Self-Stabilization, Byzantine Containment, and Maximizable Metrics: Necessary Conditions

Swan Dubois* Toshimitsu Masuzawa† Sébastien Tixeuil‡

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 is known to induce many impossibility results. In this paper, we provide two necessary conditions to construct maximum metric tree in presence of transients and (permanent) Byzantine faults.

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* [2, 3, 15] is a versatile technique that permits forward recovery from any kind of *transient* faults, while *Byzantine Fault-tolerance* [11] 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 [4, 1, 14] as impossibility results are expected in many cases.

Related Works A promizing path towards multitolerance 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 philosophers), the notion of *strict stabilization* was proposed [14, 13]. 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 [12, 7]: here, correct nodes outside the containment radius may be perturbated by the actions of Byzantine node, but only a finite number of times.

Recently, the idea of generalizing strict and strong stabilization to an area that depends on the graph topology and the problem to be solved rather than an arbitrary fixed containment radius was proposed [5, 6] and denoted by *topology aware* strict (and strong) stabilization. When maximizable metric trees are considered, [5] proposed an optimal (with respect to impossibility results) protocol for topology-aware strict stabilization, and for the simpler case of breath-first-search metric trees,

*UPMC Sorbonne Universités & INRIA, France, swan.dubois@lip6.fr
†Osaka University, Japan, masuzawa@ist.osaka-u.ac.jp
‡UPMC Sorbonne Universités & Institut Universitaire de France, France, sebastien.tixeuil@lip6.fr

[6] presented a protocol that is optimal both with respect to strict and strong variants of topology-aware stabilization. The case of optimality for topology-aware strong stabilization in the general maximal metric case remains open.

Our Contribution In this paper, we investigate the possibility of topology-aware strong stabilization for tasks that are global (*i.e.* for with there exists a causality chain of size r, where r depends on n the size of the network), and focus on the maximum metric tree problem. In more details, we provide two necessary conditions to perform Byzantine containment for maximum metric tree construction. First, we characterize a specific class of maximizable metrics (which includes breath-first-search and shortest path metrics) that prevents the exitence of strong stabilizing solutions. Then, we generalize an impossibilty result of [6] that provides a lower bound on the containmemt area for topology-aware strong stabilization.

2 Model and Definitions

2.1 State Model

A *distributed system* $S = (P, L)$ consists of a set $P = \{v_1, v_2, \ldots, v_n\}$ of processes and a set L of bidirectional communication links (simply called links). A link is an unordered pair of distinct processes. A distributed system S can be regarded as a graph whose vertex set is P and whose link set is L, so we use graph terminology to describe a distributed system S. We use the following notations: $n = |P|$, $m = |L|$ and $d(u, v)$ denotes the shortest path 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 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 labelling them.

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 $R \subseteq P$ and two configurations ρ and ρ', we denote $\rho \overset{R}{\mapsto} \rho'$ when ρ changes to ρ' by executing an action of each process in R simultaneously. Notice that ρ and ρ' can be different only in the states of processes in R. For completeness of execution semantics, we should clarify the configuration resulting from simultaneous actions of neighboring processes. The action of a process depends only on its state at ρ and the states of its neighbors at ρ, and the result of the action reflects on the state of the process at ρ'.

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 $R^i \subseteq P$ holds for each i ($i \geq 1$). An infinite sequence of configurations

$e = \rho_0, \rho_1, \ldots$ is called an *execution* from an initial configuration ρ_0 by a schedule Q, if e satisfies $\rho_{i-1} \overset{R^i}{\mapsto} \rho_i$ for each i ($i \geq 1$). Process actions are executed atomically, and we 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 where we can make no assumption on schedules.

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 ($\subseteq P$) where P_v is the set of processes whose variables appear in $spec(v)$. The variables appearing in the specification are called *output variables* (shortly, *O-variables*). In what follows, we consider a static problem defined by specification $spec(v)$.

A *self-stabilizing protocol* ([2]) 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.

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 [14] 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 [14]. The parameter f refers explicitly to the number of Byzantine processes, while [14] 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 [14] is the notion of *r-restrictive* specifications. Intuitively, a specification is r-restrictive if it prevents combinations of states that belong to two processes u and v that are at least r hops away. An important consequence related to Byzantine tolerance is that the containment radius of protocols solving those specifications is at least r. For some (global) problems r can not be bounded by a constant. In consequence, we can show that there exists no $(c, 1)$-strictly stabilizing protocol for such a problem for any (finite) integer c.

Strong stabilization To circumvent such impossibility results, [7] defines a weaker notion than the strict stabilization. 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.

In the following of this section, we recall the formal definition of strong stabilization adopted in [7]. From the states of c-correct processes, *c-legitimate configurations* and *c-stable configurations* are defined as follows.

Definition 4 (c-legitimate configuration) *A configuration ρ is c-legitimate for* spec *if every c-correct process v satisfies spec(v).*

Definition 5 (c-stable configuration) *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

Definition 6 (c-disruption) *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:*

1. *e is finite,*
2. *e contains at least one action of a c-correct process for changing the value of an O-variable,*
3. *ρ_0 is c-legitimate for* spec *and c-stable, and*
4. *ρ_t is the first configuration after ρ_0 such that ρ_t is c-legitimate for* spec *and c-stable.*

Now we can define a self-stabilizing protocol such that 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.

Definition 7 ((t, k, c, f)-time contained configuration) *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:*

1. *ρ_0 is c-legitimate for* spec *and c-stable,*

2. *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),*

3. *every execution starting from ρ_0 contains at most t c-disruptions, and*

4. *every execution starting from ρ_0 contains at most k actions of changing the values of O-variables for each c-correct process.*

Definition 8 ((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 that is reached after at most l rounds. Parameters l and k are respectively the (t, c, f)-stabilization time and the (t, c, f)-process-disruption times of A.*

Note that a (t, k, c, f)-time contained configuration is a (c, f)-contained configuration when $t = k = 0$, and thus, (t, k, c, f)-time contained configuration is a generalization (relaxation) of a (c, f)-contained configuration. Thus, a strongly stabilizing protocol is weaker than a strictly stabilizing one (as processes outside the containment radius may take incorrect actions due to Byzantine influence). However, a strongly stabilizing protocol is stronger than a classical self-stabilizing one (that may never meet their specification in the presence of Byzantine processes).

The parameters t, k and c are introduced to quantify the strength of fault containment, we do not require each process to know the values of the parameters.

Topology-aware Byzantine resilience We saw previously that there exist a number of impossibility results on strict stabilization due to the notion of r-restrictives specifications. To circumvent this impossibility result, we describe here another weaker notion than the strict stabilization: the *topology-aware strict stabilization* (denoted by TA strict stabilization for short) introduced by [5]. 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 (*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 location.

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

Definition 9 (S_B-correct node) *A node is S_B-*correct *if it is a correct node (*i.e. *not Byzantine) which not belongs to S_B.*

Definition 10 (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 11 ((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 11 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 12 ((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 $S_B = \left\{ v \in V | \min_{b \in B} (d(v, b)) \leq c \right\}$, then a (S_B, f)-topology-aware strictly stabilizing protocol is a (c, f)-strictly stabilizing protocol. Then, the concept of topology-aware strict stabilization is a generalization of the strict stabilization. However, note that a TA strictly stabilizing protocol is stronger than a classical self-stabilizing protocol (that may never meet their 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.

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 [6]. We recall in the following the formal definition of this concept.

Definition 13 (S_B-stable configuration) *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.*

Definition 14 (S_B-TA-disruption) *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:*

1. *e is finite,*
2. *e contains at least one action of a S_B-correct process for changing the value of an O-variable,*
3. *ρ_0 is S_B-legitimate for spec and S_B-stable, and*
4. *ρ_t is the first configuration after ρ_0 such that ρ_t is S_B-legitimate for spec and S_B-stable.*

Definition 15 ((t, k, S_B, f)-TA time contained configuration) *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:*

1. *ρ_0 is S_B-legitimate for* spec *and S_B-stable,*
2. *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),*
3. *every execution starting from ρ_0 contains at most t S_B-TA-disruptions, and*
4. *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 16 ((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 actions of each S_B-correct node. Parameters l and k are respectively the (t, S_B, f)-stabilization time and the (t, S_B, f)-process-disruption time of A.*

3 Maximum Metric Tree Construction

3.1 Definition and Specification

In this work, we deal with maximum (routing) metric trees as defined in [9]. Informally, the goal of a routing protocol is to construct a tree that simultaneously maximizes the metric values of all of the nodes with respect to some total ordering $\prec$. In the following, we recall all definitions and notations introduced in [9].

Definition 17 (Routing metric) *A* routing metric *(or just* metric*) is a five-tuple* $(M, W, met, mr, \prec)$ *where:*

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

Any metric value $m \in M \setminus \{mr\}$ satisfies the utility condition *(that is, there 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 four classical metrics with this model: the shortest path metric ($\mathcal{SP}$), the flow metric ($\mathcal{F}$), and the reliability metric ($\mathcal{R}$). Note also that we can modelise the construction of a spanning tree with no particular constraints in this model using the metric $\mathcal{NC}$ described below and the construction of a BFS spanning tree using the shortest path metric ($\mathcal{SP}$) with $W_1 = \{1\}$ (we denoted this metric by $\mathcal{BFS}$ in the following).

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

$$
\begin{array}{rcl}
\mathcal{R} & = & (M_3, W_3, met_3, mr_3, \prec_3) \\
\text{where} & & M_3 = [0, 1] \\
& & W_3 = [0, 1] \\
& & met_3(m, w) = m * w \\
& & mr_3 = 1 \\
& & \prec_3 \text{ is the classical } < \text{ relation}
\end{array}
\qquad
\begin{array}{rcl}
\mathcal{NC} & = & (M_4, W_4, met_4, mr_4, \prec_4) \\
\text{where} & & M_4 = \{0\} \\
& & W_4 = \{0\} \\
& & met_4(m, w) = 0 \\
& & mr_4 = 0 \\
& & \prec_4 \text{ is the classical } < \text{ relation}
\end{array}
$$

Definition 18 (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 19 (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 $\forall i \in \{1, \ldots, k-1\}, m_i = met(m_{i-1}, wf(\{v_i, v_{i-1}\})$ and $m_0 = mr$.

Definition 20 (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 21 (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 22 (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 [9] is the study of metrics that always allow the construction of a maximum metric tree. More formally, the definition follows.

Definition 23 (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.*

Given a maximizable metric $\mathcal{M} = (M, W, mr, met, \prec)$, the aim of this work is to study the construction of 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 two O-variables: a pointer to its parent in its tree ($prnt_v \in N_v \cup \{\perp\}$) and a level which stores its current metric value ($level_v \in M$). 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 24 ($\mathcal{M}$-path) *Given an assigned metric $\mathcal{M} = (M, W, mr, met, \prec, wf)$ over a system S, a path $(v_0, \ldots, v_k)$ $(k \geq 1)$ of S is a* $\mathcal{M}$-path *if and only if:*

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

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

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

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

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

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

3.2 Previous results

In this section, we summarize known results about maximum metric tree construction. The first interesting result about maximizable metrics is due to [9] that provides a fully characterization of maximizable metrics as follow.

Definition 25 (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 26 (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 [9]) *A metric is maximizable if and only if this metric is bounded and monotonic.*

Secondly, [8] provides a self-stabilizing protocol to construct a maximum metric tree with respect to any maximizable metric. Now, we focus on self-stabilizating solutions resilient to Byzantine faults. Following discussion of Section 2, it is obvious that there exists no strictly stabilizing protocol for this problem. If we consider the weaker notion of topology-aware strict stabilization, [5] 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, [5] proves that the algorithm introduced for the maximum metric spanning tree construction in [8] performed this optimal containment area. More formally, [5] proves the following results.

Theorem 2 ([5]) *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* $A_B \subsetneq S_B$.

Theorem 3 ([5]) *Given a maximizable metric* $\mathcal{M} = (M, W, mr, met, \prec)$*, the protocol of [8] is a* $(S_B, n-1)$*-TA strictly stabilizing protocol for maximum metric spanning tree construction with respect to* $\mathcal{M}$.

Some others works try to circumvent the impossibility result of strict stabilization using the concept ot strong stabilization but do not provide results for any maximizable metric. Indeed, [7] proves the following result about spanning tree.

Theorem 4 ([7]) *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, [6] proved the following impossibility result.

Theorem 5 ([6]) *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.*

These two results motivate our result related to strong stabilization in the general case (see Section 4.1) that proves a necessary condition on the maximizable metric to allow strong stabilization.

Now, if we focus on topology-aware strong stabilization, [6] proved the following results.

Theorem 6 ([6]) *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 $A_B^* \subsetneq \{v \in V | \min_{b \in B}(d(v, b)) < d(r, v)\}$ and t is a finite integer.*

Theorem 7 ([6]) *The protocol of [10] 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 and $S_B^* = \{v \in V | \min_{b \in B}(d(v, b)) < d(r, v)\}$.*

In the following, we generalize the Theorem 6 to any maximizable metric (see Section 4.2).

4 Necessary conditions

In this section, we provide our necessary conditions about containment radius (respectively area) of any strongly stabilizing (respectively TA strongly stabilizing) protocol for the maximum metric tree construction.

4.1 Strong Stabilization

We introduce here some new definitions to characterize some important properties of maximizable metrics that are used in the following.

Definition 27 (Strictly decreasing metric) *A metric $\mathcal{M} = (M, W, mr, met, \prec)$ 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 $\forall w \in W, met(m, w) = m$.*

Definition 28 (Fixed point) *A metric value m is a* fixed point *of a metric $\mathcal{M} = (M, W, mr, met, \prec)$ if $m \in M$ and if for any value $w \in W$, we have: $met(m, w) = m$.*

Then, we define a specific class of maximizable metrics and we prove that it is possible to construct a maximum metric tree in a strongly-stabilizing way only if we consider such a metric.

Definition 29 (Strongly maximizable metric) *A maximizable metric $\mathcal{M} = (M, W, mr, met, \prec)$ is strongly maximizable if and only if $|M| = 1$ or if the following properties holds:*

- $|M| \geq 2$,
- $\mathcal{M}$ *is strictly decreasing, and*
- $\mathcal{M}$ *has one and only one fixed point.*

Note that $\mathcal{NC}$ is a strongly maximizable metric (since $|M_4| = 1$) whereas $\mathcal{BFS}$ or $\mathcal{SP}$ are not (since the first one has no fixed point, the second is not strictly decreasing). If we consider the metric $\mathcal{MET}$ defined below, we can show that $\mathcal{MET}$ is a strongly maximizable metric such that $|M| \geq 2$.

$$\begin{array}{lcl} \mathcal{MET} & = & (M_5, W_5, met_5, mr_5, \prec_5) \\ \text{where} & & M_5 = \{0,1,2,3\} \\ & & W_5 = \{1\} \\ & & met_5(m,w) = max\{0, m-w\} \\ & & mr_5 = 3 \\ & & \prec_5 \text{ is the classical } < \text{ relation} \end{array}$$

Now, we can state our first necessary condition.

Theorem 8 *Given a maximizable metric $\mathcal{M} = (M, W, mr, met, \prec)$, 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:*

$$\left\{ \begin{array}{l} \mathcal{M} \text{ is not a strongly maximizable metric} \\ \quad or \\ c < |M| - 2 \end{array} \right.$$

Proof We prove this result by contradiction. We assume so that $\mathcal{M} = (M, W, mr, met, \prec)$ is a maximizable metric such that there exist a finite integer t and a protocol $\mathcal{P}$ that is a $(t, c, 1)$-strongly stabilizing protocol for maximum metric spanning tree construction with respect to $\mathcal{M}$. We distinguish the following cases (note that they are exhaustive):

Case 1: $\mathcal{M}$ is a strongly maximizing metric and $c < |M| - 2$.

As $c \geq 0$, we know that $|M| \geq 2$ and by definition of a strongly stabilizing metric, $\mathcal{M}$ is strictly decreasing, and $\mathcal{M}$ has one and only one fixed point.

By assumption on $\mathcal{M}$, we know that there exist $c+3$ distinct metric values $m_0 = mr, m_1, \ldots, m_{c+2}$ in M and $w_0, w_1, \ldots, w_{c+1}$ in W such that: $\forall i \in \{1, \ldots, c+2\}, m_i = met(m_{i-1}, w_{i-1}) \prec m_{i-1}$.

Let $S = (V, E, \mathcal{W})$ be the following weighted system $V = \{p_0 = r, p_1, \ldots, p_{2c+2}, p_{2c+3} = b\}$, $E = \{\{p_i, p_{i+1}\}, i \in \{0, \ldots, 2c+2\}\}$ and $\forall i \in \{0, c+1\}, w_{p_i,p_{i+1}} = w_{p_{2c+3-i},p_{2c+2-i}} = w_i$. Note that the choice $w_{p_{c+1},p_{c+2}} = w_{c+1}$ ensures us the following property when $level_r = level_b = mr$: $\mu(p_{c+1}, b) \prec \mu(p_{c+1}, r)$ (and by symmetry, $\mu(p_{c+2}, r) \prec \mu(p_{c+2}, b)$). Process p_0 is the real root and process b is a Byzantine one. Note that the construction of $\mathcal{W}$ ensures the following properties when $level_r = level_b = mr$: $\forall i \in \{1, \ldots, c+1\}, \mu(p_i, r) = \mu(p_{2c+3-i}, b)$, $\mu(p_i, b) \prec \mu(p_i, r)$ and $\mu(p_{2c+3-i}, r) \prec \mu(p_{2c+3-i}, b)$.

Assume that the initial configuration ρ_0 of S satisfies: $prnt_r = prnt_b = \perp$, $level_r = level_b = mr$, and other variables of b (if any) are identical to those of r (see Figure 1, variables of other processes may be arbitrary). Assume now that b takes exactly the same actions as r (if any)

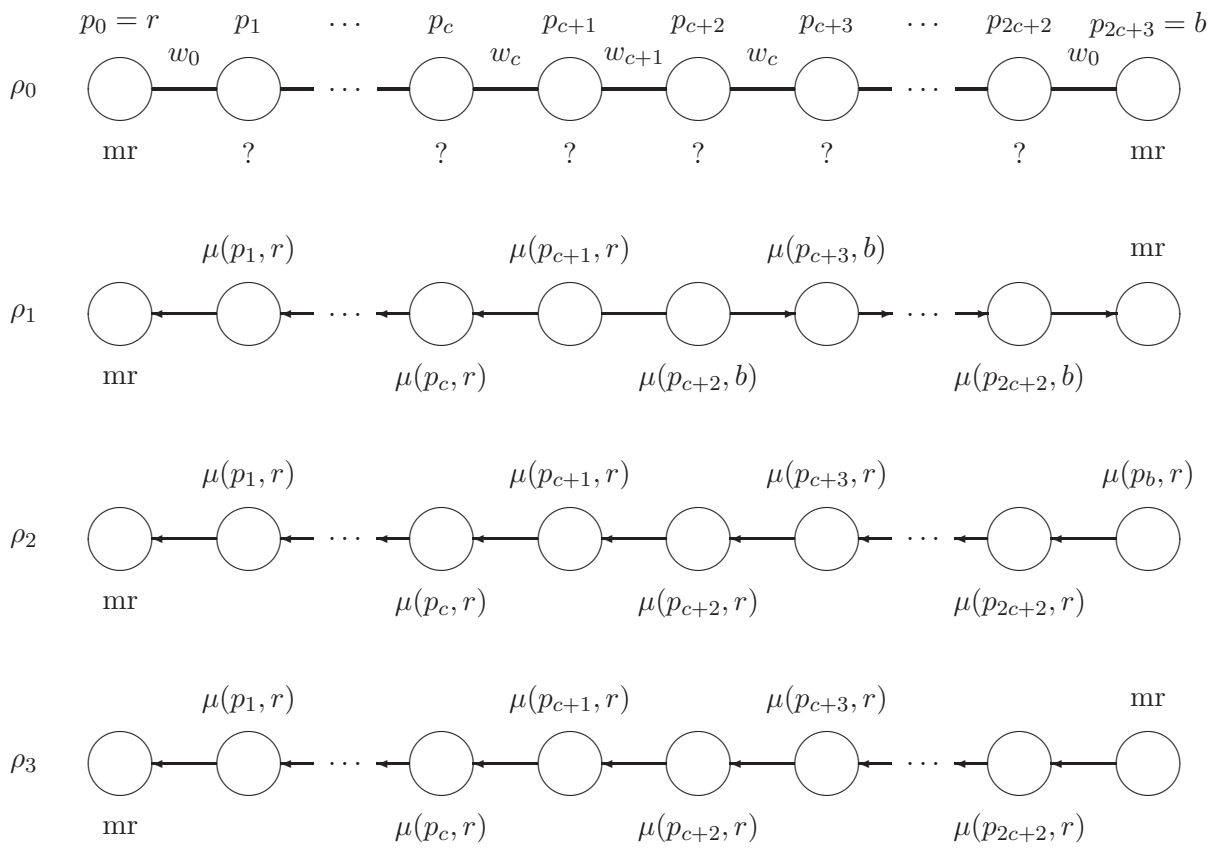

Figure 1: Configurations used in proof of Theorem 8, case 1.

immediately after r. Then, by symmetry of the execution and by convergence of $\mathcal{P}$ to *spec*, we can deduce that the system reaches in a finite time a configuration ρ_1 (see Figure 1) in which: $\forall i \in \{1, \ldots, c+1\}, prnt_{p_i} = p_{i-1}$, $level_{p_i} = \mu(p_i, r) = m_i$ and $\forall i \in \{c+2, \ldots, 2c+2\}, prnt_{p_i} = p_{i+1}$ and $level_{p_i} = \mu(p_i, b) = m_{2c+3-i}$ (because this configuration is the only one in which all correct process v satisfies $spec(v)$ when $prnt_r = prnt_b = \perp$ and $level_r = level_b = mr$ by construction of $\mathcal{W}$). Note that ρ_1 is c-legitimate and c-stable.

Assume now that the Byzantine process acts as a correct process and executes correctly its algorithm. Then, by convergence of $\mathcal{P}$ in fault-free systems (remember that a strongly-stabilizing algorithm is a special case of self-stabilizing algorithm), we can deduce that the system reach in a finite time a configuration ρ_2 (see Figure 1) in which: $\forall i \in \{1, \ldots, 2c+3\}, prnt_{p_i} = p_{i-1}$ and $level_{p_i} = \mu(p_i, r)$ (because this configuration is the only one in which all process v satisfies $spec(v)$). Note that the portion of execution between ρ_1 and ρ_2 contains at least one c-perturbation (p_{c+2} is a c-correct process and modifies at least once its O-variables) and that ρ_2 is c-legitimate and c-stable.

Assume now that the Byzantine process b takes the following state: $prnt_b = \perp$ and $level_b = mr$. This step brings the system into configuration ρ_3 (see Figure 1). From this configuration, we can repeat the execution we constructed from ρ_0. By the same token, we obtain an execution of $\mathcal{P}$ which contains c-legitimate and c-stable configurations (see ρ_1) and an infinite

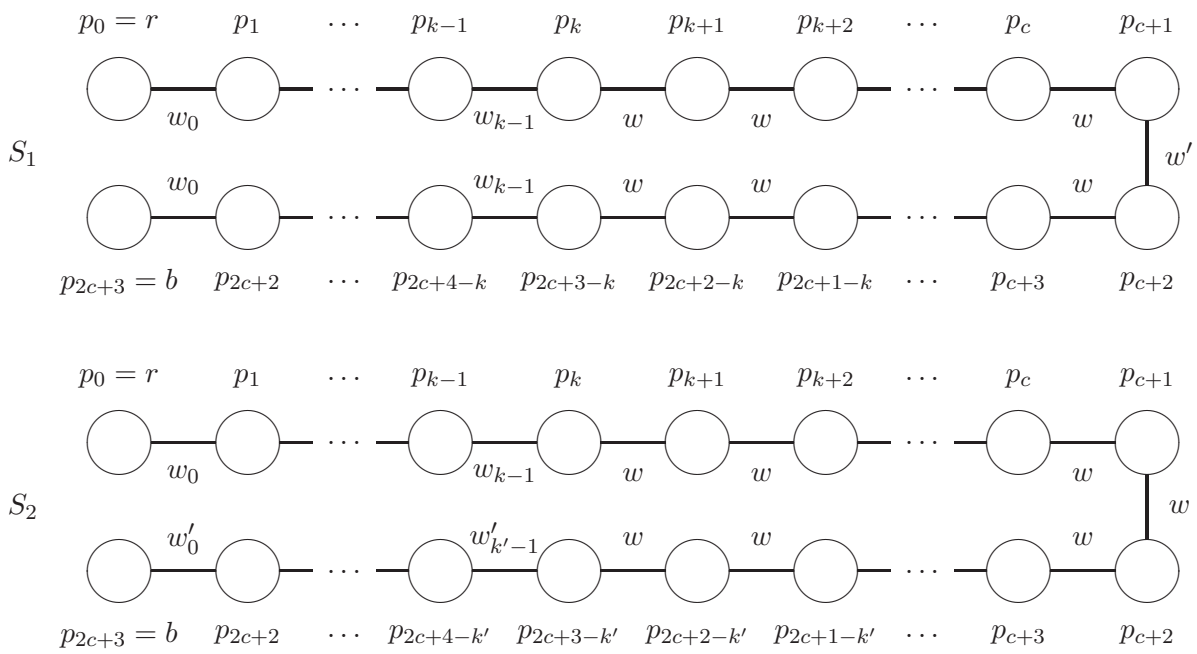

Figure 2: Configurations used in proof of Theorem 8, cases 2 and 3.

number of c-perturbation which contradicts the $(t, c, 1)$-strong stabilization of $\mathcal{P}$.

Case 2: $\mathcal{M}$ is not strictly decreasing.

By definition, we know that $\mathcal{M}$ is not a strongly maximizable metric. Hence, we have $|M| \geq 2$. Then, the definition of a strictly decreasing metric implies that there exists a metric value $m \in M$ such that: $\exists w \in W$, $met(m, w) = m$ and $\exists w' \in W, m' = met(m, w') \prec m$ (and thus m is not a fixed point of $\mathcal{M}$). By the utility condition on M, we know that there exists a sequence of metric values $m_0 = mr, m_1, \ldots, m_l = m$ in M and $w_0, w_1, \ldots, w_{l-1}$ in W such that $\forall i \in \{1, \ldots, l\}, m_i = met(m_{i-1}, w_{i-1})$. Denote by k the length of the shortest such sequence. Note that this implies that $\forall i \in \{1, \ldots, k\}, m_i \prec m_{i-1}$ (otherwise we can remove m_i from the sequence and this is contradictory with the construction of k). We distinguish the following cases:

Case 2.1: $k \geq c + 2$.

We can use the same token as case 1 above by using w' instead of w_{c+1} in the case where $k = c + 2$ (since we know that $met(m, w') \prec m$).

Case 2.2: $k < c + 2$.

Let $S_1 = (V, E, \mathcal{W})$ be the following weighted system $V = \{p_0 = r, p_1, \ldots, p_{2c+2}, p_{2c+3} = b\}$, $E = \{\{p_i, p_{i+1}\}, i \in \{0, \ldots, 2c + 2\}\}$, $\forall i \in \{0, \ldots, k - 1\}, w_{p_i,p_{i+1}} = w_{p_{2c+3-i},p_{2c+2-i}} = w_i$, $\forall i \in \{k, \ldots, c\}, w_{p_i,p_{i+1}} = w_{p_{2c+3-i},p_{2c+2-i}} = w$ and $w_{p_{c+1},p_{c+2}} = w'$ (see Figure 2). Note that this choice ensures us the following property when $level_r = level_b = mr$: $\mu(p_{c+1}, b) \prec \mu(p_{c+1}, r)$ (and by symmetry, $\mu(p_{c+2}, r) \prec \mu(p_{c+2}, b)$). Process p_0 is the real root and process b is a Byzantine one. Note that the construction of $\mathcal{W}$ ensures the following properties when $level_r = level_b = mr$: $\forall i \in \{1, \ldots, c + 1\}, \mu(p_i, r) = \mu(p_{2c+3-i}, b)$, $\mu(p_i, b) \prec \mu(p_i, r)$ and $\mu(p_{2c+3-i}, r) \prec \mu(p_{2c+3-i}, b)$.

This construction allows us to follow the same proof as in case 1 above.

Case 3: $\mathcal{M}$ has no or more than two fixed point, and is strictly decreasing.

If $\mathcal{M}$ has no fixed point and is strictly decreasing, then $|M|$ is not finite and then, we can apply the result of case 1 above since c is a finite integer.

If $\mathcal{M}$ has two or more fixed points and is strictly decreasing, denote by Υ and Υ' two fixed points of $\mathcal{M}$. Without loss of generality, assume that $\Upsilon \prec \Upsilon'$. By the utility condition on M, we know that there exists sequences of metric values $m_0 = mr, m_1, \ldots, m_l = \Upsilon$ and $m'_0 = mr, m'_1, \ldots, m'_{l'} = \Upsilon'$ in M and $w_0, w_1, \ldots, w_{l-1}$ and $w'_0, w'_1, \ldots, w'_{l'-1}$ in W such that $\forall i \in \{1, \ldots, l\}, m_i = met(m_{i-1}, w_{i-1})$ and $\forall i \in \{1, \ldots, l'\}, m'_i = met(m'_{i-1}, w'_{i-1})$. Denote by k and k' the length of shortest such sequences. Note that this implies that $\forall i \in \{1, \ldots, k\}, m_i \prec m_{i-1}$ and $\forall i \in \{1, \ldots, k'\}, m'_i \prec m'_{i-1}$ (otherwise we can remove m_i or m'_i from the corresponding sequence). We distinguish the following cases:

Case 3.1: $k > c + 2$ or $k' > c + 2$.

Without loss of generality, assume that $k > c + 2$ (the second case is similar). We can use the same token as case 1 above.

Case 3.2: $k \leq c + 2$ and $k' \leq c + 2$.

Let w be an arbitrary value of W. Let $S_2 = (V, E, \mathcal{W})$ be the following weighted system $V = \{p_0 = r, p_1, \ldots, p_{2c+2}, p_{2c+3} = b\}$, $E = \{\{p_i, p_{i+1}\}, i \in \{0, \ldots, 2c + 2\}\}$, $\forall i \in \{0, k - 1\}, w_{p_i,p_{i+1}} = w_i$, $\forall i \in \{0, k' - 1\}, w_{p_{2c+3-i},p_{2c+2-i}} = w'_i$ and $\forall i \in \{k, 2c + 2 - k'\}, w_{p_i,p_{i+1}} = w$ (see Figure 2). Note that this choice ensures us the following property when $level_r = level_b = mr$: $\mu(p_{c+1}, r) = \Upsilon \prec \Upsilon' = \mu(p_{c+1}, b)$ and $\mu(p_{c+2}, r) = \Upsilon \prec \Upsilon' = \mu(p_{c+2}, b)$. Process p_0 is the real root and process b is a Byzantine one.

This construction allows us to follow a similar proof as in case 1 above (note that any process u which satisfies $\mu(u, r) \prec \Upsilon'$ will be disturb infinitely often, in particular at least p_{c+1} and p_{c+2} which contradicts the $(t, c, 1)$-strong stabilization of $\mathcal{P}$).

In any case, we show that there exists a system which contradicts the $(t, c, 1)$-strong stabilization of $\mathcal{P}$ that ends the proof. □

4.2 Topology Aware Strong Stabilization

First, we generalize the set S^*_B previously defined for the $\mathcal{BFS}$ metric in [6] to any maximizable metric $\mathcal{M} = (M, W, mr, met, \prec)$.

$$S^*_B = \left\{ v \in V \setminus B \,\middle|\, \mu(v, r) \prec \underset{b \in B}{max_{\prec}} \{\mu(v, b)\} \right\}$$

Intuitively, S^*_B gathers the set of corrects processes that are strictly closer (according to $\mathcal{M}$) to a Byzantine process than the root. Figures from 3 to 5 provide some examples of containment areas with respect to several maximizable metrics and compare it to S_B, the optimal containment area for TA strict stabilization.

Now, we can state our generalization of Theorem 6.

Theorem 9 *Given a maximizable metric $\mathcal{M} = (M, W, mr, met, \prec)$, 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.*

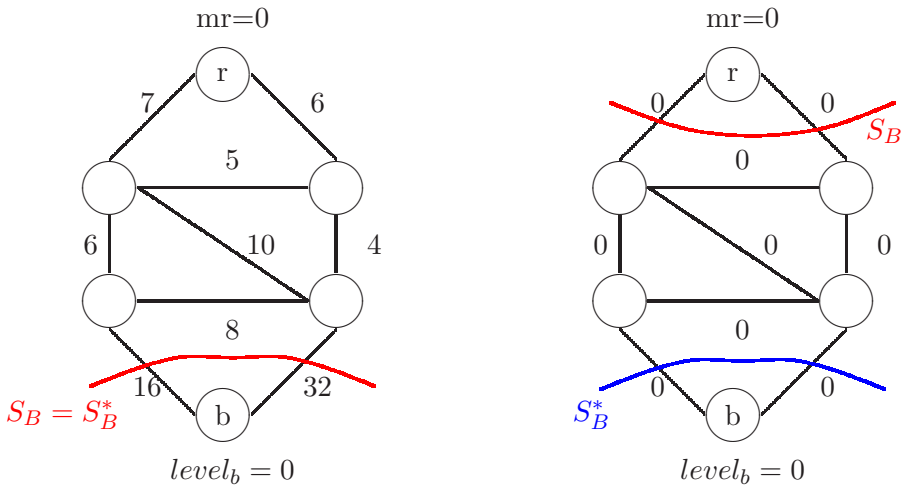

Figure 3: Examples of containment areas for SP spanning tree construction.

Proof Let $\mathcal{M} = (M, W, mr, met, \prec)$ be a maximizable metric and $\mathcal{P}$ be a $(t, A_B^*, 1)$-TA-strongly stabilizing protocol for maximum metric spanning tree construction protocol with respect to $\mathcal{M}$ where $A_B^* \subsetneq S_B^*$ and t is a finite integer. 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, we have $\mu(v, r) = \underset{b \in B}{min_{\prec}}\{\mu(v, b)\} = m$. Consequently, $S_B^* = \emptyset$ for any system. Then, it is absurd to have $A_B^* \subsetneq S_B^*$.

Case 2: $|M| \geq 2$.

By definition of a bounded metric, we can deduce that there exists $m \in M$ 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 $\mathcal{M}$.

Let S be a system such that any edge incident to the root or a Byzantine process has a weight equals to w. Then, we can deduce that we have: $m = \underset{b \in B}{max_{\prec}}\{\mu(r, b)\} \prec \mu(r, r) = mr$ and for any correct process $v \neq r$, $\mu(v, r) = \underset{b \in B}{max_{\prec}}\{\mu(v, b)\} = m$. Hence, $S_B^* = \emptyset$ for any such system. Then, it is absurd to have $A_B^* \subsetneq S_B^*$.

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: $V = \{r, u, u', v, v', b\}$, $E = \{\{r, u\}, \{r, u'\}, \{u, v\}, \{u', v'\}, \{v, b\}, \{v', b\}\}$, $w_{r,u} = w_{r,u'} = w_{v,b} = w_{v',b} = w$, 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: $prnt_r = prnt_b = \perp$, $level_r = level_b = mr$, and $prnt$, $level$ variables of other processes are arbitrary (see Figure 6, other variables may have arbitrary values but other variables of b are identical to those of r).

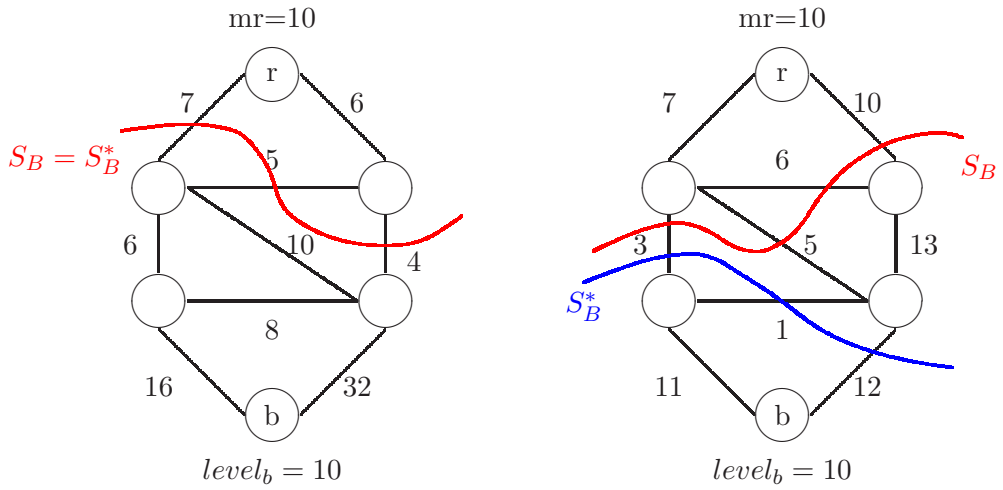

Figure 4: Examples of containment areas for flow spanning tree construction.

Assume now that b takes exactly the same actions as r (if any) immediately after r (note that $r \notin A_B^*$ and hence $prnt_r = \perp$ and $level_r = mr$ still hold by closure and then $prnt_b = \perp$ and $level_b = mr$ still hold too). Then, by symmetry of the execution and by convergence of $\mathcal{P}$ to *spec*, we can deduce that the system reaches in a finite time a configuration ρ_1 (see Figure 6) in which: $prnt_r = prnt_b = \perp$, $prnt_u = prnt_{u'} = r$, $prnt_v = prnt_{v'} = b$, $level_r = level_b = mr$, and $level_u = level_{u'} = level_v = level_{v'} = m$ (because this configuration is the only one in which all correct process v satisfies $spec(v)$ when $prnt_r = prnt_b = \perp$ and $level_r = level_b = mr$ since $met(m, w') \prec m$). Note that ρ_1 is A_B^*-legitimate for *spec* and A_B^*-stable (whatever A_B^* is).

Assume now that b behaves as a correct processor with respect to $\mathcal{P}$. Then, by convergence of $\mathcal{P}$ in a fault-free system starting from ρ_1 which is not legitimate (remember that a TA-strongly stabilizing algorithm is a special case of self-stabilizing algorithm), we can deduce that the system reach in a finite time a configuration ρ_2 (see Figure 6) in which: $prnt_r = \perp$, $prnt_u = prnt_{u'} = r$, $prnt_v = u$, $prnt_{v'} = u'$, $prnt_b = v$ (or $prnt_b = v'$), $level_r = mr$, $level_u = level_{u'} = m$ $level_v = level_{v'} = met(m, w') = m'$, and $level_b = met(m', w) = m''$. Note that processes v and v' modify their O-variables in the portion of execution between ρ_1 and ρ_2 and that ρ_2 is A_B^*-legitimate for *spec* and A_B^*-stable (whatever A_B^* is). Consequently, this portion of execution contains at least one A_B^*-TA-disruption (whatever A_B^* is).

Assume now that the Byzantine process b takes the following state: $prnt_b = \perp$ and $level_b = mr$. This step brings the system into configuration ρ_3 (see Figure 6). From this configuration, we can repeat the execution we constructed from ρ_0. By the same token, we obtain an execution of $\mathcal{P}$ which contains c-legitimate and c-stable configurations (see ρ_1) and an infinite number of A_B^*-TA-disruption (whatever A_B^* is) which contradicts the $(t, A_B^*, 1)$-TA-strong stabilization of $\mathcal{P}$.

□

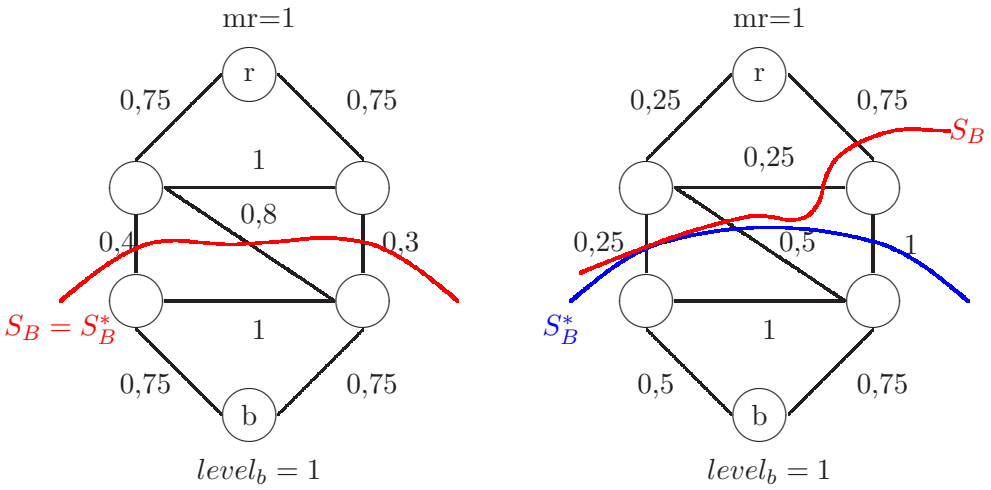

Figure 5: Examples of containment areas for reliability spanning tree construction.

5 Conclusion

In this paper, we presented two necessary conditions to achieve strong stabilization and topology-aware strong stabilization in maximum metric tree construction. Our work obviously leads to the following open question: is there a topology-aware strongly stabilizing protocol that ensures a containmemt area equal to S_B^*? We conjecture that it is the case.

References

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

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

[3] Shlomi. Dolev. *Self-stabilization*. MIT Press, March 2000.

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

[5] Swan Dubois, Toshimitsu Masuzawa, and Sébastien Tixeuil. The impact of topology on byzantine containment in stabilization. In *Proceedings of DISC 2010*, Lecture Notes in Computer Science, Boston, Massachusetts, USA, September 2010. Springer Berlin / Heidelberg.

[6] Swan Dubois, Toshimitsu Masuzawa, and Sébastien Tixeuil. On byzantine containment properties of the min+1 protocol. In *Proceedings of SSS 2010*, Lecture Notes in Computer Science, New York, NY, USA, September 2010. Springer Berlin / Heidelberg.

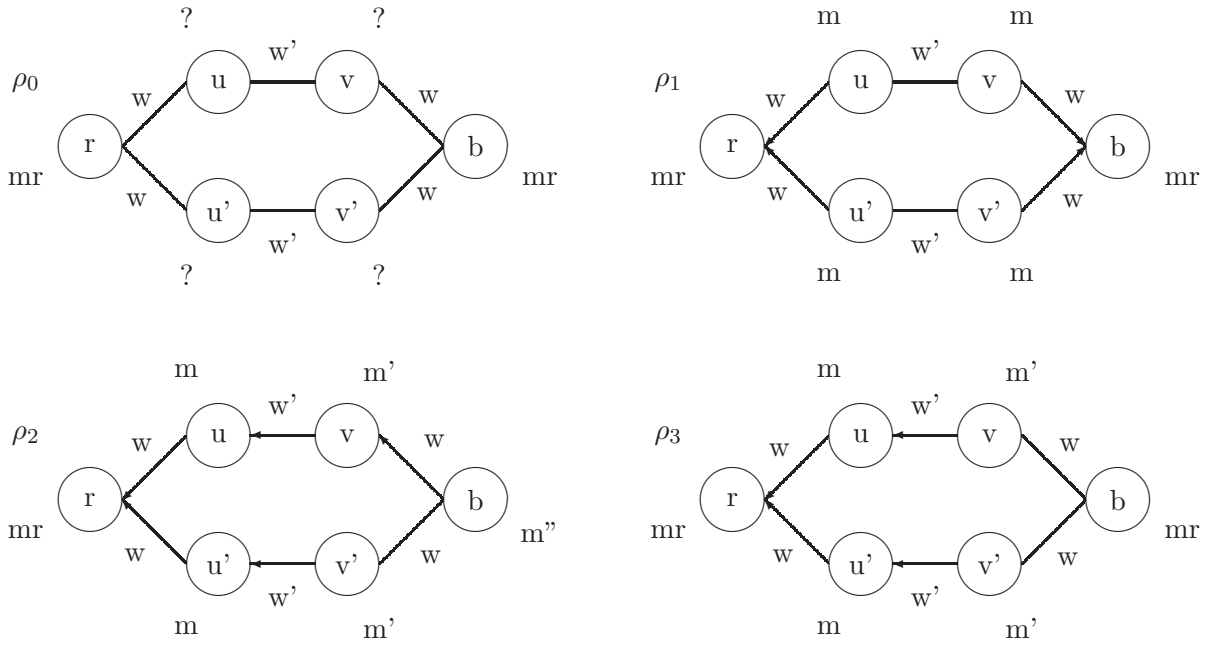

Figure 6: Configurations used in proof of Theorem 9.

[7] Swan Dubois, Toshimitsu Masuzawa, and Sébastien Tixeuil. Bounding the impact of unbounded attacks in stabilization. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 2011.

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

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

[10] Shing-Tsaan Huang and Nian-Shing Chen. A self-stabilizing algorithm for constructing breadth-first trees. *Inf. Process. Lett.*, 41(2):109–117, 1992.

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

[12] Toshimitsu Masuzawa and Sébastien Tixeuil. Bounding the impact of unbounded attacks in stabilization. In Ajoy Kumar Datta and Maria Gradinariu, editors, *SSS*, volume 4280 of *Lecture Notes in Computer Science*, pages 440–453. Springer, 2006.

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

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

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