, D\}$: hop counter.
Functions:
For any subset $A \subseteq N_v$, $choose_v(A)$ returns the first element of A that is bigger than $prnt_v$ (in a round-robin fashion).
$current\_dist_v() = \begin{cases} 0 \text{ if } level_{prnt_v} \neq level_v \\ min\{dist_{prnt_v} + 1, D\} \text{ if } level_{prnt_v} = level_v \end{cases}$
Rules:
$(\boldsymbol{R_r})$ :: $(v = r) \wedge ((level_v \neq mr) \vee (dist_v \neq 0)) \longrightarrow level_v := mr;\ dist_v := 0$
$(\boldsymbol{R_1})$ :: $(v \neq r) \wedge (prnt_v \in N_v) \wedge ((dist_v < current\_dist_v()) \vee (level_v \neq met(level_{prnt_v}, w_{v,prnt_v})))$
$\longrightarrow level_v := met(level_{prnt_v}, w_{v,prnt_v});\ dist_v := current\_dist_v()$
$(\boldsymbol{R_2})$ :: $(v \neq r) \wedge ((dist_v = D) \vee (dist_v > current\_dist_v())) \wedge (\exists u \in N_v, dist_u < D-1)$
$\longrightarrow prnt_v := choose_v(\{u \in N_v | dist_u < D-1\});\ level_v := met(level_{prnt_v}, w_{v,prnt_v});$
$dist_v := current\_dist_v()$
$(\boldsymbol{R_3})$ :: $(v \neq r) \wedge (\exists u \in N_v, (dist_u < D-1) \wedge (level_v \prec met(level_u, w_{u,v})))$
$\longrightarrow prnt_v := choose_v(\{u \in N_v | (level_u < D-1) \wedge$
$(met(level_u, w_{u,v}) = max_\prec\{met(level_q, w_{q,v}) | q \in N_v, level_q < D-1\})\});$
$level_v := met(level_{prnt_v}, w_{prnt_v,v});\ dist_v := current\_dist_v()$

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

We introduce here some notations. Given a configuration $\rho \in C$ and a metric value $m \in M$, we define the following predicate: $IM_m(\rho) \equiv \forall v \in V, level_v \preceq max_\prec\{m, max_\prec\{\mu(v,u) | u \in B \cup \{r\}\}\}$. Given an assigned metric to a system S, we can observe that the set of metric values $M(S)$ is finite and that we can label elements of $M(S)$ by $m_0 = mr, m_1, \ldots, m_k$ in a way such that $\forall i \in \{0, \ldots, k-1\}, m_{i+1} \prec m_i$. Let $\mathcal{LC}$ be the following set of configurations:

$$\mathcal{LC} = \left\{\rho \in \mathcal{C} \middle| (\forall v \in V \setminus S_B, spec(v)) \wedge (IM_{m_k}(\rho))\right\}$$

Let $E_B = S_B \setminus S_B^*$ (*i.e.* E_B is the set of process v such that $\mu(v,r) = max\{\mu(v,b) | b \in B\}$). Note that the subsystem induced by E_B may have several connected components. In the following, we use the following notations: $E_B = \{E_B^1, \ldots, E_B^\ell\}$ where each E_B^i ($i \in \{0, \ldots, \ell\}$) is a subset of E_B inducing a maximal connected component, $\delta(E_B^i)$ ($i \in \{0, \ldots, \ell\}$) is the diameter of the subsystem induced by E_B^i, and $\delta = max\{\delta(E_B^i) | i \in \{0, \ldots, \ell\}\}$. When a and b are two integers, we define the following function: $\Pi(a,b) = \frac{a^{b+1}-1}{a-1}$.

Let ρ be a configuration of $\mathcal{LC}$ and e be an execution starting from ρ. Let p be a process of E_B^i ($i \in \{0, \ldots, \ell\}$) such that there exists a neighbor q that satisfies $q \in V \setminus S_B$ and $\mu(p,r) = met(\mu(q,r), w_{p,q})$ (such a process exists by construction of E_B^i). Then, we can prove by induction the following property: if v is a process of E_B^i such that $d_{E_B^i}(p,v) = d$ (where $d_{E_B^i}$ denotes the distance in the subsystem induced by E_B^i), then v executes at most $\Pi(k,d)\Delta D$ actions in e. We can deduce the following lemma:

Lemma 1. *If ρ is a configuration of $\mathcal{LC}$, then any process $v \in E_B$ is activated at most $\Pi(k,\delta)\Delta D$ times in any execution starting from ρ.*

If we assume that there exists an execution starting from a configuration of $\mathcal{LC}$ such that a process $v \in E_B$ is never enabled and $spec(v)$ is infinitely often false, we can find a contradiction with the construction of the algorithm. Hence the following lemma holds:

Lemma 2. *If ρ is a configuration of $\mathcal{LC}$ and v is a process such that $v \in E_B$, then for any execution e starting from ρ either (i) there exists a configuration ρ' of e such that $spec(v)$ is always satisfied after ρ', or (ii) v is activated in e.*

Let $\mathcal{LC}^*$ be the following set of configurations:

$$\mathcal{LC}^* = \{\rho \in C | (\rho \text{ is } S_B^*\text{-legitimate for } spec) \wedge (IM_{m_k}(\rho) = true)\}$$

Note that, as $S_B^* \subseteq S_B$, we can deduce that $\mathcal{LC}^* \subseteq \mathcal{LC}$. Hence, properties of Lemmas 1 and 2 also apply to configurations of $\mathcal{LC}^*$ and we can deduce the following lemma:

Lemma 3. *Configurations of $\mathcal{LC}^*$ are $(n\Pi(k,\delta)\Delta D, \Pi(k,\delta)\Delta D, S_B^*, n-1)$-TA time contained for spec and any execution of $\mathcal{SSMAX}$ (starting from any configuration) contains such a configuration.*

Finally, we can deduce the main theorem:

Theorem 4. *$\mathcal{SSMAX}$ is a $(n\Pi(k,\delta)\Delta D, S_B^*, n-1)$-TA strongly stabilizing protocol for spec.*

5 Concluding Remarks

We now discuss the relationship between TA strong and strong stabilization for maximum metric tree construction. As a matter of fact, the set of assigned metrics that allow strong stabilization can be characterized. Indeed, properties about the metric itself are not sufficient to decide the possibility of strong stabilization: it is necessary to include some knowledge about the considered system (typically, the metric assignement). Informally, it is possible to construct a maximum metric tree in a strongly stabilizing way if and only if the considered metric is strongly maximizable and if the desired containment radius is sufficiently large with respect to the size of the set of used metric values. The formal statement of this result follows.

Theorem 5. *Given an assigned metric $\mathcal{AM} = (M, W, mr, met, \prec, wf)$ over a system S, there exists a $(t, c, n-1)$-strongly stabilizing protocol for maximum metric spanning tree construction with a finite t if and only if (i) $(M, W, met, mr, \prec)$ is a strongly maximizable metric, and (ii) $c \geq max\{0, |M(S)| - 2\}$.*

Table 1. Summary of results

	$\mathcal{M} = (M, W, mr, met, \prec)$ is a maximizable metric
(c, f)-strict stabilization	Impossible (for any c and f) by [7]
(t, c, f)-strong stabilization	Possible $\iff \begin{cases} \mathcal{M} \text{ is a strongly maximizable metric, and} \\ c \geq max\{0, \lvert M(S)\rvert - 2\} \end{cases}$ (for $0 \leq f \leq n-1$ and a finite t) by Theorem 5
(A_B, f)-TA strict stabilization	Impossible (for any f and $A_B \subsetneq S_B$) by [11]
(S_B, f)-TA strict stabilization	Possible (for $0 \leq f \leq n-1$) by [11] and Theorem 3
(t, A_B, f)-TA strong stabilization	Impossible (for any f and $A_B \subsetneq S_B^*$) by Theorem 2
(t, S_B^*, f)-TA strong stabilization	Possible (for $0 \leq f \leq n-1$ and a finite t) by Theorem 4

The "only if" part of this Theorem is a direct consequence of Theorem 1 when we observe that $|M(S)| \leq |M|$ by definition. The "if" part of this Theorem comes from the following observations. If $|M(S)| = 1$, it is equivalent to construct a maximum metric spanning tree for $\mathcal{M}$ and for $\mathcal{NC}$ over this system. By [10], we know that there exists a $(t, 0, n-1)$-strongly stabilizing protocol for this problem with a finite t, that proves the result. If $|M(S)| \geq 2$, we can prove that $S_B^* = \{v \in V | min\{d(v, b) | b \in B\} \leq c\}$ and then we have the result by Theorem 4.

We can now summarize all results about self-stabilizing maximum metric tree construction in presence of Byzantine faults with Table 1. Note that results provided in this paper close every open question raised in related works and that they subsume results from [10] and [12].

Our choice was to work with a specification of the problem that considers the *dist* variable as an O-variable. This choice may appear a bit strong but it permitted us to maintain consistency in the presentation of the results. In fact, impossibility results of Section 3 can be proved with a weaker specification that does not consider the *dist* variable as an O-variable (see [17]). On the other hand, the stronger specification eased the bounding of the number of disruptions of the proposed protocol. Our protocol is also TA strongly stabilizing with the weaker specification but we were not able to exactly bound the number of disruptions.

The following questions are still open. Is it possible to bound the number of disruptions for the weaker specification? Is it possible to perform TA strong stabilization using a weaker daemon requirement? Is it possible to decrease the number of disruptions without loosing containment optimality?

References

1. Dijkstra, E.W.: Self-stabilizing systems in spite of distributed control. Commun. ACM 17(11), 643–644 (1974)
2. Dolev, S.: Self-stabilization. MIT Press, Cambridge (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. Dolev, S., Welch, J.L.: Self-stabilizing clock synchronization in the presence of byzantine faults. J. ACM 51(5), 780–799 (2004)

6. Daliot, A., Dolev, D.: Self-stabilization of byzantine protocols. In: Herman, T., Tixeuil, S. (eds.) SSS 2005. LNCS, vol. 3764, pp. 48–67. Springer, Heidelberg (2005)
7. Nesterenko, M., Arora, A.: Tolerance to unbounded byzantine faults. In: 21st Symposium on Reliable Distributed Systems (SRDS 2002), p. 22. IEEE Computer Society, Los Alamitos (2002)
8. 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)
9. Masuzawa, T., Tixeuil, S.: Bounding the impact of unbounded attacks in stabilization. In: Datta, A.K., Gradinariu, M. (eds.) SSS 2006. LNCS, vol. 4280, pp. 440–453. Springer, Heidelberg (2006)
10. Dubois, S., Masuzawa, T., Tixeuil, S.: Bounding the impact of unbounded attacks in stabilization. IEEE Transactions on Parallel and Distributed Systems, TPDS (2011)
11. Dubois, S., Masuzawa, T., Tixeuil, S.: The impact of topology on byzantine containment in stabilization. In: Lynch, N.A., Shvartsman, A.A. (eds.) DISC 2010. LNCS, vol. 6343, pp. 495–509. Springer, Heidelberg (2010)
12. Dubois, S., Masuzawa, T., Tixeuil, S.: On byzantine containment properties of the min+1 protocol. In: Dolev, S., Cobb, J., Fischer, M., Yung, M. (eds.) SSS 2010. LNCS, vol. 6366, pp. 96–110. Springer, Heidelberg (2010)
13. Gouda, M.G., Schneider, M.: Maximizable routing metrics. IEEE/ACM Trans. Netw. 11(4), 663–675 (2003)
14. Gouda, M.G., Schneider, M.: Stabilization of maximal metric trees. In: Arora, A. (ed.) WSS, pp. 10–17. IEEE Computer Society, Los Alamitos (1999)
15. Huang, S.T., Chen, N.S.: A self-stabilizing algorithm for constructing breadth-first trees. Inf. Process. Lett. 41(2), 109–117 (1992)
16. Dubois, S., Masuzawa, T., Tixeuil, S.: Maximum Metric Spanning Tree made Byzantine Tolerant. Research report (2011), `http://hal.inria.fr/inria-00589234/en/`
17. Dubois, S., Masuzawa, T., Tixeuil, S.: Self-Stabilization, Byzantine Containment, and Maximizable Metrics: Necessary Conditions. Research report (2011), `http://hal.inria.fr/inria-00577062/en`