An Exercise in Selfish Stabilization

JOHANNE COHEN
LORIA-CNRS, France
ANURAG DASGUPTA and SUKUMAR GHOSH
University of Iowa, USA
and
SÉBASTIEN TIXEUIL
Université Paris-Sud, France

Stabilizing distributed systems expect all the component processes to run predefined programs that are externally mandated. In Internet scale systems, this is unrealistic, since each process may have selfish interests and motives related to maximizing its own payoff. This paper formulates the problem of selfish stabilization that shows how competition blends with cooperation in a stabilizing environment.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols—*verification*; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*; D.1.1 [**Programming Techniques**]: General; D.1.3 [**Programming Techniques**]: Concurrent Programming; D.2.5 [**Software Engineering**]: Design—*Methodologies*

General Terms: Algorithm, Design, Theory

Additional Key Words and Phrases: Stabilization, equilibrium, convergences, selfishness

1. INTRODUCTION

Current research on the design of self-stabilizing (*a.k.a.* stabilizing) distributed systems [5; 6] assumes that all processes run predefined programs mandated by an external agency who is the owner or the administrator of the entire system. The model is acceptable only when processes cooperate with one another, and the goal is purely a global one. The model falls apart when the distributed system spans over multiple administrative domains or processes have private goals too. On Internet-scale distributed systems, each process or each administrative domain may have selfish motives to maximize its own *payoff*. In fact, payoffs or cost functions have been the major driving force behind *game theory*, but individual payoffs never

Sukumar Ghosh's research was supported in part by the Alexander von Humboldt Foundation, Germany. Sébastien Tixeuil was supported in part by the FRAGILE, SR2I, and SOGEA projects. Johanne Cohen is supported by SOGEA project. Part of this work was done while one of the authors was visiting University of Iowa.
An earlier version of this work was presented in the SSS 2006.

figured into the realm of stabilizing distributed systems. There are many applications where individual payoffs are relevant, but the spirit of competition need not conflict with the general spirit of cooperation that is the driving force behind stabilizing algorithms. To clarify this issue, consider that a system of n processes for which a legal configuration is any element of the set of configurations $\{L_1, \cdots, L_k\}$, but different processes have different preferences about their ideal legal configurations. Attaining the individual goal may be possible via the use of asymmetric cost functions that are statically defined, or by the use of specific strategies that may be adopted at run time. Such strategies refine the basic move. For example, to execute the step of choosing a neighbor, different processes may adopt different strategies for choosing the neighbor. While the choice will impact the payoffs, it will not affect the global goal, or the stabilization mechanism. As an example, consider the stabilizing token circulation protocols [5] that have been widely studied by the stabilization community. If there are several classes of processes with competing interests, then in addition to the common goal of reducing the number of tokens to one, each class may try to retain the token among themselves more often than their competitors. Maximizing individual payoffs under the umbrella of stabilization characterizes the notion of *selfish stabilization*.

Related Work. Selfish stabilization blends game theory with self-stabilization. There are some strong similarities between the two paradigms, but there are significant differences too. Considering the *players* in games to be equivalent to *processes* in a stabilizing system, the equilibrium in games is comparable to the legal configuration of stabilizing systems, in as much as both satisfy the condition of convergence and closure. However, unlike stabilizing systems, games start from predefined initial configurations, and largely ignore faulty moves or transient state corruptions. An exception is the notion of *bounded rationality* (see Herbert Simon [20]) that suggests that economic agents sometimes use heuristics to make decisions rather than a strict rigid rule of optimization in light of the complexity of the situation. In the context of distributed systems, the anarchic behavior of processes for meeting selfish goals can be viewed as a weaker version of byzantine failure. Game theory so far has been a hotbed of activities in computational economics (like auctions) and algorithm design. It is also receiving attention in interdomain routing protocols like BGP. For example, in the *stable path problem* [10], each process has to choose the best path according to some local routing policy, and conflicts between local interests can lead to unstable or oscillating behavior. Cobb et al [4] proposed a stabilizing solution to the stable path problem. Halpern [11] presented a perspective of game theory for distributed systems researchers. Moscibroda, Schmid and Wattenhofer [15] studied the formation of the topology of a P2P network by selfish peers, and presented a negative result that the topology construction may not terminate due to selfish behavior, even if churns are absent. They also analyzed [16] the impact of allowing some processes to be malicious or byzantine, whereas others are selfish, and computed the price of malice. Keidar et al's *Equicast protocol* [13] for multicast in a peer-to-peer network deals with freeloaders by treating the system as a non-cooperative game, where nodes are selsh but rational. Each user chooses its own strategy regarding its level of cooperation to participate in the multicast so as to minimize its own cost. The goal of each node is to receive all the multicast

packets while minimizing its sending rate. Mavronicolas [14] used a game-theoretic view to model security in wireless sensor networks as a game between the attackers and the defenders. Cao et al [3] presented a variation of the pursuer-evader games for asset protection using a large-scale sensor-actuator network - both are only tangentially related to our work, since stabilization is not an issue. Other than these, mixing game theory with distributed computing is certainly on a fast growing curve, yet no specific work has addressed the self-stabilizing setting. This paper aims at bridging the gap.

Contributions. This paper introduces the notion of selfish stabilization, and addresses a specific problem in this domain. Given a graph $G = (V, E)$, assume that there are p different subsets (or classes or colors) of nodes. For each subset or color, there is a separate cost function that maps the set of edges to the set of positive integers. Starting from an arbitrary initial configuration, the p different classes of nodes cooperate with one another to form a spanning tree rooted at a designated node, and at the same time compete against each other to minimize their cost of communication with the root node. The communication cost may depend on various factors: for example, ownership of the routers may be a factor in determining the cost of routing traffic for any class of nodes. The processes are free to choose a strategy from a given set of strategies, and switch strategy to satisfy their individual needs. We examine strategies under which, starting from an arbitrary initial configuration, these classes of processes can stabilize to an equilibrium configuration after which no process can unilaterally decrease its communication cost to the root. The paper has three sections. Section 2 introduces the model and the notations. Section 3 show that with three (or more) different classes of nodes, there exists the possibility that no equilibrium is possible, and proves that determining whether a particular setting admits an equilibrium is NP-complete. On the positive side, Section 3 presents evidence that when there are two (or less) classes, an equilibrium always exists. A (weakly) stabilizing distributed algorithm is provided to construct an equilibrium tree. Alternative strategies for participating nodes are also discussed. Section 4 further analyzes various aspects of the problem, and provides some food for thought.

2. PRELIMINARIES

In this section, we present our model, followed by the various issues addressed in the rest of the paper.

Model and notations. Let $G = (V, E)$ denote the topology of the network where $V = \{1, 2, \ldots, n\}$ is the set of nodes (processes) and E is the set of edges connecting pairs of processes. Each process communicates with its neighbors using the locally shared memory model. The execution of the protocol is organized in steps. In each step, a process executes a guarded action $g \rightarrow A$: the guard g is a boolean whose value depends on its own state and the states of its immediate neighbors. When the guard is true, the process executes an action A that updates its own state. The global state or configuration of the system is a tuple consisting of the local states of all the processes. Unless stated otherwise, a *serial daemon* schedules the action by arbitrarily choosing a process with an enabled guard to execute its action. No fairness (of the demon) is assumed. A *computation* is a maximal sequence of global

states $S_1, S_2, ...$, where S_1 is the initial global state, and each step of the protocol changes the global state S_i to state S_{i+1}.

All nodes in V have a common goal: starting from an arbitrary initial configuration, they collaborate with one another to form a spanning tree with a given node designated as the root. We divide V into p disjoint subsets of nodes $V_1, V_1, V_2, \ldots, V_p$, such that

$$V = \cup_{i=1}^{p} V_i$$

For the ease of description, we associate a distinct color with each subset. In addition to the common goal, these subsets have their own agenda: we call them *private* goals. The private goals of the p different subsets may be conflicting - for example, these subsets of nodes may want to split a common resource. To illustrate such private goals, we convert G into a multi-weighted graph by defining a cost function w of $E \rightarrow \mathbb{N}^{*p}$, where $\mathbb{N}^*$ is the set of (strictly) positive integers. For every $i \in [1 \ldots p]$, the function w_i of $E \rightarrow \mathbb{N}^*$ is defined as:

$$\forall e \in E,\ w_i(e) = x_i \text{ if and only if } w(e) = (x_1, \ldots, x_i, \ldots, x_p)$$

So, for each node in V_i, $w_i(e)$ denotes the cost of using edge e.

When a spanning tree T exists in G, this tree defines a unique path $T_{v \rightarrow u}$ between any two nodes v and u of V. The cost of this path for a particular subset V_i is

$$w_i(T_{v \rightarrow u}) = \sum_{e \in T_{v \rightarrow u}} w_i(e)$$

In this paper, we study the cost of reaching the root r of a spanning tree T from a node v belonging to V_i. This cost is equal to $cost(v) = w_i(T_{v \rightarrow r})$.

The private goal for every node v is to minimize $cost(v)$. A node v is stable in a tree T if it has no incentive to chose another neighbor $z \in \Gamma_g(v)$, *i.e.* choosing z would not lower its cost.

Definition 2.1 *Stable node.* Let $G = (V, E)$ be a graph and r a node of V. Let $\mathcal{P}$ be a partition of V with p elements $V = \cup_{i=1}^{p} V_i$. Let w be a function on $E \rightarrow \mathbb{N}^{*p}$. Let T be a spanning tree of G. The node $v \in V_i$ is *stable in T for metric w* if and only if it satisfies the following condition:

$$\forall z \in \Gamma_G(v),\ \sum_{e \in T_{v \rightarrow r}} w_i(e) \leq w_i((v, z)) + \sum_{e \in T_{z \rightarrow r}} w_i(e)$$

In other word,

$$\forall z \in \Gamma_G(v), w_i(T_{v \rightarrow r}) \leq w_i((v, z)) + w_i(T_{z \rightarrow r})$$

Similarly, a tree T is stable if every node in V is stable.

Definition 2.2 *Stable tree.* Let $G = (V, E)$ be a graph and r a node of V. Let $\mathcal{P}$ be a partition of V with p elements $V = \cup_{i=1}^{p} V_i$. Let w be a function on $E \rightarrow \mathbb{N}^{*p}$. Let T be a spanning tree of G. The tree T is *stable for metric w* if and only if for every node $v \in V$, v is stable in T for metric w.

Unlike traditional stabilization algorithms where all processes execute the same algorithm, here we allow processes to choose different algorithms, or switch algorithms from a set $\Sigma = \{P_1, P_1, \cdots, P_k\}$ (each P_i reflects a distinct strategy) in order

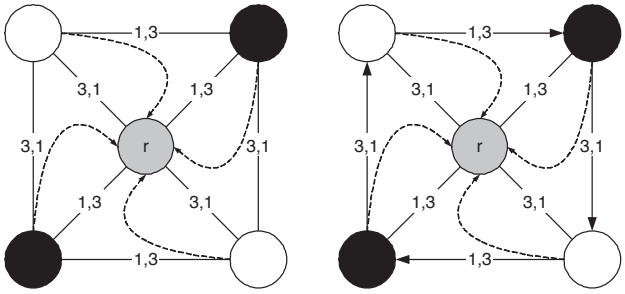

Fig. 1. Pure selfish strategies may not yield a tree. Each node picks an edge whose path towards the root has minimal cost for its own color, and this results in a cycle.

to meet their private goals. Thus, each state transition (S_i, S_{i+1}) is caused by an action of some algorithm $P_j \in \Sigma$. We will call Σ the *strategy space*.

Our goal is to devise a stabilizing mechanism for the construction of a stable tree given a particular metric – the computation should lead to a tree configuration so that the conflicting private goals of cost minimization do not interfere with the common goal of tree formation. The two components of the mechanism are:

(1) **Equilibrium.** The goal configuration corresponds to an equilibrium configuration such that no process has an eligible action that can unilaterally decrease ts own cost.
(2) **Convergence.** Starting from an arbitrary initial configuration, the system of processes must converge to an equilibrium configuration.

We add a few clarifications to explain the above two mechanisms:
Clarification 1. The equilibrium need not always correspond to a quiescent state in which all guards are false. It can also represent the dynamic behavior of a reactive system (like a token-passing system). For the current problem however, a quiescent equilibrium state will suffice, and it naturally reflects a Nash equilibrium.
Clarification 2. In principle, the equilibrium condition can be further generalized to the case where no coalition of processes can decrease the cost of a subset of them.
Clarification 3. Compared to the closure property [1] used in traditional stabilizing systems, the equilibrium criteria is more general in as much as it allows the processes to try out a different strategy that does not conflict with the spirit of cooperation.

Indeed, a selfish strategy without cooperation (such as arbitrarily picking an edge whose cost is minimal for its own color) can result in a graph that is not a tree . Even if nodes are aware of the minimum cost path towards the root, the resulting graph may still not be a tree (see Figure 1). In the subsequent section we show that the problem is non-trivial even if processes stick to a fixed strategy that is known to all. Strategy switch adds another level of complexity to it.
The roadmap. Among the set of all such problems, there are cases where no equilibrium exists, regardless of the strategy chosen by the processes. This is tantamount to presenting instances of metrics for which no stable tree exists. Fig 2 shows one such example with a 3-partition of the nodes of a complete graph of 4

nodes (called graph $\mathcal{K}$). The set of nodes V is partitioned into V_1, V_2, and V_3 such that $V_1 = \{x\}$, $V_2 = \{y\}$, and $V_3 = \{r, z\}$. The metric w is described in the figure. Let us show that there exists no stable tree here: assume there exists a stable tree T. Now, if edge (x, r) is in T, this implies that edge (z, x) is also in T (otherwise T would not be stable). Now consider node y. The edge (y, x) may not be in T (indeed, $\sum_{e \in T_{y \to r}} w_i(e) = 6 \geq w_i((r, y)) = 3$). The edge (y, z) may not be in T (indeed, $\sum_{e \in T_{y \to r}} w_i(e) = 7 \geq w_i((r, y)) = 3$). Thus, the edge (y, r) must be in T. This contradicts the stable property of T because at node x, we would have $\sum_{e \in T_{x \to r}} w_i(e) = 3 \geq w_i((x, y)) \sum_{e \in T_{y \to r}} w_i(e) = 2$. The same argument can be repeated starting from y and from z, so there exists no stable tree for this instance of w. Of course, this example can be generalized to an arbitrary $p \geq 3$.

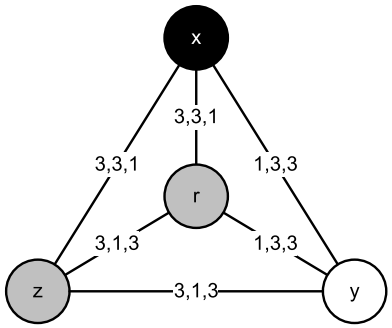

Fig. 2. The complete graph $\mathcal{K}$ with 4 nodes

Clearly, there is no point in searching for a stabilizing solution, because no such solution can be found. Deciding whether there exists an equilibrium is an NP-complete problem.

3. RESULTS

This section summarizes our main results about selfish stabilization. We first discuss results about the existence of equilibria, and subsequently present algorithms for solving the selfish stabilization problem in specific cases.

3.1 Existence of equilibrium

In this section, we discuss the existence of an equilibrium (*i.e.* a stable tree T for a particular metric w) depending on the number of conflicting interests (*i.e.* the number p in the partition of nodes of V into p sets $V_1, V_2, \ldots, V_p$). Let us first notice that the special case of $p = 1$ is the usual problem of the shortest path spanning tree with non zero edge costs, so an equilibrium always exists. We first prove that when $p \geq 3$, determining whether there exists an equilibrium for a particular metric w is an NP-complete problem.

THEOREM 3.1. *Let $G = (V, E)$ be a graph and r a node of V. Let $\mathcal{P}$ be a partition of V with $p \geq 3$ elements $V = \cup_{i=1}^{p} V_i$. Let w be a function on $E \to \mathbb{N}^{*p}$. The problem of deciding the existence of a stable spanning tree T for metric w is NP-complete.*

PROOF. It is easy to show that this problem is in NP. Indeed, given a particular tree T, checking whether T is stable for metric w can be done in polynomial time by checking if every node is stable in T for metric w.

Then, we construct a polynomial transformation (almost similar to transformation described in [10]) with the NP-complete problem 3-SAT defined as follows:

. **Instance:** Collection $C = \{c_1, c_2, \ldots c_m\}$ of clauses on a finite set U of variables such that $|c_i| = 3$ for $1 \leq i \leq m$.

. **Question:** Is there a truth assignment for U that satisfies all the clauses in C?

We now present the transformation $\mathcal{R} : (C, U) \rightarrow (G, r, \mathcal{P}, w)$:

(1) $V \leftarrow \{r\}$; $V_1 \leftarrow \emptyset$; $V_2 =\leftarrow \emptyset$; $V_3 \leftarrow \{r\}$; $E \leftarrow \emptyset$;
(2) for each variable $v \in U$ do:
   (a) $V \leftarrow V \cup \{v, \overline{v}\}$;$V_1 \leftarrow V_1 \cup \{v\}$;$V_2 \leftarrow V_2 \cup \{\overline{v}\}$;
   (b) $E \leftarrow E \cup \{(v, r), (\overline{v}, r), (v, \overline{v})\}$;
   (c) $w(v, r) \leftarrow (3, 1, 1)$; $w(\overline{v}, r) \leftarrow (1, 3, 1)$; $w(v, \overline{v}) \leftarrow (1, 1, 3)$;
(3) for each element $c = (s, t, q) \in C$ do:
   (a) $V \leftarrow V \cup \{c\}$; $V_3 \leftarrow V_3 \cup \{c\}$;
   (b) $E \leftarrow E \cup \{(c, s), (c, t), (c, q)\}$;
   (c) $w(c, s) \leftarrow (3, 3, 1)$; $w(c, t) \leftarrow (3, 3, 1)$; $w(c, r) \leftarrow (3, 3, 1)$;
   (d) a copy of graph $\mathcal{K}$ drawn in Fig. 2 whose nodes are named $\{c.x, c.y, c.z, c.r\}$ is a subgraph of G.
   (e) $E \leftarrow E \cup \{(c.r, r), (c, c.z)\}$;
   (f) $w(c.r, r) \leftarrow (3, 3, 3)$; $w(c, c.z) \leftarrow (1, 1, 3)$;
(4) $\mathcal{P} \leftarrow (V_1, V_2, V_3)$
(5) Return $(G, r, \mathcal{P}, w)$:

Figure 3 presents a partial example of this transformation. It is straightforward to verify that this transformation can be done in polynomial time in the size of the instance. We now prove that there exists a particular assignment t that satisfies all clauses of C if and only if there exists a stable tree T in G for metric w.

First, assume that there exists an assignment t that satisfies all clauses of C. We construct from t a tree T that is stable in G for metric w. For each variable $v \in U$, $(\overline{v}, v) \in T$. If $t(v) = true$, then $(r, v) \in T$ else $(t(v) = false)$, $(r, \overline{v}) \in T$. Nodes $\overline{v}$ and v are stable in T for w (since all other paths from v (or $\overline{v}$) to r are of weight strictly greater to 4). Moreover, for every $c \in C$, there exists at least one of those three elements v such that $t(v) = true$. T is constructed such that $(v, c) \in T$ and $(c, c.z) \in T$. Then, $(c.r, r) \in T$, $(c.y, c.r) \in T$, and $(c.x, c.y) \in T$. T is a stable tree in V for metric w.

Second, assume that there exists a stable tree T in G for metric w. From this stable tree T, we construct an assignment t that satisfies all clauses of C. We consider an element $c = (v, v', v'') \in C$.

Let us first notice that the shortest path according to w_3 in G from $c.r$ to r is $(c.r, r)$. This permits to deduce that for every $c \in C$, $(c.r, r) \in T$. Assume now that the path from c to r in T includes at least one of the nodes $c.x$, $c.y$, or $c.z$. Then, by construction of G, to cover nodes $c.x, c.y, c.z, c.r$, the spanning tree T

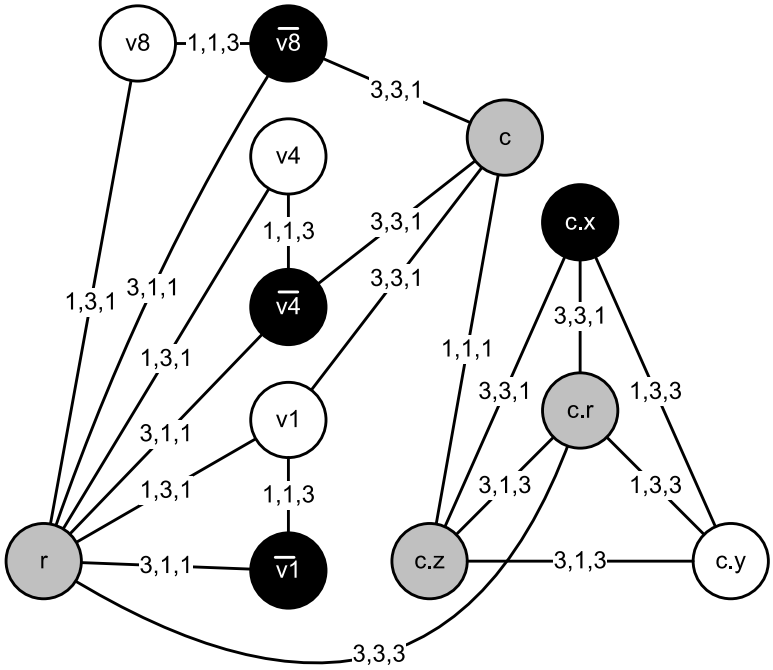

Fig. 3. Example of the transformation with a clause $c = (\overline{v_1}, v_4, \overline{v_8})$

must use a subset of the edges of the complete graph induced by the set of nodes $\{c.x, c.y, c.z, c.r\}$. This brings us back to the case of Figure 2 where there exists no stable spanning tree. This contradicts the fact that the path from c to r in T includes at least one of the three nodes $\{c.x, c.y, c.z\}$. Thus, this implies that:

(1) there exists a literal of clause c such that in graph G, the path from c to r in T includes one of the nodes in $\{v, v', v''\}$.
(2) $(c.y, c.r) \in T$ and $(c.y, c.x) \in T$

Without loss of generality, assume the path from c to r in T includes v. This path does not include $\overline{v}$ (otherwise, $\{c.z, c, v, \overline{v}, r\}$ would have a weight of 8 whereas $\{c.z, c.r, r\}$ would have a weight of 6). In this configuration, node $c.z$ is not stable in T. Thus, $(v, r) \in T$ and $(v, \overline{v}) \in T$. We now present the assignment $t : U \rightarrow \{true, false\}$, such that for each variable $v \in U$, $t(v) = true$ if and only if $(v, u) \in T$. From the previous remark, we deduce that t satisfies all clauses in C. □

We now prove that when the set of nodes is partitioned in two ($p = 2$), there always exists at least one equilibrium. Before proving that, we give a property of stable tree T in G.

LEMMA 3.2. *Let $G = (V, E)$ be a graph and r a node of V. Let $\mathcal{P}$ be a partition of V with 2 elements V_1 and V_2. Let w be a function on $E \rightarrow \mathbb{N}^{*2}$. Let v be a node of V_i (with $i \in \{1, 2\}$) such that there exists a shortest path between v and r using metric w_i that only contains nodes in V_i, then every stable tree T in G with metric w_i contains the shortest path between v and r.*

PROOF. Let $pcc = \{r, v_1, \ldots, v_\ell\}$ be a shortest path between v and r in metric w_i, solely composed of nodes of V_i. We prove this lemma by induction on parameter ℓ.

—If $\ell = 1$, then $w_i(pcc) = w_i(r, v_1)$. Let T be a stable tree such that $(r, v_1) \notin T$. As v_1 is stable in T, we have:

$$\forall z \in \Gamma_G(v_1), w_i(T_{v_1 \to r}) \leq w_i((v_1, z)) + w_i(T_{z \to r})$$

As $r \in \Gamma_G(v_1)$, we have:

$$w_i(T_{v_1 \to r}) \leq w_i(u, v_1) \leq w_i(pcc)$$

Thus, T connects r to v_1 using a shortest path according to metric w_i.

—Suppose now that the lemma is true for any j such that $0 < i < \ell$. Let T be a stable tree for metric w_i. By induction hypothesis, T connects u to $v_{\ell-1}$ through a shortest path according to metric w_i. As v_ℓ is stable in T, we have:

$$\forall z \in \Gamma_G(v_\ell), w_i(T_{v_\ell \to r}) \leq w_i((v_\ell, z)) + w_i(T_{z \to r})$$

As $v_{\ell-1} \in \Gamma_G(v_\ell)$, we have

$$w_i(T_{v_\ell \to r}) \leq w_i((v_{\ell-1}, v_\ell)) + w_i(T_{v_{\ell-1} \to r})$$

Thus, T connects u to v_ℓ through a shortest path according to metric w_i.

□

THEOREM 3.3. *Let $G = (V, E)$ be a graph and r a node of V. Let $\mathcal{P}$ be a partition of V with 2 elements V_1 and V_2, and such that V_1 contains exactly one node x. Let w be a function on $E \to \mathbb{N}^{*2}$. There always exists a stable tree T in G for metric w.*

PROOF. If $x = r$, we fall back in the known case of the construction of a shortest path spanning tree with metric w_2. In the sequel, we assume that $x \neq r$. The theorem is proved by induction on the degree d of x.

—If $d = 1$, every shortest path tree using metric w_2 is a stable tree since x has exactly one incident edge.

—Suppose now that the theorem is true for any graph where the degree of x is (strictly) lower than d. Let $z_1, \ldots, z_d$ be the neighbors of x. Let pcc_i be a shortest path between x and r including z_i according to metric w_2. For simplicity, we assume neighbors are sorted by increasing values of $w_2(pcc_i)$. Let us now consider the graph G_1 that is constructed in the following way:
—$V(G_1) = V(G)$ and $E(G_1) = E(G) \setminus \{(x, z_1)\}$
—$\forall e \in E(G_1)$, $w'(e) = w(e)$
By induction hypothesis, there exists a stable tree T^1 in G_1 according to metric w'. Suppose that T^1 is not stable in G according to metric w. By definition of w' and G_1, only x may not be stable. Thus, we have:

$$w_1((x, z_1)) + w_1(T^1_{z_1 \to r}) \leq w_1(T^1_{x \to r}) \quad (1)$$

Consider T that is a copy of T^1 except that the parent of x is z_1. Now, x is stable in T. We now prove that T is stable in G according to metric w. Let

$Y = V \cap \{y : x \in T^1_{y\to r}\}$. First, every node in $V\backslash Y$ are stable. Now consider a node $y \in V_2 \cap Y$. Let t be the neighbor of y that belong to the path $T^1_{y\to r}$. Now, y is stable in G_1 for metric w' (since $x \neq y$), so we get:

$$\forall s \in \Gamma(y)\backslash\{t\}, w_2(T^1_{y\to r}) \leq w_2(T^1_{s\to r}) + w_2((s,y)) \tag{2}$$

By definition, we have:

$$w_2(T_{y\to r}) = w_2(T^1_{y\to x}) + w_2((x,z_1)) + w_2(T^1_{z_1\to r}) \tag{3}$$

The path $T^1_{z_1\to r}$ connecting r to z_1 and belonging to T^1 is a shortest path according to metric w_2 since there exists a shortest path from r to w_1 according to metric w_2 whose nodes all belong to V_2 (see Lemma 3.2). Then, by definition of pcc_1 which is a shortest path from x to r through z_1, we have:

$$w_2(T^1_{z_1\to r}) + w_2((x,z_1)) = w_2(pcc_1) \tag{4}$$

Combining Equations 3 and 4, we obtain:

$$w_2(T_{w\to r}) = w_2(T^1_{w\to x}) + w_2(pcc_1) \tag{5}$$

By definition, $\forall z_i \in \Gamma_G(x)$, we have $w_2(pcc_i) \geq w_2(pcc_1)$. Combining the previous remark and Equation 5 gives:

$$w_2(T_{y\to r}) = w_2(T^1_{y\to x}) + w_2(pcc_1) \leq w_2(T^1_{y\to x}) + w_2(pcc_i) \tag{6}$$

$$w_2(T_{y\to r}) \leq w_2(T^1_{y\to x}) + w_2(T^1_{x\to r}) \tag{7}$$

$$w_2(T_{y\to r}) \leq w_2(T^1_{y\to r}) \tag{8}$$

Combining Equations 2 and 8, we deduce that y is stable in tree T for metric w. Thus, T is stable in G for metric w. This concludes the proof.

□

THEOREM 3.4. *Let $G = (V, E)$ be a graph and r a node of V. Let $\mathcal{P}$ be a partition of V with 2 elements V_1 and V_2, and such that V_1 contains α nodes $x_1, x_2, \ldots x_\alpha$. Let w be a function on $E \to \mathbb{N}^{*2}$. There always exists a stable tree T in G for metric w.*

PROOF. Without loss of generality, we assume that $r \in V_2$. The theorem is proved by induction on α. The case when $\alpha = 1$ corresponds to Theorem 3.3. Suppose now that the theorem is true for any value (strictly) lower than α.

Let $pcc(x_i)$ be the shortest path between x_i and r according to metric w_2. For simplicity, nodes $x_1, \ldots, x_\alpha$ are sorted with increasing values of $w_2(pcc(x_i))$. We now consider node x_1. By definition, $pcc(x_1)$ does not contain any node belonging to V_1. Node x_1 has d neighbors in G denoted by $z^1_1, \ldots, z^1_d$. Again, we prove by induction on d that there exists a stable tree T in G for metric w.

—If $d = 1$, it is sufficient to compute the stable tree T in $G \subset \{x_1\}$ using metric w (there exists such a tree by induction hypothesis). Then, the edge incident to x_1 is added to T. Such a tree is stable in G for metric w.

—Suppose now that the theorem is true for any graph where x_1 has degree (strictly) less than d. Let $z_1^1, \ldots, z_d^1$ be the neighbors of x_1. Let pcc_i the shortest path between x_1 and r through z_i^1 according to metric w_2. For simplicity, the neighbors are sorted with increasing values of $w_2(pcc_i)$. Consider now the graph G_1 that is constructed in the following way:
—$V(G_1) = V(G)$ and $E(G_1) = E(G) \setminus \{(x_1, z_1^1)\}$
—$\forall e \in E(G_1)$, $w'(e) = w(e)$
By induction hypothesis, there exists a stable tree T^1 in G_1 according to metric w'. Suppose T^1 is not stable in G according to metric w. By definition of w' and G_1, only node x_1 may not be stable. Thus:

$$w_1((x_1, z_1^1)) + \sum_{e \in T^1_{z_1^1 \to r}} w_1(e) \leq \sum_{e \in T^1_{x_1 \to r}} w_1(e) \tag{9}$$

Consider T, that is a copy of T^1 except that the parent of x_1 is z_1^1. Now, x_1 is stable in T. We now prove that T is stable in G with metric w. Let $Y = V \cap \{y : x_1 \in T^1_{y \to r}\}$. First, all nodes in $V \setminus Y$ are stable. Secondly we focus on somes nodes $y \in Y$ where $y \in V_i$ with $i \in \{1, 2\}$. Let t be the neighbor of y such that t belongs to path $T^1_{y \to r}$. Since y is stable in G_1 according to metric w' ($x_1 \neq y$), we have:

$$\forall s \in \Gamma(y) \setminus \{t\}, w_i(T^1_{y \to r}) \leq w_i(T^1_{s \to r}) + w_i((s, y)) \tag{10}$$

Assume that $y \in V_1$. We have $w_1(T_{y \to r}) = w_1(T^1_{y \to x_1}) + w_1(T_{x_1 \to r})$ and we obtain $w_1(T_{y \to r}) < w_1(T^1_{y \to r})$ by Equation 9. By combining the previous remark and Equation 10 we can deduce that

$$\forall s \in \Gamma(y), w_1(T_{y \to r}) \leq w_1(T_{s \to r}) + w_1((s, y))$$

So, all vertices in $Y \cap V_1$ are stable in tree T for metric w.
Assume that $y \in V_2$. By definition of y, we have:

$$w_2(T_{y \to r}) = w_2(T^1_{y \to x_1}) + w_2((x_1, z_1^1)) + w_2(T^1_{z_1^1 \to r}) \tag{11}$$

The path $T^1_{z_1^1 \to r}$ that connects r to z_1^1 belonging to T^1 is a shortest path for metric w_2 since by definition of x_1, there exists a shortest path from r to z_1^1 according to metric w_2 whose all nodes belong to the same set V_2 (see Lemma 3.2). Thus, by definition of pcc_1 which is a shortest path from x_1 to r through z_1^1, we have:

$$w_2(T^1_{z_1^1 \to r}) + w_2((x_1, z_1^1)) = w_2(pcc_1) \tag{12}$$

Combining Equations 11 and 12, we obtain:

$$w_2(T_{y \to r}) = w_2(T^1_{y \to x_1}) + w_2(pcc_1) \tag{13}$$

By definition $\forall z_i \in \Gamma_G(x)$, we have $w_2(pcc_i) \geq w_2(pcc_1)$. Combining the previous remark with Equation 13, we obtain:

$$w_2(T_{y \to r}) = w_2(T^1_{y \to x_1}) + w(pcc_1) \leq w_2(T^1_{y \to x_1}) + w_2(pcc_i) \tag{14}$$

$$w_2(T_{y\to r}) \leq w_2(T^1_{y\to x_1}) + w_2(T^1_{x_1\to r}) \quad (15)$$

$$w_2(T_{y\to r}) \leq w_2(T^1_{y\to r}) \quad (16)$$

Combining Equations 10 and 16, we deduce that y is stable in tree T for metric w. Thus, T is stable in G for metric w. This concludes the proof.

□

3.2 Stabilizing algorithm

Our goal in this section is to design distributed algorithms that construct a stable tree T for a particular metric w. Since the system is distributed, nodes are only aware of their neighborhood.

We assume that each node i is aware of $\Gamma(i)$, the set of its neighbors (excluding i itself). Similarly, each node i is aware of the cost of each of its adjacent edges $e = (i, j) : j \in \Gamma(i)$. The cost of an edge e is a vector

$$\omega(e) = (\omega_1(e), \omega_2(e), \omega_3(e), \ldots, \omega_p(e))$$

where $\omega_k(e)$ denotes the cost of the edge e for a node of color k. Note that by notation all local variables are represented by greek letters and that all global variables is represented by normal letters. For example, in previous section, w is used by denoted the function of edge cost. Now ω is the cost of an edge for an edge e.

Also, i maintains two variables: $\pi(i)$ and $\lambda(i)$ (commonly called the *parent* and the *label* or *distance* variables). By definition, for the root node r, $\pi(r)$ is non-existent. Every other node picks a neighboring node as its parent (*i.e.* the domain of the $\pi(i)$ variable is $\Gamma(i)$. The label for each node is a vector

$$\lambda(i) = (\lambda_1(i), \lambda_2(i), \lambda_3(i), \ldots, \lambda_p(i))$$

where $\lambda_k(i)$ denotes the distance for a node of the k^{th} color from node i to the root. By definition, $\lambda(r) = (0, 0, \ldots, 0)$.

In networks with selfish peers, the costs influencing decision-making are often of commercial nature, and are thus kept private to the participating nodes. To accommodate this feature in our framework, we assume that for each node, the value of each component of λ and the weights of the edges incident on it ω will be stored in an encrypted form. All the same color nodes share a common secret key. Thus, no node can access the component of the costs of a different color from a neighboring node or link to decide its course of action. However every node can securely extract the component of the variables corresponding to its own color. This authentication mechanism preserves fairness of the game and prevents possible foul play by deliberately tampering the variables of the nodes of opposing color. We will designate the encrypted version of a variable x by $\hat{x}$. For the sake of simplicity, we assume that [1]

$$\hat{x} + \hat{y} = \widehat{x + y}$$

We first propose a greedy algorithm for systems that contain nodes of two different colors 1 and 2 (*i.e.* $p = 2$): each node i of color k will select a parent $\pi(i)$

[1] Homomorphic encryptions like Pallier's scheme [17] satisfy this property.

that minimizes $\lambda_k(i)$, regardless of what happens to $\lambda_j(i)$ $(j \neq k)$. For the sake of brevity, we define the following:

Conditions

$LabelOK(i) \equiv \quad \widehat{\lambda(i)} = \widehat{\lambda(\pi(i))} + \widehat{\omega(i,\pi(i))}$
$ParentOK(i)_{i \in V_k} \equiv \lambda_k(\pi(i)) + \omega_k(i,p(i)) = min\{\lambda_k(j) + \omega_k(i,j) : j \in \Gamma(i)\}$

Actions

$FixLabel(i) \equiv \quad \widehat{\lambda(i)} := \widehat{\lambda(\pi(i))} + \widehat{\omega(i,\pi(i))}$
$FixParent(i)_{i \in V_k} \equiv$ select $\pi(i)$: $\lambda_k(\pi(i)) + \omega_k(i,\pi(i)) = min\{\lambda_k(j) + \omega_k(i,j) : j \in \Gamma(i)\}$

The proposed algorithm (denoted afterward as Algorithm Greedy) has a two guarded actions. The root r does not execute any action. The other nodes execute local adjustment of their labels (to make it consistent with their parent's) and of their parent neighbor (in order to locally minimize the cost of the metric for the node color). Also, the label adjustment action has higher priority than the parent adjustment action. The actions for node $i \neq r$ are described in the following algorithm:

Program for process i
{ *Fix label* }
$\neg\ LabelOK(i) \longrightarrow FixLabel(i)$;
{ *Fix parent* }
$LabelOK(i) \wedge \neg\ ParentOK(i) \longrightarrow FixParent(i)$;

3.3 Proof of correctness

THEOREM 3.5 CORRECTNESS. *Let $G = (V, E)$ be a graph and r a node of V. Let $\mathcal{P}$ be a partition of V with p elements. Let w be a function on $E \rightarrow \mathbb{N}^{*p}$. If no node has enabled rules, the structure induced by the π variables of each node executing Algorithm Greedy induces a stable tree T rooted at r for metric w.*

PROOF. Let us first prove that the induced structure is a tree. Due to the fact that the π variables are always pointing to a neighbor (for every node except r), the structure is either a tree or a tree with disconnected circuits. Assume for the purpose of contradiction that there exists at lest one circuit. As w only has costs in $\mathbb{N}^*$, all edge costs are (strictly) positive. As no rule is enabled, this implies that all labels are OK, and thus each label is strictly greater as the label of the parent. Due to the well founded property of the "lower than" relation on (positive) integers, the existence of such a circuit is impossible. Hence, the induced structure is a tree.

Now, we prove that the obtained tree T is stable for metric w. As no rules are enabled, this means that the λ variables are all properly computed from the local metric information ω, that can not be corrupted. By induction on the distance (in number of hops) to the root r, this means that the λ variables contain the correct values for the cost of every path towards to root according to metric w (that is known from the ω to every node). Now, the parent changing rules are also not enabled. This implies that the π variables are actually selecting the best neighbor according to the color k of node $i \in V_k$. This best neighbor being selected according

to the metric w, we conclude that the tree T that is induced by the π variables is a stable tree for metric w. □

When there exists a single class of nodes in the system, the problem reduces to the problem of constructing a shortest path tree to a particular node r. There exist several self-stabilizing solutions to this problem, among which [7; 12]. In the case there are exactly two classes of nodes, what we guarantee for Algorithm Greedy is *weak stabilization* [8]. Weak stabilization guarantees that starting from an arbitrary configuration, there exists *at least one computation* that leads the system to an equilibrium configuration. Weak stabilization is possible when a central scheduler (or daemon) randomly picks a process with an eligible guard, and schedules its action.

LEMMA 3.6. *Let $G = (V, E)$ be a graph and r a node of V. Let $\mathcal{P}$ be a partition of V with 2 elements V_1 and V_2, and such that V_1 contains α nodes $x_1, x_2, \ldots x_\alpha$. Let w be a function on $E \to \mathbb{N}^{*2}$. Starting from a configuration where the π variables induce a single spanning tree, there exists an execution of algorithm Greedy that reaches a terminal configuration.*

PROOF. Assume we start from a configuration where the π variables of all nodes induce a single spanning tree. By the hypothesis that $p = 2$, we know there exists at least one stable tree T. Now consider the execution in which all nodes that have their label fixing rule enabled execute it, and that nodes that have their parent fixing rule don't execute it. Since the π variables induce a tree, only a fixed number of label fixing rules can be executed. If in this configuration, no node has a rule to execute, the configuration is terminal. Otherwise, this means that there exists at least one node that is willing to change its parent. When one node changes its parent, there exists an execution in which all nodes in its subtree update their label and no other parent fixing rule is executed. Using this scheme, we always guarantee that a single tree rooted at r induced by the π variables exists in the system.

Now we mimic the construction of the stable tree T in the proofs of Theorem 3.3 and 3.4, reasoning by induction on the number of nodes in V_1, and then by induction on the degree of each node in V_1. Anytime an edge is removed then replaced to move toward a stable tree, this correspond to a parent adjustment at some node in our algorithm. Since all label adjustments are made between any two parent adjustments, we conclude that the system eventually reaches a stable tree T considering the π variables. Now, when the tree induced by th eπ variables is stable, we consider the execution in which all labels are fixed. By the acyclic nature of the tree, there is a finite number of such moves. □

LEMMA 3.7. *Let $G = (V, E)$ be a graph and r a node of V. Let $\mathcal{P}$ be a partition of V with 2 elements V_1 and V_2, and such that V_1 contains α nodes $x_1, x_2, \ldots x_\alpha$. Let w be a function on $E \to \mathbb{N}^{*2}$. Starting from an arbitrary configuration, there exists an execution of Algorithm Greedy that reaches a configuration where the π variables induce a single spanning tree.*

PROOF. In an arbitrary initial configuration, the π variables may induce either a tree, or a tree and a set of circuits. We consider the following execution: first all labels of nodes that are in the tree component and that need to be fixed are fixed

by executing the label adjustment rule. Then, for every circuit, there exists at least one node that can adjust its label (due to the well foundedness of the "less than" relation on positive integers). Then, it turns out that for every cycle, there exists an execution that makes the label grow unboundedly. We consider the execution that reaches a configuration where every node in a circuit has a label so high that it is greater than the highest label of any node in the tree component plus the weight of the edge for the node color according to metric w. Also, there is at least one such circuit such that at least one node of the circuit is also a neighbor of one node in the tree component. Now, consider one node i that is a neighbor of a node in the tree component; this node may have to execute its label adjustment rule (in case it is needed), then it may execute (immediately after) its parent change rule. In this execution, we also consider that after this parent change move, all nodes that were belonging to i circuit execute their label adjustment rule if needed. This execution leads to a configuration where there is one less circuit component. The process can be repeated, and by induction on the number of initial circuits, the result is obtained. □

THEOREM 3.8 WEAK STABILIZATION. *Let $G = (V, E)$ be a graph and r a node of V. Let $\mathcal{P}$ be a partition of V with 2 elements V_1 and V_2, and such that V_1 contains α nodes $x_1, x_2, \ldots x_\alpha$. Let w be a function on $E \rightarrow \mathbb{N}^{*2}$. Algorithm Greedy is weakly self-stabilizing for the stable tree construction problem according to metric w.*

PROOF. Starting from an arbitrary initial configuration, there exists an execution that reaches a configuration where a tree is induced by the π variables (Lemma 3.7). From such a configuration, there exists an execution that reaches a terminal configuration (Lemma 3.6). Every terminal configuration denotes a stable tree according to metric w (Theorem 3.5). □

Weak stabilization is in contrast with traditional stabilization (call it *strong stabilization* that allows the daemon to pick an arbitrary process with an enabled guard, and schedules its action. Strong stabilization requires that *all computations* starting from an arbitrary configuration lead the system to an equilibrium configuration. The execution that is presented in Figure 4 shows that the Greedy protocol is not self-stabilizing, as there exists some executions that never terminate, even considering the central daemon model.

Of course, our (weakly) self-stabilizing solution does not precule the existence of (strongly) self-stabilizing solutions, but proving the existence of such solution is an open question.

3.4 Alternative strategies

If there exists a set of strategies (synonymous with algorithms) with the property that no process can lower its cost by changing its strategy while the other processes keep their strategies unchanged, then that set of strategies and the corresponding costs constitute the *Nash Equilibrium*.

To prove that the stable configuration reflects a Nash equilibrium, we need to consider various strategies that can be adopted by the processes to lower their costs. The Greedy algorithm we proposed uses a *greedy strategy*, (call it *Strategy A*) but

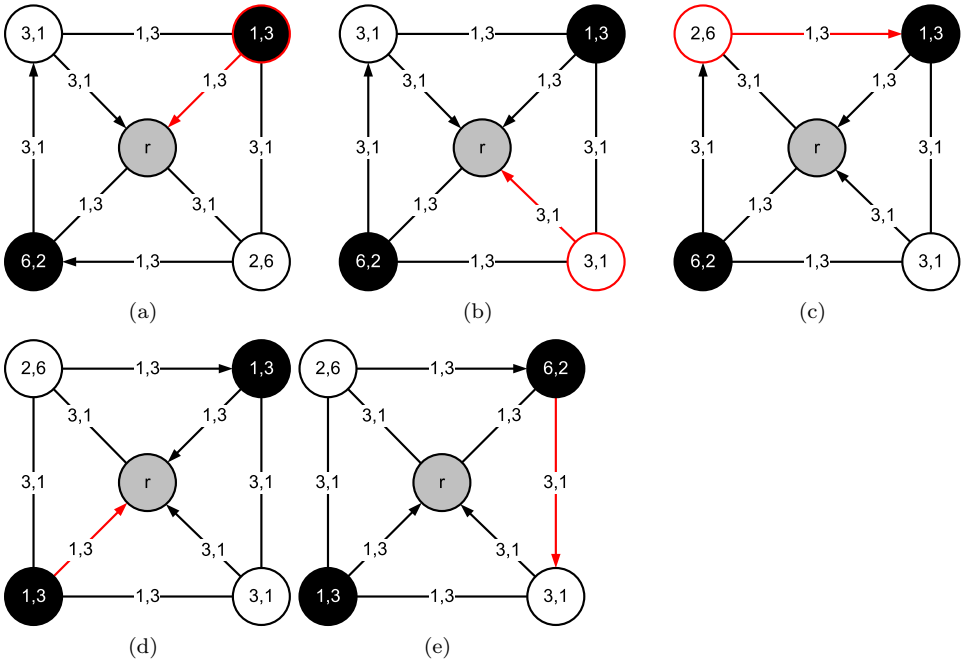

Fig. 4. Example execution with the sequential (a.k.a. central) daemon. Configurations (a) and (e) are symmetric, so repeating the process leads to Configuration (a) and the system is not stabilizing.

it is, by no means, the only possible strategy. Let us examine a second strategy for cost minimization by the individual processes. It is an *altruistic strategy*: each node picks a parent that lowers the communication cost of the nodes of the opposite color (call it *Strategy B*). As a result, processes in V_1 will help lower the cost of the processes in V_2, and vice versa. To implement Strategy B, we modify the definition of $ParentOK$ and $FixParent$ as follows (considering $\overline{\lambda}(i)$ denotes the label for the other color than i, and that $\overline{\omega}(i,k)$ denotes the weight for the other color than i) [2]:

$$ParentOK(i) \equiv \quad \pi(i) = j : \overline{\lambda}(j) + \overline{\omega}(i,j) = min\{\overline{\lambda}(k) + \overline{\omega}(i,k) : k \in \Gamma(i)\}$$

$$FixParent(i) \equiv \quad \text{select } \pi(i) := j : \overline{\lambda}(j) + \overline{\omega}(i,j) = min\{\overline{\lambda}(k) + \overline{\omega}(i,k) : k \in \Gamma(i)\}$$

Once these are appropriately defined, the main algorithm remains unchanged. Using the same line of arguments, we can show that this algorithm also stabilizes the system, but to a different configuration. This leads to the following observation:

Observation 1. *Using Strategy B, the system of processes (weakly) stabilizes*

[2]This apparently weakens the encryption mechanism since it requires $x > y \Rightarrow \hat{x} > \hat{y}$. However, using the altruistic protocol, processes in V_1 lower their cost by helping the processes in V_2 and vice versa, and this is more conducive to building a trust relationship. So we will disregard the encryption symbol.

to an equilibrium configuration, and the edges connecting the processes with their parents form a spanning tree.

The observation trivially follows from Theorem 3.8 if we swap the costs of the nodes in V_1 and V_2 for each edge.

The Cost of Equilibrium. A natural component of such an exercise is to analyze the quality of the equilibrium configuration: How bad is the cost of this configuration in comparison with the "optimal" configuration? For the nodes of a given color, define the cost of a configuration as the sum of weights of all the tree edges for that color. Define the *optimal* cost as the cost of the tree when all nodes are of the same color. The issue is: By what extent will it increase if some of the nodes belong to a different color? Here is an upper bound. Let $e_{max} = max\{\omega(e), \overline{\omega}(e) : e \in E\}$ and $e_{min} = min\{\omega(e), \overline{\omega}(e) : e \in E\}$. Then the following theorem holds.

THEOREM 3.9. *For any set of processes of a given color, the ratio of the cost of the equilibrium configuration to the cost of the optimal configuration is bounded from above by* $\frac{e_{max}}{e_{min}}$.

PROOF. A tree with N nodes has $(N-1)$ edges, so the cost of the optimal configuration has a lower bound of $(N-1).e_{min}$. To determine the maximum possible weight of the tree in an equilibrium configuration under any of the algorithms A or B (or a mix of the two), think of an adversary that can switch the color of zero or more processes so that each node chooses the edge with largest weight as its link to its parent node. The cost of the resulting configuration is bounded from above by $(N-1).e_{max}$. The ratio of the two costs will not exceed

$$\frac{e_{max}}{e_{min}}.$$

□

This is a loose upper bound. In general, when the number of processes in V_1 is much larger than the number of processes in V_2, Strategy A will lead to a lower cost for the processes in V_1, and Strategy B will lead to a lower cost for the processes in V_2. This is quite intuitive, since in Strategy A, each step by the majority (*i.e.* in V_1) processes helps lower their own cost at the expense of the competitors' cost, whereas in Strategy B, each step by the majority processes lowers the cost of the competitors at the expense of their own cost.

Simulations support this observation, although the costs do not necessarily decrease (or increase) monotonically with the number of processes switching strategies. The topology and the cost distribution play deciding roles. That said, based on the knowledge acquired during the progress of the communication, processes may be tempted to use different strategies. However, we will demonstrate that the system is robust enough to guarantee convergence to a Nash equilibrium, where all processes choose Strategy A, and no process can unilaterally lower its cost of communication with the root node.

OBSERVATION 2. *The cost of the processes in V_1 (resp. V_2) will be minimum when they use Strategy A while the processes in V_2 (resp. V_1) use strategy B.*

Viewed from the perspective of the processes in V_1, the validity of the above observation is based on the fact that that every node picks the best edge for the

processes in V_1, so the algorithm reduces to the classical stabilizing shortest path algorithm (*e.g.* [12; 7]) for the nodes in V_1.

THEOREM 3.10. *For a given graph $G = (V, E)$ with a given composition of the processes in V, and the set of strategies (A, B) the equilibrium configuration is unique, and it reflects the Nash equilibrium.*

PROOF. Assume that using whatever strategy the processes choose, the system of processes stabilizes to some configuration that determines the payoffs for the processes in the V_1 and V_2 sets. Now consider three different cases:

(1) Assume that all processes use Strategy B. Observe that one or more processes of a certain group will switch to Strategy A, since this will lower their cost (Theorem 3). However the other group might apprehend this, they will also switch from Strategy B to Strategy A.
(2) Assume that the processes in V_1 use Strategy B, while the processes in V_2 use Strategy A. However, altruism does not pay off unless everyone is altruistic. Since the processes in V_1 do not know what strategy the processes in V_2 are using, they will switch to Strategy A, and their cost will go down.
(3) Assume that all processes use Strategy A. Now no process will be motivated to switch to Strategy B, since such a switch will imply lowering the cost of the other group even if it increases the cost of its own group. Thus this is a stable configuration.

Thus, regardless of the initial strategies chosen by the processes in V_1 and V_2, all processes will eventually switch to Strategy A, and regardless of the initial values of L and p, the system will stabilize in a bounded number of steps. Furthermore, since no process can unilaterally lower its cost by switching to a different strategy, the stable configuration will reflect a Nash equilibrium. □

Note. Both strategies (A and B) can be further optimized as follows. Consider A first. There may be cases in which a node $i \in V_1$ finds multiple neighbors j satisfying the condition of being a "best parent". Instead of arbitrarily choosing one such node, i will choose a $\pi(i) = j$ for which the cost of the opposite color component of $\lambda(i)$ is the lowest. A similar step can be taken by the nodes in V_2 too. The interesting aspect of this exercise is that not only does the network state stabilize to a desirable configuration, but the strategies stabilize too, in as much as regardless of the starting strategies, all processes end up using the same final strategy. This does not rule out the invention of new strategies beyond what has been considered for this exercise.

4. CONCLUSION

Selfish stabilization reduces to classical stabilization when the private goals of the constituent processes do not conflict. The following issues are relevant about the approach taken in this paper:

The first is the separation of *cooperation* and *competition*. Assume that processes first cooperate to form a spanning tree, then try to optimize it to improve their individual payoffs. In presence of arbitrary initializations, failures, and selfish motives, such segregation of actions is difficult to implement.

The uniqueness of the equilibrium point is another significant issue. For the current problem, under each strategy, the system of processes reaches a unique equilibrium point, and the resulting Nash equilibrium is also unique. If this were not true, then there could be multiple trees, possibly of different costs, where the system of processes could stabilize to, the choice being determined by the schedule and the relative speeds of actions. However, once reaching an equilibrium point, an unhappy (or ambitious) process could deliberately introduce a perturbation (by corrupting a local variable) to possibly reach a different equilibrium point with a better payoff, and jeopardize the common goal. There is no guarantee that this will happen. But the uniqueness of the equilibrium point will prevent the constituent processes from using deliberate perturbation as a strategy to improve payoff, or at least probe the possibility of a better payoff.

Non-compliance to global mandates can have an overall negative impact on the payoffs when the Nash equilibrium corresponds to an inferior equilibrium. One approach can be the development of a payment scheme to reward compliance. Another approach involves detecting cheaters and appropriately penalizing them to force compliance. Quantification of these issues is an open problem, and is a topic of future research.

The paradigm of selfish stabilization can easily be extended in several ways. First, it can easily be extended to systems involving more than two competing groups, in the extreme case, each process caring for itself and no one else. Second, the metric used here (simple additive metric) could be replaced by any strictly monotonic metric, such as those presented in [9; 7]. This would extend those previous results on stabilizing generic routing, since our scheme allows the possibility to use different metrics for different groups of players. This would also subsume previous approaches that investigated a specific metric [4].

REFERENCES

Arora, A., Gouda, M.G.: Closure and Convergence: A foundation of fault-tolerant computing. IEEE Trans. Software Engineering **19**, (1993) 1015–1027.

Chen, N.S., Yu, H.P., Huang, S.T.: A self-stabilizing algorithm for constructing a spanning tree. Information Processing Letters **39** (1991) 147–151.

Cao, H. Ertin, E., Kulathumani, V., Sridharan, M., Arora, A.: Differential games in large-scale sensor-actuator networks. IPSN 2006: 77-84

Cobb, J.A., Gouda, M.G, Musunuri, R.: A stabilizing solution to the stable path problem. Workshop on Self-stabilizing Systems (2003) 169–183.

Dijkstra, E.W.: Self-stabilization in spite of distributed control. Communications of the ACM **17** (1974), 643–644.

Dolev, S.: Self-stabilization. MIT Press (2000).

Ducourthial, B., Tixeuil, S.: Self-stabilization with r-operators. Distributed Computing **14** (2001) 147–162.

Gouda, M.G.: The Theory of weak stabilization. Workshop on self-stabilizing systems (2001).

Gouda, M.G., Schneider, M.: Stabilization of maximal metric trees. Workshop on Self-stabilizing Systems (1999) 10–17.

Griffin, T.G., Shepherd, F.B., Wilfong, G.: The stable paths problem and interdomain routing. IEEE/ACM Transactions on Networking **10** (2002).

Halpern, J.Y.: A computer scientist looks at game theory. Invited talk at *Games 2000*. Available from http://www.econwpa.wustl.edu/listings/0411.html.

Huang, T.C.: A self-stabilizing algorithm for the shortest path problem assuming read/write Atomicity. J. Computer and System Sciences **71** (2005) 70–85.

Keidar, I. Melamed, R., Orda, A.: EquiCast: scalable multicast with selfish users. ACM Conference on Principles of Distributed Computing (PODC) 2006: 63-71.

Mavronicolas, M., Papadopoulou, V.G., Philippou, A., Spirakis, P.: A graph-theoretic network security game. First International Workshop in Internet and Network Economics (WINE 2005) 969–978.

Moscibroda, T., Schmid, S., Wattenhofer, R.: On the topology formed by selfish peers. ACM Conference on Principles of Distributed Computing (PODC), Denver, 2006.

Moscibroda, T., Schmid, S., Wattenhofer, R.: When selfish meets evil: Byzantine players in a virus inoculation game. ACM Conference on Principles of Distributed Computing (PODC), Denver, 2006.

Pallier, P.: Public-key cryptosystems based on composite degree residue classes. Eurocrypt (1999).

Osborne, M.J., Rubinstein, A.: A course in game theory. MIT Press (1994).

Roughgarden, T., Tardös E.: How bad is selfish routing? Journal of the ACM **49** (2002) 236–259.

Simon, H.: Models of Bounded Rationality: Volumes 1 and 2. MIT Press (1982)