Algorithmica manuscript No.
(will be inserted by the editor)

Maximum Metric Spanning Tree made Byzantine Tolerant

Swan Dubois · Toshimitsu Masuzawa · Sébastien Tixeuil

Received: date / Accepted: date

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. Combining these two properties is known to induce many impossibility results. Hence, there exist several fault tolerance schemes to contain Byzantine faults in self-stabilization.

S. Dubois
Sorbonne Universités, UPMC Université Paris 6, Équipe REGAL, LIP6
CNRS, UMR 7606, LIP6
Inria, Équipe-projet REGAL LIP6, Case 26-00/207, 5 place Jussieu, 75252 Paris Cedex 05, France
E-mail: swan.dubois@lip6.fr

T. Masuzawa
Osaka University
Yamadaoka 1-5, Suita, Osaka 565-0871, Japan
E-mail: masuzawa@ist.osaka-u.ac.jp

S. Tixeuil
Sorbonne Universités, UPMC Université Paris 6, Équipe NPA, LIP6
CNRS, UMR 7606, LIP6
Institut Universitaire de France
LIP6, Case 26-00/113, 5 place Jussieu, 75252 Paris Cedex 05, France
E-mail: sebastien.tixeuil@lip6.fr

A preliminary version of this work appears in the proceedings of the 25th and of the 24th International Symposium on Distributed Computing (DISC'10 and DISC'11), see [17, 19].

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

In this paper, we consider the well known problem of constructing a maximum metric tree in this context. We provide a new distributed protocol that ensures the best possible containment with respect to topology-aware strict and strong stabilization.

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* [12, 13, 35] is a versatile technique that permits forward recovery from any number and any kind of *transient* faults, while *Byzantine Fault-tolerance* [31] 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 [14, 9, 33] as impossibility results are expected in many cases.

A promising 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 [33, 32]. Strict stabilization guarantees that there exists a *containment radius* outside which the effect of permanent Byzantine 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.

This paper proposes to study a new Byzantine fault containment scheme in self-stabilization in order to circumvent those impossibility results. The weaker notion of *strong stabilization* was proposed [20]: here, correct vertices outside the containment radius may be perturbated by the actions of Byzantine vertices, 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 [18] and denoted by *topology aware* strict (and strong) stabilization.

Contributions In this paper, we illustrate the effectiveness of these Byzantine containment schemes by studying a global problem: the maximum metric spanning tree construction. The goal is to construct a particular spanning tree that maximizes the metric of all vertices in the system with respect to a given metric. Indeed, even if any spanning tree minimizes by definition the number of communication links, all are not equivalent. We can consider some criteria to distinguish spanning trees. For example, a breadth-first search (BFS) spanning tree [28, 16, 1] allows us to minimize the delay of communication between any vertex and a distinguished one (called the root of the tree) and a minimum weight spanning tree [23] minimizes the global weight of the tree (in the case of weighted communication graph).

In this paper, we focus on a large class of spanning tree constructions: the maximum metric spanning tree construction with respect to any maximizable metric [27]. Intuitively, a metric is a scheme to compute a distance between two vertices along any path of the communication graph. A metric is maximizable if there always exists a spanning tree that maximizes the metric of each vertex of any communication graph with respect to a distinguished vertex called the root. For example, the shortest path [36] or the flow metric [25] are maximizable. In contrast, there exists no maximizable metric to model the minimum weight [23] or the minimum degree [2, 4] spanning tree construction. The large span of this class of metrics motivates some previous works [26, 27].

Contributions of this paper are twofold. First, motivated by impossibility results of strict stabilization for global tasks [33] (as the spanning tree construction), we define three concepts for Byzantine containment in self-stabilization. These new concepts, respectively called strong stabilization, topology-aware strict stabilization, and topology-aware strong stabilization, weaken the constraints on the containment radius of strict stabilization in order to by-pass such impossibility results. In strong stabilization, the containment radius is weakened in time since we allow correct vertices outside the containment radius to be disturbed a finite number of times by Byzantine ones after the convergence to a legitimate configuration. In topology-aware stabilization, the weakening is in space since we generalize the containment radius to a containment area. This containment area is simply the set of correct vertices (that is a function of the communication graph) that may be infinitely often disturbed by Byzantine vertices. Note that this weakening in the containment radius into containment area may be applied both to strict and strong stabilization. Then, we prove the effectiveness of our new concepts of Byzantine containment in self-stabilization using maximum metric spanning tree construction as a benchmark. We design a distributed protocol that achieves the optimal containment areas for topology-aware strict and strong stabilization for maximum metric spanning construction for any maximizable metric.

Related Works Spanning tree construction was extensively studied in the context of distributed systems either in a fault-free setting or in presence of faults. In fault-free distributed systems, there exist a number of adaptations of centralized protocols to construct spanning trees with respect to numerous properties (*e.g.* minimum weight spanning trees [23], minimum degree spanning trees [2], or Steiner trees [7]).

Gärtner presents in [24] a good survey on self-stabilizing distributed protocols for spanning tree construction for the three simplest properties: depth-first search spanning trees, breadth-first spanning trees and shortest path spanning trees. The first self-stabilizing distributed protocol to construct a depth-first search spanning tree was by Collin and Dolev [8]. Note that any self-stabilizing token circulation (see *e.g.* [29, 10]) may be used to construct such a spanning tree. Regarding breadth-first search spanning tree construction, the first self-stabilizing solution was by Dolev, Israeli and Moran [15, 16] but a simpler one was provided by [28]. Finally, self-stabilizing solutions to shortest path spanning tree construction may be found in [30, 5] for example. Note that there also exist self-stabilizing distributed protocols for more complex properties of spanning trees as the minimum diameter spanning trees [6], minimum degree spanning trees [4], or Steiner spanning trees [3].

Gouda and Schneider [27] defined a large class of spanning tree constructions using the concept of maximizable metric. In this work, the metric of a path of the communication graph is the result of the application of the metric operator to the value of each edge of the path. For example, the shortest path metric associates a weight (a natural number) to each edge of the communication graph and the metric of a path is computed by the sum of the weight of each edge of the path. A metric is maximizable if there always exists a spanning tree that maximizes the metric of each vertex of any communication graph with respect to a distinguished vertex called the root. This concept of maximizable metric enclosed a lot of classical metrics such as breadth-first search, shortest path or flow metrics (defined in [34]), that justifies the interest in maximum metric spanning tree construction. The main result of [27] is a full characterization of maximizable metrics. A self-stabilizing distributed protocol for maximum metric spanning tree construction with respect to any maximizable metric is provided by [26].

The idea to provide a generic distributed protocol able to compute a spanning tree according to a large class of metric motivates other works. For instance, another formalism that encompasses a different set of metrics is presented in [21, 22, 11].

To our knowledge, there exists no distributed protocol for spanning tree construction in presence of both transient and Byzantine faults except our previous work on simple properties as depth-first search spanning tree [20] or breadth-first search spanning tree [18]. This paper proposes to generalize these results by studying the maximum metric spanning tree construction in distributed systems simultaneously subject to transient and Byzantine faults.

Organization of the paper Section 2 presents formally the model of computation used in this paper. Then, we state formal definitions of our Byzantine containment schemes in Section 3 while Section 4 is devoted to the specification of the maximum metric spanning tree construction. We present and prove our topology-aware stabilizing distributed protocol in Section 5. Finally, Section 6 proves the optimality of containment areas presented in the previous section. Section 7 concludes the paper.

2 State Model

A *distributed system* $g = (V, L)$ consists of a set $V = \{v_1, v_2, \dots, v_n\}$ of vertices (*a.k.a.* processes) and a set L of bidirectional communication links (simply called links). A link is an unordered pair of distinct vertices. Vertices u and v are called *neighbors* if $(u, v) \in L$. The set of neighbors of a vertex v is denoted by N_v. We do not assume existence of a unique identifier for each vertex.

Instead we assume each vertex can distinguish its neighbors from each other by locally labeling them.

A distributed system g can be regarded as a (communication) graph whose vertex set is V and whose link set is L, so we use graph terminology to describe a distributed system g. We use the following notations: $n = |V|$, $m = |L|$, $dist(g, u, v)$ denotes the distance between two vertices u and v in g (*i.e.* the length of the shortest path of g between u and v), and $deg(g)$ denotes the maximal degree of g (*i.e.* the maximal number of neighbors of a vertex in g) while $diam(g)$ denotes the diameter of g (*i.e.* the largest distance between two vertices of g).

In this paper, we consider distributed systems of arbitrary topology. We assume that a single vertex is distinguished as a *root* (denoted by r), and all the other vertices are identical. We adopt the *shared state model* as a communication model in this paper, where each vertex can directly read the states of its neighbors.

The variables that are maintained by a vertex denote a vertex state. A vertex may take actions during the execution of the system. An action is simply a function that is executed in an atomic manner by the vertex. The actions executed by each vertex are described by a finite set of guarded actions of the form $\langle\text{guard}\rangle \longrightarrow \langle\text{statement}\rangle$. Each guard of vertex 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 vertices. We define Γ to be the set of all possible configurations of a distributed system g. For a vertex set $R \subseteq V$ and two configurations γ and γ', we denote $\gamma \overset{R}{\mapsto} \gamma'$ when γ changes to γ' by executing an action of each vertex in R simultaneously. Notice that γ and γ' can be different only in the states of vertices in R. For completeness of execution semantics, we should clarify the configuration resulting from simultaneous actions of neighboring vertices. The action of a vertex depends only on its state at γ and the states of its neighbors at γ, and the result of the action reflects on the state of the vertex at γ'.

We say that a vertex is *enabled* in a configuration γ if the guard of at least one of its actions is evaluated as true in γ.

A *schedule* of a distributed system is an infinite sequence of vertex sets. An infinite sequence of pairs of configurations $\sigma = (\gamma_0, \gamma_1), (\gamma_1, \gamma_2) \dots (\gamma_i, \gamma_{i+1}) \dots$ is called an *execution* from an initial configuration γ_0 by a schedule $Q = R^1, R^2, \dots$ (where R^i is a non-empty subset of enabled vertices in γ_{i-1} for each i $(i \geq 1)$) if σ satisfies $\gamma_{i-1} \overset{R^i}{\mapsto} \gamma_i$ for each i $(i \geq 1)$. Vertex actions are executed atomically, and we distinguish some properties on the scheduler (or daemon). A *distributed daemon* schedules the actions of vertices such that any subset of vertices can simultaneously execute their actions. We say that the daemon is *central* if it schedules an action of only one vertex at any step. The set of all possible executions from $\gamma_0 \in \Gamma$ is denoted by Σ_{γ_0}. The set of all possible executions is denoted by Σ, that is, $\Sigma = \bigcup_{\gamma \in \Gamma} \Sigma_\gamma$. We consider *asynchronous* distributed systems but we may add some of the following assumptions on schedules: any schedule is weakly fair (that is, it is impossible for any vertex to be infinitely and continuously enabled without executing its action in an execution), any schedule is strongly fair (that is, it is impossible for any vertex to be infinitely often enabled without executing its action in an execution), or/and k-bounded (that is, it is impossible for any vertex to execute more than k actions between two consecutive action executions of any other vertex).

In this paper, we consider (permanent) *Byzantine faults*: a Byzantine vertex (*i.e.* a Byzantine-faulty vertex) can exhibit arbitrary behavior independently from its actions. If v is a Byzantine vertex, v can repeatedly change its variables arbitrarily. For a given execution, the number of faulty vertices is arbitrary but we assume that the root vertex is never faulty. Indeed, without this assumption, any spanning tree construction problem is clearly impossible to solve since a Byzantine root may act as a non-root vertex leading correct vertices to construct a spanning tree in a distributed system without any distinguished vertex.

3 Containing Byzantine Faults in Stabilization

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* (for short, specification), $spec(v)$, for each vertex v: a configuration is a desired one (with a solution) if every vertex 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 vertices whose variables appear in $spec(v)$. The variables appearing in the specification are called *output variables* (for short, *O-variables*). In what follows, we consider a static problem defined by specification $spec(v)$.

A *self-stabilizing protocol* [12] is a protocol that eventually reaches a *legitimate configuration*, where $spec(v)$ holds at every vertex v, regardless of the initial configuration. Once it reaches a legitimate configuration, each vertex 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 vertex correctly executes its action from the configuration, *i.e.*, we do not consider existence of permanently faulty vertices.

3.1 Strict Stabilization

When (permanent) Byzantine vertices exist, Byzantine vertices may not satisfy $spec(v)$. In addition, correct vertices near the Byzantine vertices can be influenced and may be unable to satisfy $spec(v)$. Nesterenko and Arora [33] define a *strictly stabilizing protocol* as a self-stabilizing protocol resilient to unbounded number of Byzantine vertices.

Following definitions are with respect to a specification predicate *spec*. Given an integer c, a c-correct vertex is a vertex defined as follows.

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

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

The parameter c of Definition 2 refers to the containment radius defined in [33]. The parameter f refers explicitly to the number of Byzantine vertices, while [33] dealt with unbounded number of Byzantine faults (that is $f \in \{0 \ldots n\}$).

Definition 3 (Strict stabilization) A distributed protocol π is (c, f)-strictly stabilizing for specification *spec* if, starting from any arbitrary configuration, every execution involving at most f Byzantine vertices contains a configuration that is (c, f)-contained for *spec*.

An important limitation of the model of [33] is the notion of r-*restrictive* specifications. Intuitively, a specification is r-restrictive if it prevents combinations of states that belong to two vertices u and v that are at least r hops away. An important consequence related to Byzantine tolerance is that the containment radius of strictly-stabilizing protocols solving those specifications is at least r. For some (global) problems (like spanning tree construction), r cannot be bounded by a constant. As a consequence, we can show that there exists no $(c, 1)$-strictly stabilizing protocol for such a problem for any (finite) integer c.

3.2 Strong Stabilization

To circumvent such impossibility results of strict stabilization, we previously defined a weaker notion [20]. Here, the containment radius requirement is relaxed, *i.e.* there may exist vertices outside the containment radius that invalidate the specification predicate, due to Byzantine actions. However, the impact of Byzantine triggered actions is limited in time: the set of Byzantine vertices may only impact vertices outside the containment radius a bounded number of times, even if Byzantine vertices execute an infinite number of actions.

In the following of this section, we present the formal definition of strong stabilization for a specification predicate *spec*. From the states of c-correct vertices (see Definition 1), c-legitimate configurations and c-stable configurations are defined as follows.

Definition 4 (c-legitimate configuration) A configuration γ is c-legitimate for *spec* if every c-correct vertex v satisfies $spec(v)$.

Definition 5 (c-stable configuration) A configuration γ is c-stable if every c-correct vertex never changes the values of its O-variables as long as Byzantine vertices make no action.

Roughly speaking, the aim of self-stabilization is to guarantee that a distributed protocol eventually reaches a c-legitimate and c-stable configuration. However, a self-stabilizing system can be disturbed by Byzantine vertices after reaching a c-legitimate and c-stable configuration. The c-disruption represents the period where c-correct vertices are disturbed by Byzantine vertices and is defined as follows.

Definition 6 (c-disruption) A portion of execution $\delta = (\gamma_0, \gamma_1) \dots (\gamma_{t-1}, \gamma_t)$ $(t > 1)$ is a c-disruption if and only if the following holds:

1. δ is finite;
2. δ contains at least one action of a c-correct vertex that changes the value of an O-variable;
3. γ_0 is c-legitimate for *spec* and c-stable; and
4. γ_t is the first configuration after γ_0 such that γ_t is c-legitimate for *spec* and c-stable.

Now we can define a self-stabilizing distributed protocol such that Byzantine vertices may only impact vertices outside the containment radius a bounded number of times, even if Byzantine vertices execute an infinite number of actions.

Definition 7 ((t, k, c, f)-time contained configuration) A configuration γ_0 is (t, k, c, f)-time contained for *spec* if given at most f Byzantine vertices, the following properties are satisfied:

1. γ_0 is c-legitimate for *spec* and c-stable;
2. every execution starting from γ_0 contains a c-legitimate configuration for *spec* after which the values of all the O-variables of c-correct vertices remain unchanged (even when Byzantine vertices make actions repeatedly and forever),
3. every execution starting from γ_0 contains at most t c-disruptions, and
4. every execution starting from γ_0 contains at most k actions of changing the values of O-variables for each c-correct vertex.

Definition 8 (Strong stabilization) A distributed protocol π is (t, c, f)-strongly stabilizing for *spec* if and only if there exists $k \in \mathbb{N}$ such that every execution starting from an arbitrary configuration and involving at most f Byzantine vertices contains a (t, k, c, f)-time contained configuration for *spec* 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)-vertex disruption times of π.

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

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

3.3 Topology-Aware Stabilization

We defined another weaker notion than the strict stabilization: the topology-aware strict stabilization [18] (denoted by TA strict stabilization for short). Here, the requirement to the containment radius is relaxed, *i.e.* the set of vertices that may be disturbed by Byzantine ones is not reduced to the union of c-neighborhood of Byzantine vertices (*i.e.* the set of vertices at distance at most c from a Byzantine vertex) but can be defined depending on the communication graph and Byzantine vertices location.

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

Definition 9 (C_B-correct vertex) A vertex is C_B-correct if it is a correct vertex (*i.e.* not Byzantine) that does not belong to C_B.

Definition 10 (C_B-legitimate configuration) A configuration γ is C_B-legitimate for *spec* if every C_B-correct vertex v is legitimate for *spec* (*i.e.* if *spec*(v) holds).

Definition 11 ((C_B, f)-topology-aware containment) A configuration γ_0 is (C_B, f)-topology-aware contained for specification *spec* if, given at most f Byzantine vertices, in any execution $\sigma = (\gamma_0, \gamma_1)\ldots$, every configuration is C_B-legitimate and every C_B-correct vertex never changes its O-variables.

The parameter C_B of Definition 11 refers to the containment area. Any vertex that belongs to this set may be infinitely often disturbed by Byzantine vertices. The parameter f refers explicitly to the number of Byzantine vertices.

Definition 12 (Topology-aware strict stabilization) A distributed protocol is (C_B, f)-topology-aware strictly stabilizing for specification *spec* if, given at most f Byzantine vertices, any execution contains a configuration that is (C_B, f)-topology-aware contained for *spec*.

Note that, if B denotes the set of Byzantine vertices and $C_B = \Big\{v \in V | \min_{b\in B}\{dist(g, v, b)\} \leq c\Big\}$ then a (C_B, f)-topology-aware strictly stabilizing distributed protocol is a (c, f)-strictly stabilizing distributed protocol since C_B is then equals to the union of the c-neighborhood of Byzantine vertices. Then, the concept of the 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 vertices). The parameter C_B is introduced to quantify the strength of fault containment, we do not require each vertex to know the actual definition of the function.

Similarly to topology-aware strict stabilization, we can weaken the notion of strong stabilization using the notion of containment area. We present in the following the formal definition of this concept.

Definition 13 (C_B-stable configuration) A configuration γ is C_B-stable if every C_B-correct vertex never changes the values of its O-variables as long as Byzantine vertices make no action.

Definition 14 (C_B-TA disruption) A portion of execution $\delta = (\gamma_0, \gamma_1) \dots (\gamma_{t-1}, \gamma_t)$ $(t > 1)$ is a C_B-TA-disruption if and only if the followings hold:

1. δ is finite;
2. δ contains at least one action of a C_B-correct vertex that changes the value of a O-variable;
3. γ_0 is C_B-legitimate for *spec* and C_B-stable; and
4. γ_t is the first configuration after γ_0 such that γ_t is C_B-legitimate for *spec* and C_B-stable.

Definition 15 ((t, k, C_B, f)-TA time contained configuration) A configuration γ_0 is (t, k, C_B, f)-TA time contained for *spec* if given at most f Byzantine vertices, the following properties are satisfied:

1. γ_0 is C_B-legitimate for *spec* and C_B-stable;
2. every execution starting from γ_0 contains a C_B-legitimate configuration for *spec* after which the values of all the O-variables of C_B-correct vertices remain unchanged (even when Byzantine vertices make actions repeatedly and forever);
3. every execution starting from γ_0 contains at most t C_B-TA-disruptions; and
4. every execution starting from γ_0 contains at most k actions of changing the values of O-variables for each C_B-correct vertex.

Definition 16 (Topology-aware strong stabilization) A distributed protocol π is (t, C_B, f)-TA strongly stabilizing for *spec* if and only if there exists $k \in \mathbb{N}$ such that every execution starting from an arbitrary configuration and involving at most f Byzantine vertices contains a (t, k, C_B, f)-TA time contained configuration for *spec* that is reached after at most ℓ rounds.Parameters ℓ and k are respectively the (t, C_B, f)-stabilization and the (t, C_B, f)-vertex disruption times of π.

4 Maximum Metric Tree Construction

In this section, we formally define maximum (routing) metric trees using formalism introduced by [27]. Informally, the goal of a distributed routing distributed protocol is to construct a tree that simultaneously maximizes the metric values of all of the vertices with respect to some total ordering $\prec$. Then, we can specify the problem considered in this paper.

Maximum metric tree First, we recall definitions and notations of [27] and state the main result about characterization of maximizable metrics (that is, metrics such that there always exists a tree maximizing the metric of each vertex).

Definition 17 (Routing metric) A routing metric (or just metric) $\mathcal{M}$ is a five-tuple $\mathcal{M} = (M, W, met, mr, \prec)$ 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 :
   $$\textit{if } m \prec m' \textit{ and } m' \prec m'' \textit{ then } m \prec m''$$
   (c) totality: $m \prec m'$ or $m' \prec m$ or $m = m'$.

Moreover, 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 below the definition of three classical metrics with this model:

– the shortest path metric ($\mathcal{SP}$), where the distance from any vertex to the root is minimized. Edge weights are natural numbers and the metric operator is the sum.
– the flow metric ($\mathcal{F}$), where each vertex chooses the path of maximal flow (*i.e.* the weight of the edge of minimum weight of the path) to the root. Edge weights are natural numbers and the metric operator is the minimum function.
– the reliability metric ($\mathcal{R}$), where each vertex chooses the path of maximal reliability (*i.e.* the product of edge weights of the path) to the root. Edge weights are real numbers (between 0 and 1) and the metric operator is the product.

Note also that we can model 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 denote this metric by $\mathcal{BFS}$ in the following).

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

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

Definition 18 (Assigned metric) An assigned metric over a communication graph g 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 g a weight in W.

Let a rooted path (from v) be an elementary path from a vertex 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 communication graph g.

Definition 19 (Metric of a rooted path) The metric of a rooted path in g is the prefix sum of *met* over the edge weights in the path and *mr*.

For example, if a rooted path p in g 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 $\forall i \in \{1, \ldots, k\}, m_i = met(m_{i-1}, wf(\{v_i, v_{i-1}\}))$ and $m_0 = mr$.

Definition 20 (Maximum metric path) A rooted path p from v in g 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 g, 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 vertex) The maximum metric of a vertex $v \neq r$ (or simply metric value of v) in g is defined by the metric of a maximum metric path from v. The maximum metric of r is *mr*.

Definition 22 (Maximum metric spanning tree) A spanning tree t of g is a maximum metric spanning tree with respect to an assigned metric over g if and only if every rooted path in t is a maximum metric path in g with respect to the assigned metric.

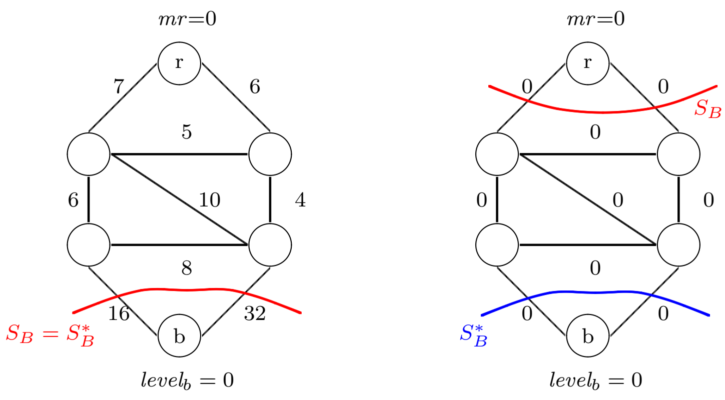

Fig. 1 Examples of containment areas for $\mathcal{SP}$.

The goal of the work of [27] is the study of metrics that always allow the construction of a maximum metric spanning tree. The definition follows.

Definition 23 (Maximizable metric) A metric is maximizable if and only if for any assignment of this metric over any communication graph g, there is a maximum metric spanning tree for g with respect to the assigned metric.

An interesting result about maximizable metrics due to [27] provides a full characterization of maximizable metrics as follows. First, they define two classes of metrics. A metric is bounded if and only if the application of the metric function to any metric value does not increase it (for any edge weight) whereas a metric is monotonic if and only if the metric function preserves the order $\prec$ on metric values. Formal definitions follow.

Definition 24 (Boundedness) A metric $(M, W, met, mr, \prec)$ is bounded if and only if: $\forall m \in M, \forall w \in W, met(m, w) \prec m \text{ or } met(m, w) = m$

Definition 25 (Monotonicity) A metric $(M, W, met, mr, \prec)$ is monotonic if and only if:

$$\forall (m, m') \in M^2, \forall w \in W, m \prec m' \Rightarrow \\ (met(m, w) \prec met(m', w) \text{ or } met(m, w) = met(m', w))$$

Then, [27] proves that a metric is maximizable if and only if it belongs to the intersection of these two classes of metrics.

Theorem 1 (Characterization of maximizable metrics [27]) *A metric is maximizable if and only if this metric is bounded and monotonic.*

Specification Given a maximizable metric $\mathcal{M} = (M, W, mr, met, \prec)$, the aim of this paper is to study the construction of a maximum metric spanning tree with respect to $\mathcal{M}$ rooted to a pre-defined vertex r (called the root) in simultaneous presence of transient and Byzantine faults. Note that we must assume that the root vertex is never a Byzantine one. It is obvious that these Byzantine vertices may disturb some correct vertices. Therefore, we relax the problem in the following way: we want to construct a maximum metric spanning forest with respect to $\mathcal{M}$. The root of any tree of this spanning forest must be either the real root or a Byzantine vertex.

Each vertex v has three O-variables: a pointer to its parent in its tree ($prnt_v \in N_v \cup \{\perp\}$), a variable that stores its current metric value ($level_v \in M$) and an integer that stores a distance ($dist_v \in \mathbb{N}$). 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 communication graph g and two vertices u and v, we denote by $max\_met(g, u, v)$ the maximum metric of vertex u when v plays the role of the root of the communication graph (that is, when $level_v = mr$). If u and v are neighbors, we denote by $w_{u,v}$ the weight of the edge $\{u, v\}$ (that is, the value of $wf(\{u, v\})$).

We define the legal distance of a vertex v with respect to one of its neighbors u (denoted $legal\_dist(v, u)$) in the following way:

$$legal\_dist(v,u) = \begin{cases} dist_u + 1 \ \ if\ level_v = level_u \\ 0 \ \ otherwise \end{cases}$$

Definition 26 ($\mathcal{M}$-path) Given an assigned metric $\mathcal{M} = (M, W, mr, met, \prec, wf)$ over a communication graph g, a path $(v_0, \ldots, v_k)$ $(k \geq 1)$ of g is a $\mathcal{M}$-path if and only if:

1. $prnt_{v_0} = \perp$, $level_{v_0} = mr$, $dist_{v_0} = 0$, and $v_0 \in B \cup \{r\}$;
2. $\forall i \in \{1, \ldots, k\}$, $prnt_{v_i} = v_{i-1}$, $level_{v_i} = met(level_{v_{i-1}}, w_{v_i,v_{i-1}})$, and $dist_{v_i} = legal\_dist(v_i, v_{i-1})$;
3. $\forall i \in \{1, \ldots, k\}$,
$met(level_{v_{i-1}}, w_{v_i,v_{i-1}}) = \max_{\prec, u \in N_v}\{met(level_u, w_{v_i,u})\}$
4. $level_{v_k} = max\_met(g, v_k, v_0)$.

We can now specify the problem of the maximum metric spanning tree construction.

Specification 1 ($spec_{MMT}$) *The specification predicate $spec_{MMT}(v)$ of the maximum metric tree construction with respect to a maximizable metric $\mathcal{M}$ for vertex v follows:*

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

In other words, this specification states that every correct vertex belongs to a maximum metric spanning tree rooted either on the real root of the distributed system or to a Byzantine vertex that exhibits exactly the same state as the correct root.

At first glance, it may be surprising to define our problem in this way since correct vertices cannot know if they belong to a legitimate tree (that is, rooted at the real root of the distributed system) or a “fake” one (that is, rooted at a Byzantine vertex). The motivation behind this specification is the following. Recall that it is impossible for a correct vertex to detect whether one of its neighbors is Byzantine or not. Then, by symmetry, it is impossible to guarantee that correct neighbors of a Byzantine vertex (that acts as the correct root) does not begin to construct a spanning tree rooted at this Byzantine vertex if we want that correct neighbors of the correct root eventually join a spanning tree rooted at it. By repeating this argument, we can conclude that we have to accept the relaxation of the specification to a spanning forest construction since it is impossible to provide stronger guarantees for vertices that belong to a “fake” spanning tree.

Set of used metric values We introduce here a new definition that is used in the following of this paper. Given an assigned metric over a communication graph, a used metric value is a metric value that is the metric of a rooted path of the communication graph (either rooted at the root or at a Byzantine vertex). More formally,

Definition 27 (Set of used metric values) Given an assigned metric $\mathcal{AM} = (M, W, met, mr, \prec, wf)$ over a communication graph g, the set of used metric values of $\mathcal{AM}$ is defined as:

$$M(g) = \{m \in M | \exists v \in V, (max\_met(g, v, r) = m) \vee (\exists b \in B, max\_met(g, v, b) = m)\}$$

Note that for any communication graph g and any assigned metric $\mathcal{AM} = (M, W, met, mr, \prec, wf)$ over g, we have: $M(g) \subseteq M$.

5 Topology-Aware Stabilizing Solution

This section aims to present a distributed protocol for the maximum metric spanning tree construction with respect to any maximizable metric in systems simultaneously subject to transient and Byzantine faults. Section 5.1 presents the distributed protocol while Sections 5.2 and 5.3 prove respectively its TA strict stabilization and its TA strong stabilization. Note that we prove the optimality of containment areas of our protocol in Section 6.

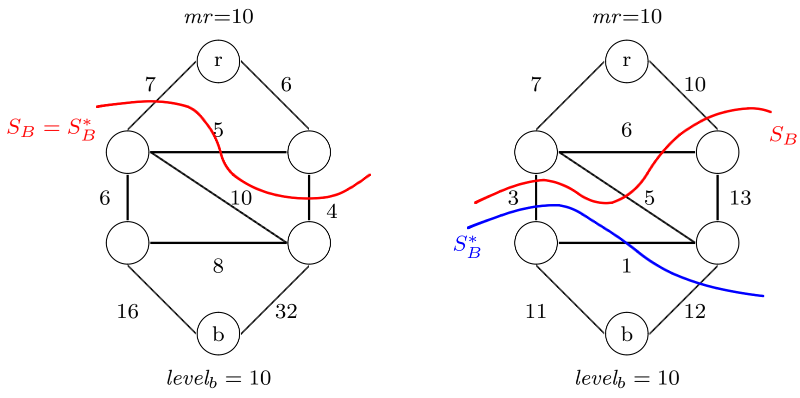

Fig. 2 Examples of containment areas for $\mathcal{F}$.

Containment areas We define now the containment areas performed by our protocol. First, the containment area for topology-aware strict stabilization is the following (see Theorem 2):

$$S_B = \left\{ v \in V \setminus B \middle| max\_met(g,v,r) \preceq \max_{b \in B}{}_{\prec} \{max\_met(g,v,b)\} \right\} \setminus \{r\}$$

Intuitively, S_B gathers the set of correct vertices that are closer or at equals distance (according to $\mathcal{M}$) to a Byzantine vertex than the root. Then, we can define the containment area for topology-aware strong stabilization (see Theorem 3):

$$S_B^* = \left\{ v \in V \setminus B \middle| max\_met(g,v,r) \prec \max_{b \in B}{}_{\prec} \{max\_met(g,v,b)\} \right\}$$

Intuitively, S_B^* gathers the set of correct vertices that are strictly closer (according to $\mathcal{M}$) to a Byzantine vertex than the root. Note that we assume for the sake of clarity that $V \setminus S_B^*$ induces a connected communication subgraph. If it is not the case, then S_B^* is extended to include all vertices belonging to connected communication subgraphs of $V \setminus S_B^*$ that not include r. Figures from 1 to 3 provide some examples of containment areas S_B and S_B^* with respect to maximizable metrics previously presented.

5.1 Distributed Protocol

A self-stabilizing distributed protocol for the maximum metric spanning tree construction with respect to any maximizable metric has been proposed by [26]. In this distributed protocol, any vertex try to maximize its *level* variable in the tree by choosing as its parent (*prnt* variable) the neighbor that provides the best metric value. Using this strategy, the arbitrary initial configuration

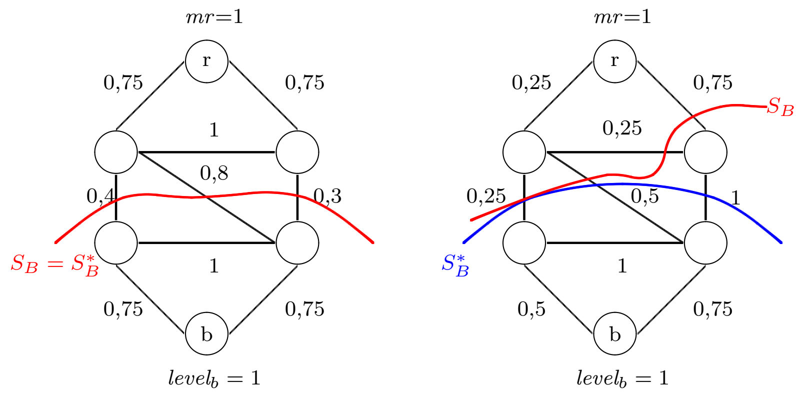

Fig. 3 Examples of containment areas for $\mathcal{R}$.

may lead to the formation of cycles. The key idea of this distributed protocol is to use the *dist* variable (upper bounded by a given constant D) to detect and break cycles of vertices that has the same (incorrect) maximum metric. The choice of the constant D is obviously capital for the self-stabilization of the distributed protocol. Gouda and Schneider proved that their distributed protocol is self-stabilizing if D is an upper bound on the length of the longest path of the desired tree.

A natural way to provide a topology-aware stabilizing solution to the maximum metric spanning tree construction is then to adapt the idea of round robin choice over neighbors presented in [20] to the distributed protocol of [26]. It is possible to prove that this strategy is sufficient to perform the $(S_B, n-1)$-TA strict stabilization. Unfortunately, this strategy is not suitable for topology-aware strong stabilization.

Indeed, an execution of the distributed protocol of [26] may be subject to an infinite number of S_B^*-disruptions due to the following fact: a Byzantine vertex can independently lie about its *level* and its *dist* variable. For example, a Byzantine vertex can provide a *level* equals to mr and a *dist* arbitrarily large. In this way, it may lead a correct vertex of $S_B \setminus S_B^*$ to have a *dist* variable equals to $D-1$ such that no other correct vertex 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 equals to D). Then, this vertex may always prevent some of its neighbors to join a $\mathcal{M}$-path connected to the root and hence allow another Byzantine vertex to perform an infinite number of disruptions.

In contrast, we want to provide a distributed protocol that is simultaneously $(S_B, n-1)$-TA strictly stabilizing and $(t, S_B^*, n-1)$-TA strongly stabilizing for maximum metric spanning tree construction. To perform this goal, our distributed protocol needs a supplementary assumption on the assignement of the considered maximizable metric over the communication graph.

We assume that we always have $|M(g)| \geq 2$ (the necessity of this assumption is explained below). Nevertheless, note that the contrary case ($|M(g)| = 1$) is possible if and only if the assigned maximizable metric is equivalent to $\mathcal{NC}$. As the distributed protocol presented in [20] 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 $|M(g)| = 1$ (since this implies that $S_B^* = \emptyset$). In this way, this assumption does not weaken the possibility result.

We already said that the distributed protocol of [26] is not suitable for our purposes but our distributed protocol borrows fundamental strategy from it. Indeed, we use almost the same ideas with the following two exceptions: (i) we ensure a fair selection along the set of neighbors with a round-robin order for the *prnt* variable and (ii) we modify the management of the *dist* variable to avoid executions exhibiting an infinite number of S_B^*-disruptions.

In order to contain the effect of Byzantine vertices on *dist* variables, each vertex 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 vertex modifying its *dist* variable can only affect correct vertices that have the same *level*. Consequently, in the case where $|M(g)| \geq 2$, we are ensured that correct vertices of $S_B \setminus S_B^*$ cannot keep a *dist* variable equals or greater than $D - 1$ infinitely. Hence, a correct vertex 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 $|M(g)| \geq 2$ is essential to perform the topology-aware strong stabilization. Indeed, in the case where $|M(g)| = 1$, Byzantine vertices can play exactly the scenario described above (in this case, our distributed protocol is equivalent to the one of [26]).

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

Our protocol, $\mathcal{SSMMT}$ (for $\mathcal{S}$trictly/strongly $\mathcal{S}$tabilizing $\mathcal{M}$aximum $\mathcal{M}$etric $\mathcal{T}$ree), is formally described in Protocol 1.

5.2 Proof of Topology-Aware Strict Stabilization

This section is devoted to the proof of the $(S_B, n-1)$-TA strict stabilization of $\mathcal{SSMMT}$ under the distributed weakly fair daemon (see Theorem 2). This proof is an induction proof with respect to the maximal metric of each correct vertex but the main difficulty comes from the fact that several vertices along a path could have the same maximal metric with respect to the root (or to a Byzantine vertex).

Protocol 1 $\mathcal{SSMMT}$: $(S_B, n-1)$-TA strictly and $(t, S_B^*, n-1)$-TA strongly stabilizing distributed protocol for $spec_{MMT}$ for vertex v.

```
Constants:
    $N_v$: set of neighbors of $v$ (ordered in a round robin fashion)
    $D$: upper bound of the number of vertices in an elementary path
Variables:
    $prnt_v \in \begin{cases} \{\bot\} \text{ if } v = r \\ N_v \text{ if } v \neq r \end{cases}$ : parent of $v$ in the current tree
    $level_v \in M$: metric of $v$ in the current tree
    $dist_v \in \{0, \dots, D\}$: distance of $v$ in the current tree
Functions:
    $next_v$: for any subset $A \subseteq N_v$, $next_v(A)$ returns the first element of $A$ that is bigger
    than $prnt_v$ in a round-robin fashion and an arbitrary element of $A$ if $prnt_v = \bot$
    $current\_dist_v() = \begin{cases} 0 \text{ if } level_{prnt_v} \neq level_v \\ min\{dist_{prnt_v}+1, D\} \text{ if } level_{prnt_v} = level_v \end{cases}$
Rules:
    $(\boldsymbol{R_r})$ :: $(v = r) \wedge ((level_v \neq mr) \vee (dist_v \neq 0))$
        $\longrightarrow level_v := mr;\ dist_v := 0$
    $(\boldsymbol{R_1})$ :: $(v \neq r) \wedge \big[(dist_v < current\_dist_v()) \vee (level_v \neq met(level_{prnt_v}, w_{v,prnt_v}))\big]$
        $\longrightarrow level_v := met(level_{prnt_v}, w_{v,prnt_v});\ dist_v := current\_dist_v()$
    $(\boldsymbol{R_2})$ :: $(v \neq r) \wedge \big[(dist_v = D) \vee (dist_v > current\_dist_v())\big] \wedge (\exists u \in N_v, dist_u < D-1)$
        $\longrightarrow prnt_v := next_v\big(\{u \in N_v | dist_u < D-1\}\big);$
            $level_v := met(level_{prnt_v}, w_{v,prnt_v});\ dist_v := current\_dist_v()$
    $(\boldsymbol{R_3})$ :: $(v \neq r) \wedge (\exists u \in N_v, (dist_u < D-1) \wedge (level_v \prec met(level_u, w_{u,v})))$
        $\longrightarrow prnt_v := next_v\big(\{u \in N_v | (dist_u < D-1) \wedge$
                $(met(level_u, w_{u,v}) = \max_{\prec, q \in N_v \wedge dist_q < D-1} \{met(level_q, w_{q,v})\})\}\big);$
            $level_v := met(level_{prnt_v}, w_{prnt_v,v});\ dist_v := current\_dist_v()$
```

We must first prove some lemmas. From now, we consider that $\mathcal{M} = (M, W, mr, met, \prec)$ is a maximizable metric assigned over our communication graph $g = (V, E)$ by the weight function *wf*. First, we provide a useful property about $\mathcal{M}$.

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

$$\forall u \in N_v, met\Big(\max_{\prec\, p \in B \cup \{r\}} \{max\_met(g, u, p)\}, w_{u,v}\Big) \preceq \max_{\prec\, p \in B \cup \{r\}} \{max\_met(g, v, p)\}$$

Proof By contradiction, assume that there exists a neighbor u of a vertex v such that:

$$\max_{\prec\, p \in B \cup \{r\}} \{max\_met(g, v, p)\} \prec met\Big(\max_{\prec\, p \in B \cup \{r\}} \{max\_met(g, u, p)\}, w_{u,v}\Big)$$

Let $q \in B \cup \{r\}$ be one of the vertices such that $\max_{\prec\, p \in B \cup \{r\}} \{max\_met(g, u, p)\} = max\_met(g, u, q)$. Then, the construction of q allows us to deduce that:

$$\max_{\prec\, p \in B \cup \{r\}} \{max\_met(g, v, p)\} \prec met(max\_met(g, u, q), w_{u,v})$$

Since we have $met(max\_met(g,u,q), w_{u,v}) \preceq max\_met(g,\ v,q)$, we conclude that:

$$\underset{p \in B \cup \{r\}}{max_{\prec}} \{max\_met(g,v,p)\} \prec max\_met(g,v,q)$$

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

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

$$IM_m(\gamma) \equiv \forall v \in V, level_v \preceq max_{\prec}\Big\{m,\ \underset{u \in B \cup \{r\}}{max_{\prec}} \{max\_met(g,v,u)\}\Big\}$$

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

Proof Let m be a metric value ($m \in M$). Let $\gamma \in \Gamma$ be a configuration such that $IM_m(\gamma) = true$ and $\gamma' \in \Gamma$ be a configuration such that (γ, γ') is an action of $\mathcal{SSMMT}$.

If the root vertex r is activated during action (γ, γ') (respectively a Byzantine vertex b is activated during action (γ, γ')), then we have $level_r = mr$ (respectively $level_b \preceq mr$) in γ' by construction of $\boldsymbol{(R_r)}$ (respectively by definition of $level_b$). Hence, we have:

$$level_r \preceq max_{\prec}\Big\{m,\ \underset{u \in B \cup \{r\}}{max_{\prec}} \{max\_met(g,\ r,u)\}\Big\} = mr$$
$$level_b \preceq max_{\prec}\Big\{m,\ \underset{u \in B \cup \{r\}}{max_{\prec}} \{max\_met(g,b,u)\}\Big\} = mr$$

If a correct vertex v is activated during action (γ, γ') with $v \neq r$, then there exists a neighbor p of v such that $level_p \preceq max_{\prec}\Big\{m,\ \underset{u \in B \cup \{r\}}{max_{\prec}} \{max\_met(g,p,u)\}\Big\}$ in γ (since $IM_m(\gamma) = true$) and $prnt_v = p$ and $level_v = met(level_p,\ w_{v,p})$ in γ' (since v is activated during this action).

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

$$met\Big(\underset{u \in B \cup \{r\}}{max_{\prec}} \{max\_met(g,p,u)\}, w_{v,p}\Big) \preceq \underset{u \in B \cup \{r\}}{max_{\prec}} \{max\_met(g,v,u)\}$$

Consequently, we obtain that $level_v = met(level_p,\ w_{v,p})$ in γ'. The monotonicity of $\mathcal{M}$ allows us to deduce

$$level_v \preceq met\Big(max_{\prec}\{m,\ \underset{u \in B \cup \{r\}}{max_{\prec}} \{max\_met(g,p,u)\}\}, w_{v,p}\Big)$$

Then,

$$level_v \preceq max_{\prec}\Big\{met(m, w_{v,p}), met\big(\underset{u \in B \cup \{r\}}{max_{\prec}} \{max\_met(g,p,u)\}, w_{v,p}\big)\Big\}$$

As $met(m, w_{v,p}) \preceq m$, we can conclude that:

$$level_v \preceq max_{\prec}\Big\{m, \max_{\prec\, u\in B\cup\{r\}}\{max\_met(g,v,u)\}\Big\}$$

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

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

For any $m_i \in M(g)$, we introduce the following set of notations:

$$\begin{aligned}
P_{m_i} &= \big\{v \in V \setminus S_B \big| max\_met(g,v,r) = m_i\big\}\\
V_{m_i} &= \bigcup_{j=0}^{i} P_{m_j}\\
I_{m_i} &= \big\{v \in V \big| \max_{\prec\, u\in B\cup\{r\}}\{max\_met(g,v,u)\} \prec m_i\big\}\\
\mathcal{LC}_{m_i} &= \big\{\gamma \in \Gamma \big| (\forall v \in V_{m_i}, spec_{MMT}(v)) \wedge (IM_{m_i}(\gamma))\big\}
\end{aligned}$$

Then, we define the following set of configurations : $\mathcal{LC}_{MMT} = \mathcal{LC}_{m_k}$

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

Proof Let m_i be a metric value from $M(g)$ and γ be a configuration of $\mathcal{LC}_{m_i}$. By construction, any vertex $v \in V_{m_i}$ satisfies $spec_{MMT}(v)$ in γ.

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

In the same way, any vertex $v \in V_{m_i}$ satisfies: $prnt_v \in N_v$, $level_v = met(level_{prnt_v}, w_{prnt_v,v})$, $dist_v = legal\_dist(v, prnt_v)$, and $level_v = \max_{\prec\, u\in N_v}\{met(level_u, w_{u,v})\}$. Note that, as $v \in V_{m_i}$ and $spec_{MMT}(v)$ holds in γ, we have: $level_v = max\_met(g, v, r) = \max_{\prec\, p\in B\cup\{r\}}\{max\_met(g,v,p)\}$ and $dist_v \leq D-1$ by construction of D. Hence, vertex v is not enabled by $\mathcal{SSMMT}$ in γ.

Assume that there exists a vertex $v \in V_{m_i}$ that is activated during an action (γ', γ'') in an execution starting from γ (without loss of generality, assume that v is the first vertex of $v \in V_{m_i}$ that is activated in this execution). Then, we know that $v \neq r$. This activation implies that a neighbor $u \notin V_{m_i}$ (since v is the first vertex of V_{m_i} to be activated) of v modified its *level* variable to a metric value $m \in M$ such that $level_v \prec met(m, w_{u,v})$ in γ' (note that O-variables of v and of $prnt_v$ remain consistent since v is the first vertex to be activated in this execution).

Hence, we have $level_v = \max_{\prec\, p\in B\cup\{r\}}\{max\_met(g,v,p)\} = max\_met(g,v,r)$ (since $spec_{MMT}(v)$ holds), $level_v \prec met(m, w_{u,v})$ (since u causes an action of

v), and $m_i \preceq \mathit{level}_v$ (since $v \in V_{m_i}$ and $\mathit{level}_v = \mathit{max\_met}(g, v, r)$). Moreover, the closure of IM_{m_i} (established in Lemma 2) ensures us that:

$$m = \mathit{level}_u \preceq \mathit{max}_{\prec}\Big\{m_i, \underset{p \in B \cup \{r\}}{\mathit{max}_{\prec}} \{\mathit{max\_met}(g, u, p)\}\Big\}$$

Let us study the two following cases:

Case 1: We have :

$$\mathit{max}_{\prec}\Big\{m_i, \underset{p \in B \cup \{r\}}{\mathit{max}_{\prec}} \{\mathit{max\_met}(g, u, p)\}\Big\} = m_i$$

We have then $m \preceq m_i$. As the boundedness of $\mathcal{M}$ ensures that $\mathit{met}(m, w_{u,v}) \preceq m$, we can conclude that $\mathit{level}_v \prec \mathit{met}(m, w_{u,v}) \preceq m \preceq m_i \preceq \mathit{level}_v$, that is absurd.

Case 2: We have :

$$\mathit{max}_{\prec}\Big\{m_i, \underset{p \in B \cup \{r\}}{\mathit{max}_{\prec}} \{\mathit{max\_met}(g, u, p)\}\Big\} = \underset{p \in B \cup \{r\}}{\mathit{max}_{\prec}} \{\mathit{max\_met}(g, u, p)\}$$

We have then $m \preceq \underset{p \in B \cup \{r\}}{\mathit{max}_{\prec}} \{\mathit{max\_met}(g, u, p)\}$. By monotonicity of $\mathcal{M}$, we can deduce that $\mathit{met}(m, w_{u,v}) \preceq \mathit{met}(\underset{p \in B \cup \{r\}}{\mathit{max}_{\prec}} \{\mathit{max\_met}(g, u, p)\}, w_{u,v})$.

Consequently, we obtain that:

$$\underset{p \in B \cup \{r\}}{\mathit{max}_{\prec}} \{\mathit{max\_met}(g, v, p)\} \prec \mathit{met}(\underset{p \in B \cup \{r\}}{\mathit{max}_{\prec}} \{\mathit{max\_met}(g, u, p)\}, w_{u,v})$$

This is contradictory with the result of Lemma 1.

In conclusion, no vertex $v \in V_{m_i}$ is activated in any execution starting from γ and then always satisfies $\mathit{spec}_{MMT}(v)$. Then, the closure of IM_{m_i} (established in Lemma 2) concludes the proof. □

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

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

Lemma 5 *Starting from any configuration of Γ, any execution of $\mathcal{SSMMT}$ under the distributed weakly fair daemon reaches in a finite time a configuration of $\mathcal{LC}_{mr}$.*

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

If r does not satisfy $\mathit{spec}_{MMT}(r)$ in γ, then r is continuously enabled. Since the daemon is weakly fair, r is activated in a finite time and then r satisfies $\mathit{spec}_{MMT}(r)$ in a finite time. Denote by γ' the first configuration where $\mathit{spec}_{MMT}(r)$ holds. Note that r is never activated in any execution starting from γ'.

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

Otherwise ($P_{mr} \neq \{r\}$), observe that, for any configuration of an execution starting from γ', if all vertices of P_{mr} are not enabled, then any vertex v of P_{mr} satisfies $spec_{MMT}(v)$. Assume now that there exists an execution σ starting from γ' where some vertices of P_{mr} are infinitely often activated. By construction, at least one of these vertices (note it v) has a neighbor u that is activated only a finite number of time in σ (recall that P_{mr} induces a connected subsystem and that r is not activated in σ). After u takes its last action of σ, we can observe that $level_u = mr$ and $dist_u < D - 1$ (otherwise, u is activated in a finite time that contradicts its construction).

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

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

Lemma 6 *For any $m_i \in M(g)$ and for any configuration $\gamma \in \mathcal{LC}_{m_i}$, any execution of $\mathcal{SSMMT}$ starting from γ under the distributed weakly fair daemon reaches in a finite time a configuration such that:*

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

Proof Let m_i be an arbitrary metric value of $M(g)$ and γ_0 be an arbitrary configuration of $\mathcal{LC}_{m_i}$. Let $\sigma = (\gamma_0, \gamma_1) \ldots$ be an execution starting from γ_0.

Note that γ_0 satisfies IM_{m_i} by construction. Hence, we have $\forall v \in I_{m_i}$, $level_v \preceq m_i$. The closure of IM_{m_i} (proved in Lemma 2) ensures us that this property is satisfied in any configuration of σ.

If any vertex $v \in I_{m_i}$ satisfies $level_v \prec m_i$ in γ_0, then the result is obvious. Otherwise, we define the following variant function. For any configuration γ_j of σ, we denote by A_j the set of vertices v of I_{m_i} such that $level_v = m_i$ in γ_j. Then, we define $f(\gamma_j) = \underset{v \in A_j}{min}\{dist_v\}$. We then prove the result by showing that there exists an integer k such that $f(\gamma_k) = D$.

First, if a vertex v joins A_j (that is, $v \notin A_{j-1}$ but $v \in A_j$), then it takes a $dist$ value greater or equals to $f(\gamma_{j-1}) + 1$ by construction of $\mathcal{SSMMT}$. We can deduce that any vertex that joins A_j does not decrease f. Moreover, the construction of $\mathcal{SSMMT}$ implies that a vertex v such that $v \in A_j$ and $v \in A_{j+1}$ cannot decrease its $dist$ value in the action (γ_j, γ_{j+1}).

Then, consider for a given configuration γ_j a vertex $v \in A_j$ such that $dist_v = f(\gamma_j) < D$. We claim that v is enabled by $\mathcal{SSMMT}$ in γ_j and that the execution of the enabled rule either increases strictly $dist_v$ or removes v from A_{j+1}. To prove this claim, we distinguish the following cases:

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

The fact that $v \in I_{m_i}$, the boundedness of $\mathcal{M}$ and the closure of IM_{m_i} (established in Lemma 2) imply that $prnt_v \in A_j$ (and, hence that $level_{prnt_v} = m_i$). Then, by construction of $f(\gamma_j)$, we know that $dist_{prnt_v} \geq f(\gamma_j) = dist_v$. Hence, we have $dist_v < dist_{prnt_v} + 1$ in γ_j. Then, v is enabled by $\boldsymbol{(R_1)}$ in γ_j and $dist_v$ increases by at least 1 during the action (γ_j, γ_{j+1}) if this rule is executed.

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

Assume that v is activated by $\boldsymbol{(R_2)}$ or $\boldsymbol{(R_3)}$ during the action (γ_j, γ_{j+1}). If v does not belong to A_{j+1} (if $level_v \neq m_i$ in γ_{j+1}), the claim is satisfied. In the contrary case (v belongs to A_{j+1}), we know that $level_v = m_i$ in γ_{j+1}. The boundedness of $\mathcal{M}$ and the closure of IM_{m_i} (established in Lemma 2) imply that $level_{prnt_v} = m_i$ in γ_{j+1}. We can conclude that $dist_v$ increases by at least 1 during the action (γ_j, γ_{j+1}) since the new parent of v has a distance greater than $f(\gamma_j)$ by construction of A_{j+1}.

Otherwise, we know that the rule $\boldsymbol{(R_1)}$ is enabled for v in γ_j. If this rule is executed during the action (γ_j, γ_{j+1}), one of the two following sub cases appears.

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

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

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

Remind that the closure of IM_{m_i} (established in Lemma 2) implies then that $level_{prnt_v} = m_i$. By construction of $f(\gamma_j)$, we have $dist_{prnt_v} \geq f(\gamma_j)$ in γ_j. Then, we can see that $dist_v$ increases by at least 1 during the action (γ_j, γ_{j+1}).

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

As I_{m_i} is finite and the daemon is weakly fair, we can deduce that f increases in a finite time in any execution starting from γ_j. By repeating the argument at most D time, we can deduce that σ contains a configuration γ_k such that $f(\gamma_k) = D$, that shows the result. □

Lemma 7 *For any $m_i \in M(g)$ and for any configuration $\gamma \in \mathcal{LC}_{m_i}$ such that $\forall v \in I_{m_i}, level_v = m_i \Rightarrow dist_v = D$, any execution of $\mathcal{SSMMT}$ starting from γ under the distributed weakly fair daemon reaches in a finite time a configuration such that: $\forall v \in I_{m_i}, level_v \prec m_i$*

Proof Let $m_i \in M(g)$ be an arbitrary metric value and γ_0 be a configuration of $\mathcal{LC}_{m_i}$ such that $\forall v \in I_{m_i}, level_v = m_i \Rightarrow dist_v = D$. Let $\sigma = (\gamma_0, \gamma_1) \ldots$ be an arbitrary execution starting from γ_0.

For any configuration γ_j of σ, let us denote $A_j = \{v \in I_{m_i} | level_v = m_i\}$. By the closure of IM_{m_i} (that holds by definition in γ_0) established in Lemma 2, we obtain the result if there exists a configuration γ_j of σ such that $A_j = \emptyset$.

If there exist some vertices $v \in I_{m_i} \setminus A_0$ (and hence $level_v \prec m_i$) such that $prnt_v \in A_0$ and $met(level_{prnt_v}, w_{v,prnt_v}) = m_i$ in γ_0, then we can observe that these vertices are continuously enabled by $\boldsymbol{(R_1)}$. As the daemon is weakly fair,

v executes this rule in a finite time and then, $level_v = m_i$ and $dist_v = D$. In other words, v joins A_ℓ for a given integer ℓ. We can conclude that there exists an integer k such that the following property $\boldsymbol{(P)}$ holds: for any $v \in I_{m_i} \setminus A_0$, either $prnt_v \notin A_k$ or $met(level_{prnt_v}, w_{v,prnt_v}) \prec m_i$.

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

Case 1: v executes $\boldsymbol{(R_1)}$ during the action (γ_j, γ_{j+1}).
Note that the property $\boldsymbol{(P)}$ still holds in γ_j by the construction of j. Hence, we know that $prnt_v \notin A_j$ in γ_j. But in this case, we have: $level_{prnt_v} \prec m_i$. The boundedness of $\mathcal{M}$ implies that $level_v = met(level_{prnt_v}, w_{v,prnt_v}) \prec m_i$ in γ_{j+1} that contradicts the fact that $v \in A_{j+1}$.

Case 2: v executes either $\boldsymbol{(R_2)}$ or $\boldsymbol{(R_3)}$ during the action (γ_j, γ_{j+1}).
That implies that v chooses a new parent that has a distance smaller than $D-1$ in γ_j. This implies that this new parent does not belongs to A_j. Then, we have $level_{prnt_v} \prec m_i$. The boundedness of $\mathcal{M}$ implies that $level_v = met(level_{prnt_v}, w_{v,prnt_v}) \prec m_i$ in γ_{j+1} that contradicts the fact that $v \in A_{j+1}$.

In the two cases, our claim is satisfied. In other words, there exists a point of the execution (namely γ_k) afterwards the set A cannot grow (this implies that, if a vertex leaves the set A, it is a definitive leaving).

Assume now that there exists an action (γ_j, γ_{j+1}) (with $j \geq k$) such that a vertex $v \in A_j$ is activated. Observe that the closure of IM_{m_i} (established in Lemma 2) implies that v can not be activated by the rule $\boldsymbol{(R_3)}$. If v activates $\boldsymbol{(R_1)}$ during this action, then v modifies its *level* during this action (otherwise, we have a contradiction with the fact that $level_{prnt_v} = m_i \Rightarrow dist_v = D$). The closure of IM_{m_i} implies that v leaves the set A during this action. If v activates $\boldsymbol{(R_2)}$ during this action, then v chooses a new parent that has a distance smaller than $D-1$ in γ_j. This implies that this new parent does not belongs to A_j. Then, we have $level_{prnt_v} \prec m_i$. The boundedness of $\mathcal{M}$ implies that $level_v \prec m_i$ in γ_{j+1}. In other words, if a vertex of A_j is activated during the action (γ_j, γ_{j+1}), then it satisfies $v \notin A_{j+1}$.

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

Lemma 8 *For any $m_i \in M(g)$ and for any configuration of $\mathcal{LC}_{m_i}$, any execution of $\mathcal{SSMMT}$ starting from γ under the distributed weakly fair daemon reaches in a finite time a configuration such that $IM_{m_{i+1}}$ holds.*

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

Lemma 9 *For any $m_i \in M(g)$ and for any configuration $\gamma \in \mathcal{LC}_{m_i}$, any execution of $\mathcal{SSMMT}$ starting from γ under the distributed weakly fair daemon reaches in a finite time a configuration of $\mathcal{LC}_{m_{i+1}}$.*

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

We know that $V_{m_i} \neq \emptyset$ since $r \in V_{m_i}$ for any $i \geq 0$. Remind that $V_{m_{i+1}}$ is connected by the boundedness of $\mathcal{M}$. Then, we know that there exists at least one vertex p of $P_{m_{i+1}}$ that has a neighbor q in V_{m_i} such that $max\_met(g,p,r) = met(max\_met(g,q,r), w_{p,q})$. Moreover, Lemma 3 ensures us that any vertex of V_{m_i} is not activated in any execution starting from γ'.

Observe that, for any configuration of any execution starting from γ', if any vertex of $P_{m_{i+1}}$ is not enabled, then all vertices v of $P_{m_{i+1}}$ satisfy $spec_{MMT}(v)$. Assume now that there exists an execution σ starting from γ' where some vertices of $P_{m_{i+1}}$ are infinitely often activated. By construction, at least one of these vertices (note it v) has a neighbor u such that $max\_met(g,v,r) = met(max\_met(g,u,r),\ w_{v,u})$ that takes only a finite number of actions in σ (recall the construction of p). After u takes its last action of σ, we can observe that $level_u = max\_met(g,u,r)$ and $dist_u < D-1$ (otherwise, u is activated in a finite time that contradicts its construction).

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

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

Lemma 10 *Starting from any configuration, any execution of $\mathcal{SSMMT}$ under the distributed weakly fair daemon reaches a configuration of $\mathcal{LC}_{MMT}$ in a finite time.*

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

Theorem 2 *$\mathcal{SSMMT}$ is a $(S_B, n-1)$-TA strictly stabilizing distributed protocol for $spec_{MMT}$ under the distributed weakly fair daemon.*

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

5.3 Proof of Topology-Aware Strong Stabilization

In this section, we prove the $(t, S_B^*, n-1)$-TA strong stabilization of $\mathcal{SSMMT}$ under the distributed k-bounded strongly fair daemon. Note that k may be any arbitrary natural number. Nevertheless, the actual value of k influences the maximal number of disruptions t of $\mathcal{SSMMT}$.

The key idea is to focus on vertices of $S_B \setminus S_B^*$ after the convergence of $\mathcal{SSMMT}$ on S_B and to prove the two following properties: any such vertex executes a bounded number of steps in any execution and cannot remain unactivated if it does not satisfy $spec_{MMT}$.

Let us denote $E_B = S_B \setminus S_B^*$ (*i.e.* E_B is the set of vertices v such that $max\_met(g,v,r) = \max_{b \in B}\{max\_met(g,v,b)\}$). Intuitively, E_B gathers vertices that are at equal distance (with respect to $\mathcal{M}$) from the root and from the nearest Byzantine vertex. Note that the communication subgraph $g(E_B)$ induced by E_B may have several connected components. In the following, we use the following notations: $E_B = \{E_B^0, \dots, E_B^\ell\}$ where each E_B^i ($i \in \{0, \dots, \ell\}$) is a subset of E_B inducing a maximal (in the number of vertices) connected component of g, $g(E_B^i)$ ($i \in \{0, \dots, \ell\}$) is the communication subgraph induced by E_B^i, and then $diam(g(E_B)) = \max_{i \in \{0,\dots,\ell\}}\{diam(g(E_B^i))\}$. When a and b are two integers, we define the following function: $\Pi(a,b) = \frac{a^{b+1}-1}{a-1}$.

Lemma 11 *If γ is a configuration of $\mathcal{LC}_{MMT}$, then any vertex $v \in E_B$ is activated at most $\Pi(k, diam(g(E_B))) \times deg(g) \times D$ time in any execution starting from γ.*

Proof Let γ be a configuration of $\mathcal{LC}_{MMT}$ and σ be an execution starting from γ. Let p be a vertex of E_B^i ($i \in \{0, \dots, \ell\}$) such that there exists a neighbor q that satisfies $q \in V \setminus S_B$ and $max\_met(g,p,r) = met(max\_met(g,q,r), w_{p,q})$ (such a vertex exists by construction of E_B^i). We are going to prove by induction on d the following property:
$\boldsymbol{(P_d)}$: if v is a vertex of E_B^i such that $dist(g(E_B^i), p, v) = d$, then v executes at most $\Pi(k,d) \times deg(g) \times D$ actions in σ.

Initialization: $d = 0$.

This implies that $v = p$. Then, by construction, there exists a neighbor q that satisfies $q \in V \setminus S_B$ and $max\_met(g,p,r) = met(max\_met(g,q,r), w_{p,q})$. As $\gamma \in \mathcal{LC}_{MMT}$, Lemma 4 ensures us that $level_q = max\_met(g,q,r)$ and $dist_q < D - 1$ in any configuration of σ. Then, the boundedness of $\mathcal{M}$ implies that q belongs to the set that is parameter to the function $next_v$ at any execution of rules $\boldsymbol{(R_2)}$ or $\boldsymbol{(R_3)}$ by p. Consequently, p executes at most $deg(g)$ time rules $\boldsymbol{(R_2)}$ and $\boldsymbol{(R_3)}$ in σ before choosing q as its parent. Moreover, note that p can execute rule $\boldsymbol{(R_1)}$ at most D time between two consecutive executions of rules $\boldsymbol{(R_2)}$ and $\boldsymbol{(R_3)}$ (because $\boldsymbol{(R_1)}$ only increases $dist_p$ that is bounded by D). Consequently, p executes at most $deg(g) \times D$ actions before choosing q as its parent.

By Lemma 4, we know that q takes no action in σ. Once p chooses q as its parent, its state is consistent with the one of q (by construction of rules $\boldsymbol{(R_2)}$ and $\boldsymbol{(R_3)}$). Hence, p is never enabled after choosing q as its parent. Consequently, we obtain that p takes at most $deg(g) \times D$ actions in σ, that proves $\boldsymbol{(P_0)}$.

Induction: $d > 0$ and $\boldsymbol{(P_{d-1})}$ is true.

Let v be a vertex of E_B^i such that $dist(g(E_B^i), p, v) = d$. By construction, there exists a neighbor u of v that belongs to E_B^i such that $dist(g(E_B^i), p, u) = d-1$. By $\boldsymbol{(P_{d-1})}$, we know that u takes at most $\Pi(k, d-1) \times deg(g) \times D$ actions in σ. The k-bounded- ness of the daemon allows us to conclude that v takes at most $k \times \Pi(k, d-1) \times deg(g) \times D$ actions before the last action of u. Then, a similar reasoning to the one of the initialization part allows us to say that v takes at most $deg(g) \times D$ actions after the last action of u (note that the fact that $|M(g)| \geq 2$, the construction of D and the management of $dist$ variables imply that $dist_u < D - 1$ after the last action of u). In conclusion, v takes at most $k \times \Pi(k, d-1) \times deg(g) \times D + deg(g) \times D = \Pi(k, d) \times deg(g) \times D$ actions in σ, that proves $\boldsymbol{(P_d)}$.

As $diam(g(E_B))$ is the maximal diameter of connected components of the communication subgraph induced by E_B, then we know that $dist(g(E_B^i), p, v) \leq diam(g(E_B))$ for any vertex v in E_B^i. For any vertex v of E_B, there exists $i \in \{0, \dots, \ell\}$ such that $v \in E_B^i$. We can deduce that any vertex of E_B takes at most $\Pi(k, diam(g(E_B))) \times deg(g) \times D$ actions in σ, that implies the result. $\square$

Lemma 12 *If γ is a configuration of $\mathcal{LC}_{MMT}$ and v is a vertex such that $v \in E_B$, then for any execution σ starting from γ either*

1. *there exists a configuration γ' of σ such that $spec_{MMT}(v)$ is always satisfied after γ'; or*
2. *v is activated in σ.*

Proof Let γ be a configuration of $\mathcal{LC}_{MMT}$ and v be a vertex such that $v \in E_B$. By contradiction, assume that there exists an execution σ starting from γ such that (i) $spec_{MMT}(v)$ is infinitely often false in σ and (ii) v is never activated in σ.

For any configuration $\gamma*$, let us denote by $P_v(\gamma*) = (v_0 = v, v_1 = prnt_v, v_2 = prnt_{v_1}, \dots, v_k = prnt_{v_{k-1}}, p_v = prnt_{v_k})$ the maximal sequence of vertices following pointers $prnt$ (maximal means here that either $prnt_{p_v} = \bot$ or p_v is the first vertex such that there $p_v = v_i$ for some $i \in \{0, \dots, k\}$).

Let us study the following cases:

Case 1: $prnt_v \in V \setminus S_B$ in γ.

Since $\gamma \in \mathcal{LC}_{MMT}$, $prnt_v$ satisfies $spec_{MMT}(prnt_v)$ in γ and in any execution starting from γ (by Lemma 4). Hence, $prnt_v$ is never activated in σ. If v does not satisfy $spec_{MMT}(v)$ in γ, then we have $level_v \neq met(level_{prnt_v}, w_{v,prnt_v})$

or $dist_v \neq 0$ in γ. Then, v is continuously enabled in σ and we have a contradiction between assumption (ii) and the strong fairness of the daemon. This implies that v satisfies $spec_{MMT}(v)$ in γ. The fact that $prnt_v$ is never activated in γ and that the state of v is consistent with the one of $prnt_v$ ensures us that v is never enabled in any execution starting from γ. Hence, $spec_{MMT}(v)$ remains true in any execution starting from γ. This contradicts the assumption (i) on σ.

Case 2: $prnt_v \notin V \setminus S_B$ in γ.
By the assumption (i) on σ, we can deduce that there exists infinitely many configurations γ' such that a vertex of $P_v(\gamma')$ is enabled (since $spec_{MMT}(v)$ is false only when the state of a vertex of $P_v(\gamma')$ is not consistent with the one of its parent that made it enabled). By construction, the length of $P_v(\gamma')$ is finite for any configuration γ' and there exists only a finite number of vertices in the communication graph. Consequently, there exists at least one vertex that is infinitely often enabled in σ. Since the daemon is strongly fair, we can conclude that there exists at least one vertex that is infinitely often activated in σ.
Let A_σ be the set of vertices that are infinitely often activated in σ. Note that $v \notin A_\sigma$ by assumption (ii) on σ. Let σ' be the suffix of σ starting from γ' that contains only activations of vertices of A_σ. Let p be the first vertex of $P_v(\gamma')$ that belongs to A_σ (p exists since at least one vertex of P_v is enabled when $spec_{MMT}(v)$ is false). By construction, the prefix of $P_v(\gamma'')$ from v to p in any configuration γ'' of σ remains the same as the one of $P_v(\gamma')$. Let p' be the vertex such that $prnt_{p'} = p$ in σ' (p' exists since $v \neq p$ implies that the prefix of $P_v(\gamma')$ from v to p counts at least two vertices). As p is infinitely often activated and as any activation of p modifies the value of $level_p$ or of $dist_p$ (at least one of these two variables takes at least two different values in σ'), we can deduce that p' is infinitely often enabled in σ' (since the value of $level_{p'}$ is constant by construction of σ' and p). Since the daemon is strongly fair, p' is activated in a finite time in σ', that contradicts the construction of p.

In the two cases, we obtain a contradiction with the construction of σ, that proves the result. $\square$

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

$$\mathcal{LC}^*_{MMT} = \Big\{\gamma \in \Gamma \big| (\gamma \textit{ is } S^*_B\textit{-legitimate for } spec_{MMT}) \wedge (IM_{m_k}(\gamma) = true)\Big\}$$

Note that, as $S^*_B \subseteq S_B$, we can deduce that $\mathcal{LC}^*_{MMT} \subseteq \mathcal{LC}_{MMT}$. Hence, properties of Lemmas 11 and 12 also apply to configurations of $\mathcal{LC}^*_{MMT}$.

Lemma 13 *Any configuration of $\mathcal{LC}^*_{MMT}$ is $(n \times \Pi(k, diam(g(E_B)) \times deg(g) \times D, \Pi(k, diam(g(E_B)) \times deg(g) \times D, S^*_B, n-1)$-TA time contained for $spec_{MMT}$.*

Proof Let γ be a configuration of $\mathcal{LC}^*_{MMT}$. As $S^*_B \subseteq S_B$, we know by Lemma 4 that any vertex v of $V \setminus S_B$ satisfies $spec_{MMT}(v)$ and takes no action in any execution starting from γ.

Let v be a vertex of E_B. By Lemmas 11 and 12, we know that v takes at most $\Pi(k, diam(g(E_B)) \times deg(g) \times D$ actions in any execution starting from γ. Moreover, we know that v satisfies $spec_{MMT}(v)$ after its last action (otherwise, we obtain a contradiction between the two lemmas). Hence, any vertex of E_B takes at most $\Pi(k, diam(g(E_B)) \times deg(g) \times D$ actions and then, there are at most $n \times \Pi(k, diam(g(E_B)) \times deg(g) \times D$ S_B^*-TA disruptions in any execution starting from γ (since $|E_B| \leq n$).

By definition of a TA time contained configuration, we obtain the result. □

Lemma 14 *Starting from any configuration, any execution of $\mathcal{SSMMT}$ under the distributed k-bounded strongly fair daemon reaches a configuration of $\mathcal{LC}^*_{MMT}$ in a finite time.*

Proof Let γ be an arbitrary configuration. We know by Lemma 10 that any execution starting from γ reaches in a finite time a configuration γ' of $\mathcal{LC}_{MMT}$.

Let v be a vertex of E_B. By Lemmas 11 and 12, we know that v takes at most $\Pi(k, diam(g(E_B)) \times deg(g) \times D$ actions in any execution starting from γ'. Moreover, we know that v satisfies $spec_{MMT}(v)$ after its last action (otherwise, we obtain a contradiction between the two lemmas). This implies that any execution starting from γ' reaches a configuration γ'' such that any vertex v of E_B satisfies $spec_{MMT}(v)$. It is easy to see that $\gamma'' \in \mathcal{LC}^*_{MMT}$, that ends the proof. □

Theorem 3 *$\mathcal{SSMMT}$ is a $(n \times \Pi(k, diam(g(E_B)) \times deg(g) \times D), S_B^*, n-1)$-TA strongly stabilizing distributed protocol for $spec_{MMT}$ under the distributed k-bounded strongly fair daemon.*

Proof This result is a direct consequence of Lemmas 13 and 14. □

6 Optimality of Containment Areas

This section presents two impossibility results that prove the optimality of containment areas provided by the distributed protocol of the previous section. Indeed, Theorem 4 states that there exists no topology-aware strictly stabilizing distributed protocol for maximum metric spanning tree construction for any containment area strictly included in S_B while Theorem 5 proves that there exists no topology-aware strongly stabilizing distributed protocol for maximum metric spanning tree construction for any containment area strictly included in S_B^*.

These proofs are based on the construction of a communication graph (depending of the characteristic of the considered maximizable metric) and of a Byzantine behavior that allows us to invalidate the topology-aware strict (respectively strong) stabilization of any distributed protocol exhibiting better containment areas than $\mathcal{SSMMT}$.

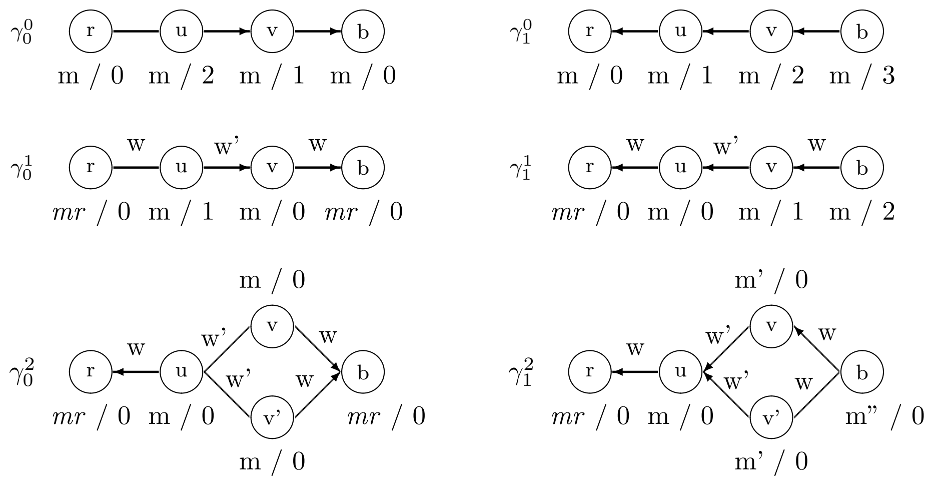

Fig. 4 Configurations used in proof of Theorem 4.

6.1 Topology-Aware Strict Stabilization

Theorem 4 *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 distributed protocol for $spec_{MMT}$ with respect to $\mathcal{M}$ where $A_B \subsetneq S_B$.*

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

Case 1: $|M| = 1$.

Denote by m the metric value such that $M = \{m\}$. For any communication graph and for any vertex $v \neq r$, we have:

$$max\_met(g, v, r) = \underset{b \in B}{min_{\prec}}\{max\_met(g, v, b)\} = m$$

Consequently, $S_B = V \setminus (B \cup \{r\})$ for any communication graph.

Consider the following communication graph: $V = \{r, u, v, b\}$ and $E = \{\{r, u\}, \{u, v\}, \{v, b\}\}$ (b is a Byzantine vertex). As $S_B = \{u, v\}$ and $A_B \subsetneq S_B$, we have: $u \notin A_B$ or $v \notin A_B$. Consider now the following configuration γ_0^0: $prnt_r = prnt_b = \bot$, $prnt_v = b$, $prnt_u = v$, $level_r = level_u = level_v = level_b = m$, $dist_r = dist_b = 0$, $dist_v = 1$ and $dist_u = 2$ (see Figure 4, other variables may have arbitrary values). Note that γ_0^0 is A_B-legitimate for $spec_{MMT}$ (whatever A_B is).

Assume now that b behaves as a correct vertex with respect to π. Then, by convergence of π in a fault-free system starting from γ_0^0 that is not $\emptyset$-legitimate (remember that a topology-aware strictly stabilizing distributed protocol is a special case of self-stabilizing distributed protocol), we can deduce that π reaches in a finite time a configuration γ_1^0 (see Figure 4), where: $prnt_r = \bot$, $prnt_u = r$, $prnt_v = u$, $prnt_b = v$, $level_r = level_u =$

$\mathit{level}_v = \mathit{level}_b = m$, $\mathit{dist}_r = 0$, $\mathit{dist}_u = 1$, $\mathit{dist}_v = 2$ and $\mathit{dist}_b = 3$. Note that vertices u and v modify their O-variables in this execution. This contradicts the $(A_B, 1)$-TA strict stabilization of π (whatever A_B is).

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

By definition of a bounded metric and from the utility condition, we can deduce that there exist $m \in M$ and $w \in W$ such that $m = \mathit{met}(mr, w) \prec mr$. Then, we must distinguish the following cases:

Case 2.1: m is a fixed point of $\mathcal{M}$.

Consider the following communication graph: $V = \{r, u, v, b\}$, $E = \{\{r, u\}, \{u, v\}, \{v, b\}\}$, $w_{r,u} = w_{v,b} = w$, and $w_{u,v} = w'$ (b is a Byzantine vertex). As for any $w' \in W$, $\mathit{met}(m, w') = m$ (by definition of a fixed point), we have: $S_B = \{u, v\}$. Since $A_B \subsetneq S_B$, we have: $u \notin A_B$ or $v \notin A_B$. Consider now the following configuration γ_0^1: $\mathit{prnt}_r = \mathit{prnt}_b = \bot$, $\mathit{prnt}_v = b$, $\mathit{prnt}_u = v$, $\mathit{level}_r = \mathit{level}_b = mr$, $\mathit{level}_u = \mathit{level}_v = m$, $\mathit{dist}_r = \mathit{dist}_b = 0$, $\mathit{dist}_v = 0$ and $\mathit{dist}_u = 1$ (see Figure 4, other variables may have arbitrary values). Note that γ_0^1 is A_B-legitimate for spec_{MMT} (whatever A_B is).

Assume now that b behaves as a correct vertex with respect to π. Then, by convergence of π in a fault-free system starting from γ_0^1 that is not $\emptyset$-legitimate (remember that a topology-aware strictly stabilizing distributed protocol is a special case of self-stabilizing distributed protocol), we can deduce that π reaches in a finite time a configuration γ_1^1 (see Figure 4), where: $\mathit{prnt}_r = \bot$, $\mathit{prnt}_u = r$, $\mathit{prnt}_v = u$, $\mathit{prnt}_b = v$, $\mathit{level}_r = mr$, $\mathit{level}_u = \mathit{level}_v = \mathit{level}_b = m$ (since m is a fixed point), $\mathit{dist}_r = 0$, $\mathit{dist}_u = 0$, $\mathit{dist}_v = 1$ and $\mathit{dist}_b = 2$. Note that vertices u and v modify their O-variables in this execution. This contradicts the $(A_B, 1)$-TA strict stabilization of π (whatever A_B is).

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

This implies that there exists $w' \in W$ such that: $\mathit{met}(m, w') \prec m$ (remember that $\mathcal{M}$ is bounded). Consider the following communication graph:

$V = \{r, u, v, v', b\}$, $E = \{\{r, u\}, \{u, v\}, \{u, v'\}, \{v, b\}, \{v', b\}\}$, $w_{r,u} = w_{v,b} = w_{v',b} = w$, and $w_{u,v} = w_{u,v'} = w'$ (b is a Byzantine vertex). 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 γ_0^2: $\mathit{prnt}_r = \mathit{prnt}_b = \bot$, $\mathit{prnt}_v = \mathit{prnt}_{v'} = b$, $\mathit{prnt}_u = r$, $\mathit{level}_r = \mathit{level}_b = mr$, $\mathit{level}_u = \mathit{level}_v = \mathit{level}_{v'} = m$, $\mathit{dist}_r = \mathit{dist}_b = 0$, $\mathit{dist}_v = \mathit{dist}_{v'} = 0$ and $\mathit{dist}_u = 0$ (see Figure 4, other variables may have arbitrary values). Note that γ_0^2 is A_B-legitimate for spec_{MMT} (whatever A_B is).

Assume now that b behaves as a correct vertex with respect to π. Then, by convergence of π in a fault-free system starting from γ_0^2 that is not $\emptyset$-legitimate (remember that a topology-aware strictly stabilizing distributed protocol is a special case of self-stabilizing distributed protocol), we can deduce that π reaches in a finite time a configuration γ_1^2 (see Figure 4), where: $\mathit{prnt}_r = \bot$, $\mathit{prnt}_u = r$, $\mathit{prnt}_v = \mathit{prnt}_{v'} = u$, $\mathit{prnt}_b = v$ (or $\mathit{prnt}_b = v'$), $\mathit{level}_r = mr$, $\mathit{level}_u = m$ $\mathit{level}_v = \mathit{level}_{v'} = \mathit{met}(m, w') =$

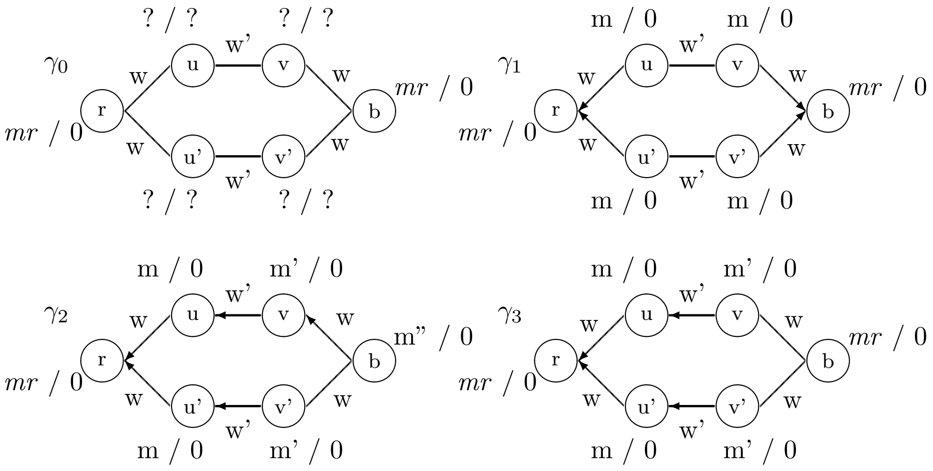

Fig. 5 Configurations used in proof of Theorem 5.

m', $level_b = met(m', w) = m''$, $dist_r = 0$, $dist_u = 0$, $dist_v = dist_{v'} = 0$ and $dist_b = 0$. Note that vertices v and v' modify their O-variables in this execution. This contradicts the $(A_B, 1)$-TA strict stabilization of π (whatever A_B is).

□

6.2 Topology-Aware Strong Stabilization

Theorem 5 *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 distributed protocol for $spec_{MMT}$ with respect to $\mathcal{M}$ where $A_B^* \subsetneq S_B^*$ and t is any finite integer.*

Proof Let $\mathcal{M} = (M, W, mr, met, \prec)$ be a maximizable metric and π be a $(t, A_B^*, 1)$-TA strongly stabilizing protocol for $spec_{MMT}$ 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 communication graph and for any vertex v, we have:

$$max\_met(g, v, r) = \underset{b \in B}{min_\prec}\{max\_met(g, v, b)\} = m$$

Consequently, $S_B^* = \emptyset$ for any communication graph. Then, it is absurd to have $A_B^* \subsetneq S_B^*$.

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

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

Case 2.1: m is a fixed point of $\mathcal{M}$.

Let g be a communication graph such that any edge incident to the root or a Byzantine vertex has a weight equals to w. Then, we can deduce that we have:

$$m = \max_{b \in B}{}_{\prec}\{max\_met(g, r, b)\} \prec max\_met(g, r, r) = mr$$

and

$$\forall v \in V \setminus (B \cup \{r\}), max\_met(g, v, r) = \max_{b \in B}{}_{\prec}\{max\_met(g, v, b)\} = m$$

Hence, $S_B^* = \emptyset$ for any such communication graph. 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 communication graph: $V = \{r, u, u', v, v', b\}$, $E = \{\{r, u\}, \{r, u'\}, \{u, v\}, \{u', v'\}, \{v, b\}, \{v', b\}\}$, $w_{r,u} = w_{r,u'} = w_{v,b} = w_{v',b} = w$, and $w_{u,v} = w_{u',v'} = w'$ (b is a Byzantine vertex). 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 γ_0: $prnt_r = prnt_b = \bot$, $level_r = level_b = mr$, $dist_r = dist_b = 0$ and $prnt$, $level$, and $dist$ variables of other vertices are arbitrary (see Figure 5, other variables may have arbitrary values but other variables of b are identical to those of r).

Assume now that b executes exactly the same actions as r (if any) immediately after r (note that $r \notin A_B^*$ and hence $prnt_r = \bot$, $level_r = mr$, and $dist_r = 0$ still hold by closure and then $prnt_b = \bot$, $level_b = mr$, and $dist_r = 0$ still hold too). Then, by symmetry of the execution and by convergence of π to $spec_{MMT}$, we can deduce that π reaches in a finite time a configuration γ_1 (see Figure 5), where: $prnt_r = prnt_b = \bot$, $prnt_u = prnt_{u'} = r$, $prnt_v = prnt_{v'} = b$, $level_r = level_b = mr$, $level_u = level_{u'} = level_v = level_{v'} = m$, and $\forall v \in V, dist_v = legal\_dist(v, prnt_v)$ (because this configuration is the only one where every correct vertex v satisfies $spec_{MMT}(v)$ when $prnt_r = prnt_b = \bot$ and $level_r = level_b = mr$ since $met(m, w') \prec m$). Note that γ_1 is A_B^*-legitimate for $spec_{MMT}$ and A_B^*-stable (whatever A_B^* is).

Assume now that b behaves as a correct vertex with respect to π. Then, by convergence of π in a fault-free system starting from γ_1 that is not $\emptyset$-legitimate (remember that a TA strongly stabilizing distributed protocol is a special case of self-stabilizing distributed protocol), we can deduce that π reaches in a finite time a configuration γ_2 (see Figure 5), where: $prnt_r = \bot$, $prnt_u = prnt_{u'} = r$, $prnt_v = u$,

$prnt_{v'} = u'$, $prnt_b = v$ (or $prnt_b = v'$), $level_r = mr$, $level_u = level_{u'} = m$ $level_v = level_{v'} = met(m, w') = m'$, $level_b = met(m', w) = m''$, and $\forall v \in V$, $dist_v = legal\_dist(v, prnt_v)$. Note that vertices v and v' modify their O-variables in the portion of execution between γ_1 and γ_2 and that γ_2 is A^*_B-legitimate for $spec_{MMT}$ 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 vertex b takes the following state: $prnt_b = \perp$, $level_b = mr$, and $dist_b = 0$. This action brings the system into configuration γ_3 (see Figure 5). From this configuration, we can repeat the execution we constructed from γ_0. By the same token, we obtain an execution of π that contains A^*_B-legitimate and A^*_B-stable configurations (see γ_1) and an infinite number of A^*_B-TA disruptions (whatever A^*_B is), which contradicts the $(t, A^*_B, 1)$-TA strong stabilization property of π.

□

7 Conclusion

Summary This paper focused on maximum metric spanning tree construction in distributed systems simultaneously subject to transient and Byzantine faults. Spanning tree construction is a fundamental task in distributed systems since it permits to construct a virtual communication structure that allows every vertex to communicate using a minimal number of edges of the original communication graph. According to desired characteristics of the spanning tree (minimum weight, shortest path to the root, minimal degree,...), there exist numerous self-stabilizing distributed protocols.

To our knowledge, our work is the first to consider spanning tree construction in presence of both transient and Byzantine faults. As this problem is global (whatever the considered spanning tree properties are), there exists no strictly-stabilizing solution for any (finite) containment radius by the generic impossibility result of Nesterenko and Arora [33]. Therefore, our main contribution is to propose three new schemes of Byzantine containment in self-stabilization in order to by-pass this impossibility result.

First, we proposed strong stabilization, where the constraint about the containment radius is relaxed, *i.e.* there may exist vertices outside the containment radius that invalidate the specification predicate, due to Byzantine actions. However, the impact of Byzantine triggered-actions is limited in times: the set of Byzantine vertices may only impact vertices outside the containment radius a bounded number of times, even if Byzantine vertices execute an infinite number of actions. This new scheme of Byzantine containment in self-stabilization generalizes strict stabilization as a strictly stabilizing distributed protocol is a strongly stabilizing one with a maximal number of disruptions equal to 0.

	Containment	Result	Proved by...
(C_B, f)-TA strict stabilization	$C_B \subsetneq S_B, f = 1$	Impossible	Theorem 4
	$C_B = S_B, f = n - 1$	Possible	Theorem 2
(t, C_B, f)-TA strong stabilization	$C_B \subsetneq S_B^*, f = 1$	Impossible	Theorem 5
	$C_B = S_B^*, f = n - 1$	Possible	Theorem 3

Table 1 Summary of results related to $spec_{MMT}$.

Although this new scheme is sufficient to by-pass some impossibility results [20], it is still too strong for some problems as there remain impossibility results in the context of strong stabilization. We proposed a new notion for Byzantine containment in self-stabilization: the topology-aware stabilization. Here, the requirement about the containment radius is relaxed to a containment area, *i.e.* the set of vertices that may be disturbed by Byzantine ones is not reduced to the union of c-neighborhood of Byzantine vertices but is defined as a function of the communication graph and Byzantine vertices locations. Note that this relaxation may be applied either to strict or to strong stabilization.

To demonstrate the effectiveness of our notions of topology-aware stabilization, we focused on a large class of spanning tree constructions: the maximum metric spanning tree construction with respect to any maximizable metric. Intuitively, a metric is a scheme to compute a distance along any path of the communication graph. A metric is maximizable if there always exists a spanning tree that maximizes the metric of each vertex of any communication graph with respect to a distinguished vertex called the root. For example, the shortest path or the flow metric are maximizable. In contrast, there exists no maximizable metric to model the minimum weight or the minimum degree spanning tree construction. In this paper, we characterize the possibility range of each of these fault tolerance schemes. All results are summarized in Table 1. Note that these results subsume those presented in related previous works [20, 18].

Open questions The results presented in this paper show that our new notions of topology-aware stabilization are convenient to circumvent impossibility results related to strict stabilization. Nevertheless, interesting open questions remain.

Daemon requirements for the correctness of our distributed protocols were not discussed here. It would be interesting to prove their necessity or to provide distributed protocols operating with weaker daemon, if possible. In this case, is it possible to keep the optimality of containment areas? Another way to complement results of this paper is to study the relationship between the containment areas and the maximal number of disruptions. Intuitively, if constraints on containment area are weakened, the maximal number of disruptions may decrease. While maximizable metrics are a large class of metrics, there exist numerous other metrics to construct spanning tree. We think that the study of Byzantine containment properties of these metrics is worth studying.

Note that the distributed protocols provided in this paper that achieve topology-aware strong stabilization require a strongly fair daemon. We conjecture that the strong fairness property is necessary to perform (topology-aware) strong stabilization. Formally proving this conjecture is an interesting open question.

Finally, we presented definitions for strong stabilization and topology-aware stabilization for static problems, *i.e.* problems that require the system to find static solutions, such as the maximum metric spanning tree construction. An interesting path for future research may be to provide strong or topology-aware stabilizing distributed protocols for other static problems or to extend our definitions to dynamic problems such as token circulation.

References

1. Afek Y, Kutten S, Yung M (1990) Memory-efficient self stabilizing protocols for general networks. In: 4th International Workshop on Distributed Algorithms (WDAG 1990), pp 15–28
2. Blin L, Butelle F (2004) The first approximated distributed algorithm for the minimum degree spanning tree problem on general graphs. International Journal of Foundations of Computer Science 15(3):507–516
3. Blin L, Potop-Butucaru M, Rovedakis S (2009) A superstabilizing log()-approximation algorithm for dynamic steiner trees. In: 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2009), pp 133–148
4. Blin L, Potop-Butucaru M, Rovedakis S (2011) Self-stabilizing minimum degree spanning tree within one from the optimal degree. Journal of Parallel and Distributed Computing 71(3):438–449
5. Burman J, Kutten S (2007) Time optimal asynchronous self-stabilizing spanning tree. In: 21st International Symposium on Distributed Computing (DISC 2007), pp 92–107
6. Butelle F, Lavault C, Bui M (1995) A uniform self-stabilizing minimum diameter tree algorithm (extended abstract). In: 9th International Workshop on Distributed Algorithms (WDAG 1995), pp 257–272
7. Chen GH, Houle ME, Kuo MT (1993) The steiner problem in distributed computing systems. Information Sciences 74(1-2):73 – 96
8. Collin Z, Dolev S (1994) Self-stabilizing depth-first search. Information Processing Letters 49(6):297–301
9. Daliot A, Dolev D (2005) Self-stabilization of byzantine protocols. In: 7th International Symposium on Self-Stabilizing Systems (SSS 2005), pp 48–67
10. Datta AK, Johnen C, Petit F, Villain V (2000) Self-stabilizing depth-first token circulation in arbitrary rooted networks. Distributed Computing 13:207–218

11. Delaët S, Ducourthial B, Tixeuil S (2006) Self-stabilization with r-operators revisited. Journal of Aerospace Computing, Information, and Communication 3(10):498–514
12. Dijkstra EW (1974) Self-stabilizing systems in spite of distributed control. Communication of ACM 17(11):643–644
13. Dolev S (2000) Self-stabilization. MIT Press
14. Dolev S, Welch JL (2004) Self-stabilizing clock synchronization in the presence of byzantine faults. Journal of the ACM 51(5):780–799
15. Dolev S, Israeli A, Moran S (1990) Self-stabilization of dynamic systems assuming only read/write atomicity. In: 9th Annual ACM Symposium on Principles of Distributed Computing (PODC 1990), pp 103–117
16. Dolev S, Israeli A, Moran S (1993) Self-stabilization of dynamic systems assuming only read/write atomicity. Distributed Computing 7(1):3–16
17. Dubois S, Masuzawa T, Tixeuil S (2010) The impact of topology on byzantine containment in stabilization. In: 24th International Symposium on Distributed Computing (DISC 2010)
18. Dubois S, Masuzawa T, Tixeuil S (2010) On byzantine containment properties of the min+1 protocol. In: 12th International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2010)
19. Dubois S, Masuzawa T, Tixeuil S (2011) Maximum metric spanning tree made byzantine tolerant. In: 25th International Symposium on Distributed Computing (DISC 2011)
20. Dubois S, Masuzawa T, Tixeuil S (2012) Bounding the impact of unbounded attacks in stabilization. IEEE Transactions on Parallel and Distributed Systems 23(3):460–466
21. Ducourthial B, Tixeuil S (2001) Self-stabilization with r-operators. Distributed Computing 14(3):147–162
22. Ducourthial B, Tixeuil S (2003) Self-stabilization with path algebra. Theoretical Computer Science 293(1):219–236
23. Gallager RG, Humblet PA, Spira PM (1983) A distributed algorithm for minimum-weight spanning trees. ACM Transactions on Programming Languages and Systems 5(1):66–77
24. Gärtner FC (2003) A survey of self-stabilizing spanning-tree construction algorithms. Technical report ic/2003/38, EPFL
25. Gouda MG, Schneider M (1994) Maximum flow routing. In: Joint Conference on Information Sciences (JCIS 1994)
26. Gouda MG, Schneider M (1999) Stabilization of maximal metric trees. In: ICDCS Workshop on Self-stabilizing Systems (WSS 1999), pp 10–17
27. Gouda MG, Schneider M (2003) Maximizable routing metrics. IEEE/ACM Transactions on Networks 11(4):663–675
28. Huang ST, Chen NS (1992) A self-stabilizing algorithm for constructing breadth-first trees. Information Processing Letters 41(2):109–117
29. Huang ST, Wuu LC (1997) Self-stabilizing token circulation in uniform networks. Distributed Computing 10(4):181–187
30. Huang TC, Lin JC (2002) A self-stabilizing algorithm for the shortest path problem in a distributed system. Computers and Mathematics with

Applications 43(1-2):103 – 109
31. Lamport L, Shostak RE, Pease MC (1982) The byzantine generals problem. ACM Transactions on Programming Langage System 4(3):382–401
32. Masuzawa T, Tixeuil S (2007) Stabilizing link-coloration of arbitrary networks with unbounded byzantine faults. International Journal of Principles and Applications of Information Science and Technology 1(1):1–13
33. Nesterenko M, Arora A (2002) Tolerance to unbounded byzantine faults. In: 21st Symposium on Reliable Distributed Systems (SRDS 2002), IEEE Computer Society, pp 22–29
34. Schneider M (1997) Flow routing in computer networks. PhD thesis, University of Texas at Austin
35. Tixeuil S (2009) Algorithms and Theory of Computation Handbook, Second Edition, CRC Press, Taylor & Francis Group, chap Self-stabilizing Algorithms, pp 26.1–26.45. Chapman & Hall/CRC Applied Algorithms and Data Structures
36. Tsai MS, Huang ST (1994) A self-stabilizing algorith for the shortest paths problem with a fully distributed demon. Parallel Processing Letters 4:65–72