Theoretical Computer Science 412 (2011) 4325–4335

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Nash equilibria in stabilizing systems

M.G. Gouda [b,a], H.B. Acharya [b,*]

[a] The National Science Foundation, USA
[b] The University of Texas at Austin, USA

ARTICLE INFO

Keywords:
Nash equilibrium
Selfishness
Stabilizing system
Perturbation-proof
Perturbation-prone

ABSTRACT

The objective of this paper is three-fold. First, we specify what it means for a fixed point of a stabilizing distributed system to be a Nash equilibrium. Second, we present methods that can be used to verify whether or not a given fixed point of a given stabilizing distributed system is a Nash equilibrium. Third, we argue that in a stabilizing distributed system, whose fixed points are all Nash equilibria, no process has an incentive to perturb its local state, after the system reaches one fixed point, in order to force the system to reach another fixed point where the perturbing process achieves a better gain. If the fixed points of a stabilizing distributed system are all Nash equilibria, then we refer to the system as perturbation-proof. Otherwise, we refer to the system as perturbation-prone. We identify four natural classes of perturbation-(proof/prone) systems. We present system examples for three of these classes of systems, and show that the fourth class is empty.

1. Introduction

The main objective of this paper is to argue that Nash equilibria, which were introduced as termination criteria of games [1], can be used to discourage malicious perturbations in stabilizing distributed systems. But let us start, from the beginning, by describing how a Nash equilibrium can be used as a termination criterion for a well-known game called the two-prisoner dilemma [2].

Consider a "game" that involves two prisoners: prisoner 0 and prisoner 1. This game ends when each prisoner settles on one strategy, out of the two possible strategies of "stay silent" or "betray other prisoner", that maximizes the value of its gain function.

Each prisoner i has a variable $x.i$ whose value, 0 or 1, is assigned as follows:

$x.i = 0$ if prisoner i selects the "stay silent" strategy
$\quad\quad 1$ if prisoner i selects the "betray other prisoner" strategy

The gain function $g.i$ of prisoner i is defined, based on the values of the two variables $x.0$ and $x.1$, as follows:

$$\begin{aligned} g.i = 1 &\text{ if } x.i = 0 \wedge x.(i+1 \bmod 2) = 0 \\ -1 &\text{ if } x.i = 1 \wedge x.(i+1 \bmod 2) = 1 \\ 2 &\text{ if } x.i = 1 \wedge x.(i+1 \bmod 2) = 0 \\ -2 &\text{ if } x.i = 0 \wedge x.(i+1 \bmod 2) = 1 \end{aligned}$$

* Corresponding author. Tel.: +1 512 961 9466.
E-mail addresses: mgouda@nsf.gov (M.G. Gouda), acharya@cs.utexas.edu (H.B. Acharya).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.11.027

Thus, the value of the gain function of prisoner i is increased (either from 1 to 2, or from -2 to -1) when the value of variable $x.i$ is changed from 0 to 1 while the value of the other variable $x.(i + 1 \bmod 2)$ remains unchanged. In other words, the action of prisoner i to select a strategy, out of the two possible strategies, is as follows:

$$x.i = 0 \rightarrow x.i := 1$$

Note that this action of prisoner i can be executed only when its execution is guaranteed to increase the value of the gain function of prisoner i.

This game of selecting strategies by the two prisoners can start at any global state, for example one where $x.0 = 0 \wedge x.1 = 0$, and then the enabled actions of the two prisoners can be executed, one at a time, until the game reaches the fixed point where $x.0 = 1 \wedge x.1 = 1$ and the game terminates (since neither action can be executed at this fixed point). The fixed point of this game, of selecting strategies, is a Nash equilibrium.

Thus, a Nash equilibrium of a game is a global state of the game where no player can execute an action to change its local state and increase the value of its own gain function.

The subject matter of this paper is to discuss the role of Nash equilibria in stabilizing distributed systems, rather than in games. On the surface, games and stabilizing distributed systems seem similar. On one hand, a game involves several players, and each player is specified by some local variables, some actions, and a gain function. On the other hand, a stabilizing distributed system involves several processes, and each process is specified by some local variables, some actions, and a gain function.

But as one looks deeper, significant differences between games and stabilizing distributed systems become clear:

1. The actions of each player in a game are intended only to increase the value of the gain function of that player, whereas the actions of each process in a system are intended to perform other functions (e.g. construct a spanning tree, elect a leader, or reach consensus) and may not always increase the value of the gain function of that process.
2. Each fixed point of a game is a Nash equilibrium, whereas some fixed points of a system may not be Nash equilibria. Assume for example that a system has a fixed point s such that if a process i perturbs its local state at s, then the system reaches another fixed point s' where the value of the gain function of i at s' is higher than its value at s. In this case, the fixed point s cannot be considered a Nash equilibrium for this system.
3. The role of Nash equilibria in a game is to signal game termination. By contrast, the role of Nash equilibria in a stabilizing distributed system is to discourage the system processes from maliciously perturbing their local states to force the system into fixed points with higher values of their gain functions.

The notion that each process in a stabilizing distributed system may have a distinct gain function (that the process seeks to maximize during the system execution) has appeared in the pioneering work of [3,4]. This concept, called *selfish stabilization*, allows every process to have actions from multiple different algorithms. At each step, a process chooses which algorithm to execute actions from, then chooses and executes an action from this algorithm. The goal of a process is not just to bring the system to a legitimate configuration, but to a legitimate configuration where its own gain is as high as possible. Unfortunately, the authors provide mostly negative results, and conclude that it is far from trivial to "segregate cooperation from competition" — that is, to let the processes cooperate to reach a legitimate configuration, then compete to improve their individual gain. It seems to be a difficult problem to combine both objectives (stabilization and optimization of individual gains) while designing a system.

In this paper, we develop and present a different approach to the problem of separating cooperation from competition. We note that there are two classes of events in any distributed system — action executions and faults. Action executions occur when a process chooses and executes an action from its action system. Faults are usually taken to occur randomly, and have random effects on the state of the process where they occur. In order to segregate cooperation and competition, we suggest the notion of selfish (as opposed to Byzantine) faults, which we call *perturbations*. In a perturbation, a process changes the value of its local variables, thereby changing the system state. The action systems of all processes in the system are designed to ensure stabilization; the selfish actions of processes, by means of which they attempt to increase their individual gain, are represented by perturbations. The key insight of this paper is, if we assume that the processes causing faults are selfish (rather than Byzantine), it is possible to design systems that are stabilizing even in the presence of perturbations; if the system converges to a fixed point such that no process can single-handedly increase its gain by causing a perturbation, then in the absence of collusion the system will be stable in this state. We define such a state as a Nash equilibrium for the system.

For convenience, we enumerate the differences between the problems addressed in the current paper and those addressed in [3,4].

1. In [3,4], the gain function of each process in a stabilizing distributed system is defined at each state (whether stable or not) of the system. By contrast, in the current paper, the gain function of each process in the system is defined only at the (stable) fixed points of the system.
2. In [3,4], the actions of each process in the system are intended to increase the value of the gain function of that process. In the current paper, the actions of each process are only intended to force the system into a fixed point.

3. In [3,4], once the system reaches a fixed point, no process can perturb its local state. In the current paper, once the system reaches a fixed point, any process can perturb its local state in order to push the system towards another fixed point, provided that the gain function of the process has a higher value at this new fixed point.

2. Stabilizing systems and Nash equilibria

A *distributed system* consists of n processes that communicate through their shared memory, as described below. Each *process* i in a distributed system, where i is in the range $0 \ldots (n-1)$, has a number of local variables and a number of actions. Each *action* of process i is of the following form:

$$\langle guard \rangle \rightarrow \langle statement \rangle$$

where $\langle guard \rangle$ is a boolean expression over the local variables of process i and the local variables of the neighboring processes of process i, and $\langle statement \rangle$ is an assignment statement that reads the local variables of process i and the local variables of the neighboring processes of process i and writes the local variables of process i.

A *local state* of process i in a distributed system is defined by a value for each local variable in process i. A *global state* of a distributed system is defined by a local state of every process in the distributed system.

A *transition* of a distributed system is a pair (s, s') where the following two conditions hold:

1. Both s and s' are global states of the distributed system.
2. There is an action c in some process in the distributed system such that the guard of c is true when the system is in state s and executing the statement of the action c when the system is in state s yields the system in state s'.

A global state s of a distributed system is called a *fixed point* of the system iff the guard of each action in each process in the system is false when the system is in state s.

A *computation* of a distributed system is a sequence $(s.0, s.1, \ldots)$ where the following three conditions hold:

1. Each of the sequence elements $s.0, s.1, \ldots$ is a global state of the distributed system.
2. Each pair of consecutive states $(s.j, s.(j+1))$ in the sequence is a transition of the distributed system.
3. Either the sequence is infinite, or it is finite and its last global state is a fixed point of the system.

A global state s' is said to be *reachable from* a global state s iff the distributed system has a computation, where s is the initial state and s' is a state in the computation.

A distributed system is called *stabilizing* iff every computation of the system is finite. (Thus, each computation of a stabilizing distributed system is guaranteed to end at a fixed point of the system.) The notion of stabilization was first introduced by Dijkstra [10]. An interested reader can find more results about system stabilization in Gouda [12] and Dolev [11].

Consider a stabilizing distributed system that has n processes. A *gain function* $g.i$ for process i in this system is a function that assigns, to the local state of process i and to the local states of the neighboring processes (of process i) when the system is in a fixed point s, an integer value called the value of the gain function $g.i$ at the fixed point s.

Note that the value of a gain function is defined only when the system is at a fixed point. Thus, the gain of a process is not defined, except at fixed points. (This differs from the earlier approach of Ghosh et al. in [4], where the gain of a process is defined at all states.)

Henceforth we adopt the notation $\{g.i\}$ to indicate a set of gain functions that contains exactly one gain function $g.i$ for each process i in the system. Different processes may have different gain functions.

Consider a stabilizing distributed system. Let s be a fixed point of this system, and $\{g.i\}$ be a set of gain functions for this system. The fixed point s is called a *Nash equilibrium* w.r.t. $\{g.i\}$ iff for every process i in the system and for every global state s', that results from perturbing (i.e., changing in any way) the local state of process i starting from state s, the following condition holds:

- The value of the gain function $g.i$ at some fixed point s'', reachable from s', is no more than the value of $g.i$ at the fixed point s.

Equivalently, the fixed point s is not a Nash equilibrium w.r.t. $\{g.i\}$ iff there exists a process i in the system, and there exists a global state s', that results from perturbing the local state of process i starting from state s, such that the following condition holds:

- The value of the gain function $g.i$ at every fixed point s'', reachable from s', is more than the value of $g.i$ at the fixed point s.

The significance of a fixed point of a stabilizing distributed system being a Nash equilibrium w.r.t. $\{g.i\}$ can be explained as follows. Recall that each computation of the system is guaranteed to end at a fixed point since the system is stabilizing. Now assume that the system's computation ends at a fixed point s. If s is a Nash equilibrium w.r.t. $\{g.i\}$, then no process i in the system has an incentive to perturb its local state — any such perturbation may lead the system to a fixed point where the value of $g.i$ is no more than its value at s. On the other hand, if s is not a Nash equilibrium w.r.t. $\{g.i\}$, then at least one

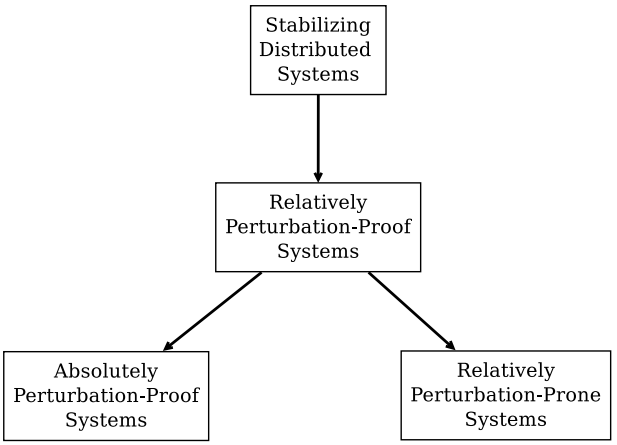

Fig. 1. A taxonomy.

process i in the system has an incentive to perturb its local state in some way, because this perturbation is guaranteed to lead the system to a fixed point where the value of $g.i$ is more than its value at s.

In summary, whereas a Nash equilibrium in a game is an indication that no move by a game player is gainful to this player, a Nash equilibrium in a stabilizing distributed system is an indication that no perturbation of the local state of a system process is gainful to this process. The common feature is that, in a Nash equilibrium, a player or process does not have the power to improve its gain if it only has control over its own actions.

3. Perturbation-proof and -prone systems

In the previous section, we have presented a definition of when a fixed point of a stabilizing distributed system is a Nash equilibrium with respect to a given set $\{g.i\}$ of gain functions. Utilizing this definition of a Nash equilibrium in a stabilizing distributed system, we can classify stabilizing distributed systems into the following four classes:

1. *Relatively perturbation-proof systems*:
   A stabilizing distributed system is called *relatively perturbation-proof* iff there is a set of gain functions $\{g.i\}$ for this system such that every fixed point of the system is a Nash equilibrium w.r.t. $\{g.i\}$.
2. *Relatively perturbation-prone systems*:
   A stabilizing distributed system is called *relatively perturbation-prone* iff there is a set of gain functions $\{g.i\}$ for this system such that some fixed point of the system is not a Nash equilibrium w.r.t. $\{g.i\}$.
3. *Absolutely perturbation-proof systems*:
   A stabilizing distributed system is called *absolutely perturbation-proof* iff for every set of gain functions $\{g.i\}$ for this system, every fixed point of the system is a Nash equilibrium w.r.t. $\{g.i\}$.
4. *Absolutely perturbation-prone systems*:
   A stabilizing distributed system is called *absolutely perturbation-prone* iff for every set of gain functions $\{g.i\}$ for this system, there is a fixed point of the system that is not a Nash equilibrium w.r.t. $\{g.i\}$.

Note that each absolutely perturbation-proof system is also relatively perturbation-proof. Similarly, each absolutely perturbation-prone system is also relatively perturbation-prone.

Consider the case where the designers of a stabilizing distributed system wish to make this system perturbation-proof. In this case, if the designers can determine the natural set of gain functions for this system, then they should design the system to be relatively perturbation-proof w.r.t. this set of gain functions. On the other hand, if the designers cannot determine the one natural set of gain functions for this system, then they should design the system to be absolutely perturbation-proof.

In the next four sections, we give nontrivial examples of systems that are relatively perturbation-proof, relatively perturbation-prone, and absolutely perturbation-proof. We have chosen example systems which are as similar as possible; in all our examples, the system is a ring of n processes, numbered $0, 1, \ldots, n-1$. Each process i in the ring has two neighbors, $i-1$ and $i+1$; note that, throughout the paper, we use $i-1$ and $i+1$ to denote $(i-1 \bmod n)$ and $(i+1 \bmod n)$, respectively. The only state variable of a process is an integer (and is restricted to always have one of three values).

A very surprising result that we discover is that the class of absolutely perturbation prone systems is in fact empty. This indicates that every stabilizing distributed system is relatively perturbation-proof. From this last fact and from the fact that every stabilizing distributed system is either relatively perturbation-prone or absolutely perturbation-proof, we obtain the taxonomy of stabilizing distributed systems shown in Fig. 1.

In fact, seeing as the above classification can be simplified into two classes of systems instead of four, we suggest that in future work the terms "perturbation-proof" and "perturbation-prone" may be used to mean absolutely perturbation-proof and relatively perturbation-prone, respectively. All stabilizing systems are relatively perturbation-proof, so they need not

be given a special name. However, in this paper, we still need to prove the non-existence of absolutely perturbation-prone systems, so we do not use this simplified nomenclature; we use the full names of all four classes of system for the purpose of developing our theory.

4. Relatively perturbation-proof systems

In the next theorem, we state a sufficient condition that can be used to verify that a given stabilizing distributed system is relatively perturbation-proof.

Theorem 1. *A stabilizing distributed system that has n processes is relatively perturbation-proof if there is a set $\{g.i\}$ of gain functions for this system such that every fixed point s of this system satisfies at least one of the following two conditions:*

(a) *The gain function $g.i$ has its maximum value at s.*
(b) *For each global state s' that results from perturbing the local state of process i at s,*
- *either s' is a fixed point where the value of $g.i$ at s' is no more than its value at s,*
- *or process i has an action whose execution starting at s' returns the system to state s.*

Proof. We show that, if either (a) or (b) holds, then no perturbation of the local state of process i can guarantee that the value of the gain function $g.i$ (of process i) will increase. Thus, there is no incentive for any process to perturb its local state at s, and s is a Nash equilibrium w.r.t. $\{g.i\}$.

If (a) holds, then the value of the gain function $g.i$ at any fixed point s', other than s, cannot be greater than its value at the fixed point s. Thus, when s is the state of the system, no perturbation of the local state of process i will increase the value of $g.i$.

If (b) holds,then any perturbation of the local state of process i at s will either lead back to s, or lead to another fixed point s', where the value of $g.i$ is no more than its value at s. In either case, the perturbation of the local state of process i will not increase the value of $g.i$. □

Next, we provide an example of a relatively perturbation-proof system. Our objective of this exercise is two-fold. First, we want to show how to use Theorem 1 (above) to verify that a stabilizing distributed system is relatively perturbation-proof. Second, we want to demonstrate that the class of relatively perturbation-proof systems admits interesting systems.

Consider a ring that has n processes. Each process i has one local variable named $c.i$ that can be regarded as the color of process i. The value of $c.i$ is taken from the set $\{0, 1, 2\}$. A state s of this ring is a fixed point iff every two neighboring processes have distinct colors at state s.

Process i in this ring is specified as follows:

process $i : 0 \ldots (n-1)$
variable $c.i : 0, 1, 2$
begin
$i > 0 \wedge c.(i-1) = c.i \rightarrow c.i := (c.i + 1 \bmod 3)$
$[]\ i = n-1 \wedge c.(i+1) = c.i \rightarrow c.i := (c.i + 1 \bmod 3)$
end

It is straightforward to show that this ring is stabilizing and that the following predicate P holds at each fixed point of the ring:

$P =$ (For every i, where i is in the range $0 \ldots n-1$, $c.i \neq c.(i+1)$)

(Recall that we use $i-1$ and $i+1$ to mean $(i-1 \bmod n)$ and $(i+1 \bmod n)$ respectively.)

Consider the following set of gain functions for this system:

$$g.i = 0$$

In other words, the gain of every process is 0 at every fixed point.

The above set of gain functions may seem trivial, but it is in fact very interesting. Note that, at any fixed point, every gain function $g.i$ is at its maximum value, i.e., 0. Hence, by part (a) of Theorem 1 above, we conclude that any fixed point is a Nash equilibrium for this system with respect to the given $\{g.i\}$. In other words, the system is relatively perturbation-proof under this $\{g.i\}$.

Note that this proof is completely independent of the system we chose. No matter what system we choose, if we set all the gain functions to be $g.i = 0$ (or in general $g.i = c$ for any constant c, provided all gain functions are set to the same constant), all fixed points of the system become Nash equilibria. Thus, all stabilizing systems are relatively perturbation-proof. This is a very important observation — we will prove it formally in Section 7.

5. Relatively perturbation-prone systems

In the next theorem, we state a sufficient condition that can be used to verify that a given stabilizing distributed system is relatively perturbation-prone.

Theorem 2. *A stabilizing distributed system that has n processes is relatively perturbation-prone if there is a set $\{g.i\}$ of gain functions for this system such that at some fixed point s of this system, there is a process i that satisfies the following condition:*

- *The system has a second fixed point s' where s and s' differ only in the local state of process i and the value of $g.i$ at s is less than its value at s'.*

Proof. Let s and s' be two fixed points of a stabilizing distributed system whose set of gain functions is $\{g.i\}$. Assume that s and s' differ only in the local state of process i. Also assume that the value of the gain function $g.i$ (of process i) at state s is less than its value at state s'. Therefore, if process i perturbs its local state, starting from state s and forcing the system into state s', then the value of its gain function $g.i$ is guaranteed to increase. Thus, the fixed point s is not a Nash equilibrium. □

It is interesting to note that the coloring-ring system presented in Section 4 above is also an example of a relatively perturbation-prone system. We now use Theorem 2 (above) to show that the system is indeed relatively perturbation-prone.

Of course, the gain function presented in Section 4 is not suitable for proving that the system is relatively perturbation-prone, as we have already demonstrated that every fixed point of the system is indeed a Nash equilibrium with respect to this gain function. Hence, we now specify a new gain function $g.i$ for each process i in the ring:

$g.i = 0$ if $c.i = 0$

1. if $c.i \neq 0$ and $(c.(i-1) \neq 0$ or $c.(i+1) \neq 0)$
2. if $c.i \neq 0$ and $c.(i-1) = 0$ and $c.(i+1) = 0$

We use Theorem 2 to show that some fixed point of this ring is not a Nash equilibrium w.r.t. this set of gain functions. Consider the following state s of the ring (assuming that n is even):

$$c.0 = 1 \wedge c.1 = 0 \wedge c.2 = 1 \wedge c.3 = 0 \wedge \cdots \wedge c.(n-2) = 1 \wedge c.(n-1) = 0$$

s is clearly a fixed point, as no two neighboring processes have the same color.

To show that s is not a Nash equilibrium w.r.t. $\{g.i\}$, it is sufficient, by Theorem 2, to exhibit another fixed point s' where s and s' differ only in the value of exactly one $c.i$, say $c.1$, and show that the value of $g.1$ at s is less than its value at s'. Now consider the following state s' of the ring:

$$c.0 = 1 \wedge c.1 = 2 \wedge c.2 = 1 \wedge c.3 = 0 \wedge \cdots \wedge c.(n-2) = 1 \wedge c.(n-1) = 0$$

s' is also clearly a fixed point, as no two neighboring processes have the same color.

The two fixed points s and s' differ only in the value of $c.1$ and the value of $g.1$ at s is 0, less than its value 1 at s'. This proves that s is not a Nash equilibrium w.r.t. $\{g.i\}$. (Note that, as the value of the gain function $g.1$ increases from 0 to 1, the values of each of the two gain functions $g.0$ and $g.2$ is decreased from 2 to 1.)

Because some fixed point of this ring is not a Nash equilibrium w.r.t. the set of gain functions $\{g.i\}$ specified above, the ring is relatively perturbation-prone.

6. Absolutely perturbation-proof systems

In the next theorem, we state a sufficient condition that can be used to verify that a given stabilizing distributed system is absolutely perturbation-proof.

Theorem 3. *A stabilizing distributed system that has n processes is absolutely perturbation-proof if, for every set $\{g.i\}$ of gain functions for this system, every fixed point s of this system satisfies the following condition:*

- *For each global state s' that results from perturbing the local state of any process i at s, process i has an action whose execution starting at s' returns the system to the fixed point s.*

Proof. Let s be a fixed point of a stabilizing distributed system. Assume that for each process i in the system, and for each global state s' that results from perturbing the local state of process i at s, process i has an action whose execution, starting at s', returns the system to s. Therefore, no process i can be guaranteed to increase the value of its gain function by perturbing its local state at s. As no process has an incentive to perturb its local state at s, s is a Nash equilibrium w.r.t. any set of gain functions. □

We now give an example of an absolutely perturbation-proof system. While we would like to continue with the same system that was used as an example in the previous two sections, it should be clear that this is impossible. To see why, note that a relatively perturbation-prone system, by definition, must have a fixed point that is not a Nash equilibrium w.r.t. some set of gain functions, whereas an absolutely perturbation-proof system cannot have such a fixed point. In other words, a system is relatively perturbation-prone iff it is not absolutely perturbation-proof. In fact, the set of stabilizing systems can be partitioned into these two classes. (Similarly, the set of stabilizing systems can be partitioned into absolutely perturbation-prone and relatively perturbation-proof systems, but the set of absolutely perturbation-prone systems is empty. We will show this result in Section 7.) Hence, we use a new (but similar) system as an example of an absolutely perturbation-proof system. We also use Theorem 3 (above) to show that this system is indeed absolutely perturbation-proof.

Consider a ring that has n processes. Each process i has one local variable named $c.i$ that can be regarded as the color of process i. In specifying the actions of each process in this ring, we adopt the notation $A \equiv B$ to indicate that the two sets A and B are equal, and adopt the notation $A \subseteq B$ to indicate that set A is a subset of set B. Process i in this ring is specified as follows:

process $i : 0 \ldots (n-1)$
variable $c.i : \{0, 1, 2\}$
begin
$c.i \neq 0 \wedge \{c.(i-1), c.(i+1)\} \equiv \{1, 2\} \rightarrow c.i := 0$
$[]\ c.i \neq 1 \wedge \{c.(i-1), c.(i+1)\} \equiv \{0, 2\} \rightarrow c.i := 1$
$[]\ c.i \neq 1 \wedge \{c.(i-1), c.(i+1)\} \equiv \{2\} \rightarrow c.i := 1$
$[]\ c.i \neq 2 \wedge \{c.(i-1), c.(i+1)\} \subseteq \{0, 1\} \rightarrow c.i := 2$
end

A ranking function R for this ring can be specified as follows:

$$\begin{aligned} R = {} & 3 \times \#(\{i | c.i = c.(i-1) \vee c.i = c.(i+1)\}) \\ & + \#(\{i | c.i = 0\}) \\ & + \#(\{i | c.i = 1 \wedge c.(i-1) = 0 \wedge c.(i+1) = 0\}) \end{aligned}$$

It is straightforward to show that each action execution in this ring causes the value of this ranking function R to decrease by at least 1. (For example, suppose the first action in process 0 is executed starting at a global state where $c.(n-1) = 1$ and $c.0 = 1$ and $c.1 = 2$. This execution changes the value of $c.0$ to 0 and causes the value of R to be reduced by at least 4.) The existence of such a ranking function for this system guarantees that the system will eventually reach a global state where no action can be executed, i.e., a fixed point.

The following predicate P holds at each fixed point of this ring:

$$\begin{aligned} P = {} & (\forall i, i \text{ is in the range } 0 \ldots (n-1), \\ & (c.i = 0 \wedge \{c.(i-1), c.(i+1)\} \equiv \{1, 2\}) \vee \\ & (c.i = 1 \wedge \{c.(i-1), c.(i+1)\} \equiv \{0, 2\}) \vee \\ & (c.i = 1 \wedge \{c.(i-1), c.(i+1)\} \equiv \{2\}) \vee \\ & (c.i = 2 \wedge \{c.(i-1), c.(i+1)\} \subseteq \{0, 1\})) \end{aligned}$$

Note that when predicate P holds, the value of each $c.i$ is different from the values of $c.(i-1)$ and $c.(i+1)$.

We use Theorem 3 to show that each fixed point of this ring is a Nash equilibrium w.r.t. any set of gain functions. Consider a fixed point s of this ring where P holds. At s, each $c.i$ has any one of the values 0, 1, or 2. If the value of $c.i$ is 0 at s, and if this value is perturbed to 1 or 2 yielding the ring in a global state s', then the first action of process i can be executed at s' to return the ring to the fixed point s. Also if the value of $c.i$ is 1 at s, and if this value is perturbed to 0 or 2 yielding the ring in a global state s', then either the second or third action of process i can be executed at s' to return the ring to s. Similarly, if the value of $c.i$ is 2 at s, and if this value is perturbed to 0 or 1 yielding the ring in a global state s', then the fourth action of process i can be executed at s' to return the ring to s. Thus, by Theorem 3, the fixed point s is a Nash equilibrium w.r.t. any set of gain functions.

Because each fixed point of this ring is a Nash equilibrium w.r.t. any set of gain functions, we conclude that this ring is absolutely perturbation-proof.

This system example illustrates a method, implied by Theorem 3, for designing absolutely perturbation-proof systems. This method can be summarized as follows. To ensure that a stabilizing distributed system is absolutely perturbation-proof, consider each global state s' of this system, where s' differs from a fixed point s of the system only in the local state of process i, and ensure that process i has an action whose execution starting at s' yields the system in s.

To conclude the section, we identify an interesting subclass of absolutely perturbation-proof systems.

Theorem 4. *Any stabilizing distributed system, that has exactly one fixed point, is absolutely perturbation-proof.*

Proof. Consider a stabilizing distributed system that has only one fixed point s. If any process i perturbs its local state at s, the system (being stabilizing) is guaranteed to stabilize and thus return to state s. Therefore, the value of the gain function of process i remains unchanged. Thus, no process has an incentive to perturb its local state at s, so s is a Nash equilibrium w.r.t. any set of gain functions. The system is absolutely perturbation-proof. □

7. Absolutely perturbation-prone systems

In this section, we demonstrate that the class of absolutely perturbation-prone systems is empty.

Theorem 5. *Consider any stabilizing distributed system that has n processes. There is a set of gain functions {g.i} for this system such that every fixed point of the system is a Nash equilibrium w.r.t. {g.i}. Thus, no stabilizing distributed system is absolutely perturbation-prone. (In other words, every stabilizing distributed system is relatively perturbation-proof.)*

Proof. We define a set of gain functions in which each gain function $g.i$ has the (same) value 0 at every fixed point of the stabilizing distributed system. From Theorem 1 Part(a), every fixed point of the system is a Nash equilibrium w.r.t. this set of defined gain functions $\{g.i\}$.

Thus, for every stabilizing distributed system there exists at least one set of gain functions $\{g.i\}$ such that every fixed point of the system is a Nash equilibrium w.r.t. $\{g.i\}$. This implies that every stabilizing distributed system is relatively perturbation-proof, so the theorem holds. □

8. Global gain and Nash spread

In the previous sections, we study the gain of individual processes by assigning one gain function to each process. In this section, we discuss the *global gain*, i.e., the gain of the entire system at a fixed point. There are several measures for global gain; we use the standard measure, the *utilitarian* gain, which is the sum of the gain functions of all players.

The most interesting property of Nash equilibria is that they do not necessarily provide the maximum possible value of global gain, and are thus in a sense not optimal [1]. This property has been extensively studied, and has been formalized by Papadimitriou et al. as a measure called the *Price of Anarchy (PoA)* [5]. If we name the fixed point with the highest global gain the *social optimum*, then the Price of Anarchy is the ratio of the global gain at the social optimum to the lowest global gain possible at a Nash equilibrium.

In addition, we note that a system may have many Nash equilibria with respect to a given set of gain functions, and the global gain may be different at these different Nash equilibria. Clearly, except in the special case where the social optimum is a Nash equilibrium, the Price of Anarchy is not a suitable metric to measure how much the global gain varies between different Nash equilibria for the same system (and with the same set of gain functions). For example, consider two systems A and B. Both A and B have two Nash equilibria with respect to a given set of gain functions. In A, the global gain at each equilibrium is 1. In B, the global gains at the two equilibria are 1 and 99. Now suppose the social optimum for both systems has global gain 100. Then the PoA for both systems is 100. There is no way to differentiate between A, where the global gain does not vary between different Nash equilibria, and B, where it varies greatly. To formalize this notion, we need to define a new term, which we name the Nash Spread.

The *Nash Spread* of a system with respect to a set of gain functions is defined as the difference of the global gain at the best Nash equilibrium of the system, i.e., the Nash equilibrium with the greatest global gain, and the global gain at the worst Nash equilibrium of the system, i.e., the Nash equilibrium with the least global gain.

We now illustrate the Nash Spread in a stabilizing distributed system with an example. For our example, we choose a widely-studied system, the maximal matching bidirectional ring. In our example, the ring consists of n processes, where n is a multiple of 6.

Each process i in this ring has two neighbors: process $i-1$ and process $i+1$. Also, each process i in this ring has only one local variable named $m.i$ whose value is taken from the set $\{i-1, i, i+1\}$. When the value of $m.i$ is $i-1$ or $i+1$, we say that process i is *matched* to process $i-1$ or process $i+1$, respectively. When the value of $m.i$ is i, we say that process i is not matched. Process i in this ring is specified as follows:

process $i : 0 \ldots (n-1)$
variable $m.i : \{i-1, i, i+1\}$
begin

$$
\begin{aligned}
& m.i = i-1 \wedge m.(i-1) = i-2 \rightarrow m.i := i \\
[] \; & m.i = i+1 \wedge m.(i+1) = i+2 \rightarrow m.i := i \\
[] \; & m.i = i \wedge m.(i-1) = i \rightarrow m.i := i-1 \\
[] \; & m.i = i \wedge m.(i+1) = i \rightarrow m.i := i+1 \\
[] \; & m.i = i \wedge m.(i-1) = i-1 \wedge m.(i+1) \neq i \rightarrow m.i := i-1 \\
[] \; & m.i = i \wedge m.(i+1) = i+1 \wedge m.(i-1) \neq i \rightarrow m.i := i+1
\end{aligned}
$$

end

To show that this ring is stabilizing, we first specify a ranking function R that assigns to each global state s of the ring a nonnegative integer. We then establish that for each action execution, that causes the global state of the ring to change from s to s', $R(s) > R(s')$. A ranking function R for this ring can be specified as follows:

$$R = R.0 + R.1 + \cdots + R.(n-1)$$

where each $R.i$ is specified as follows:

$$\begin{aligned} R.i = 3 \text{ if } & (m.i = i-1 \wedge m.(i-1) = i-2) \vee \\ & (m.i = i+1 \wedge m.(i+1) = i+2) \\ 2 \text{ if } & (m.i = i) \\ 1 \text{ if } & (m.i = i-1 \wedge m.(i-1) = i-1) \vee \\ & (m.i = i+1 \wedge m.(i+1) = i+1) \\ 0 \text{ if } & (m.i = i-1 \wedge m.(i-1) = i) \vee \\ & (m.i = i+1 \wedge m.(i+1) = i) \end{aligned}$$

- When $R.i = 3$, process i is said to be *deluded*. It is paired to a process which is paired with someone else.
- When $R.i = 2$, process i is said to be *single*.
- When $R.i = 1$, process i is said to be *kind*. It is paired to a process which is single.
- When $R.i = 0$, process i is said to be *married*. It is paired to a process which is paired to it.

We now demonstrate that every one of the six actions in the action system, when executed, reduces the ranking function.

- We consider the first action, where $m.i = i-1$ transitions to $m.i = i$.
  We know that $m.(i-1) \neq i$ for the guard of the action to be true, and as the action has taken place, we know that its guard was true. Thus, process $i-1$ is not affected, and $R.(i-1)$ does not change.
  i transitions from deluded to single, so $R.i$ is reduced by 1.
  If $m.(i+1) = i$ then process $i+1$ transitions from deluded to kind, so $R.(i+1)$ is reduced by 2.
  Otherwise, $R.(i+1)$ is not reduced.
  No other processes are affected.
  Hence, R is reduced by either $0+1+0 = 1$ or $0+1+2 = 3$.
- The analysis for the second action is similar to that for the first.
- To consider the third action, where $m.i = i$ transitions to $m.i = i-1$.
  For this action, we check that $m.(i-1) = i$.
  $i-1$ transitions from kind to married. $R.(i-1)$ is reduced by 1.
  i transitions from single to married. $R.i$ is reduced by 2.
  If $m.(i+1) = i$, then $i+1$ transitions from kind to deluded. $R.(i+1)$ is increased by 2.
  Otherwise, $i+1$ is not affected, so $R.(i+1)$ is increased by 0.
  No other processes are affected.
  Hence, R is reduced by either $1+2-2 = 1$, or $1+2-0 = 3$.
- The analysis for the fourth action is similar to that for the third.
- To consider the fifth action, where $m.i = i$ transitions to $m.i = i-1$.
  For this action, we check that $m.(i-1) = i-1$.
  Process $i-1$ was single and remains single. Hence, $R.(i-1)$ does not change.
  i transitions from single to kind. Hence, $R.i$ is reduced by 1.
  We check that $m.(i+1) \neq i$. So process $i+1$ is not affected, and $R.(i+1)$ does not change.
  No other processes are affected.
  Hence, R is reduced by $0+1+0 = 1$.
- The analysis for the sixth (last) action is similar to that for the fifth.

The following predicate P holds at each fixed point of this ring:

$$\begin{aligned} P = & (\forall i, \text{ where } i \text{ is in the range } 0 \ldots n-1, \\ & (m.i = i-1 \rightarrow m.(i-1) = i) \wedge \\ & (m.i = i+1 \rightarrow m.(i+1) = i) \wedge \\ & (m.i = i \rightarrow m.(i-1) = i-2 \wedge m.(i+1) = i+2)) \end{aligned}$$

To see this, note that the guard of each action in the ring is false iff predicate P holds.

The gain function $g.i$ for each process i in this ring is as follows:

$$\begin{aligned} g.i = & 0 \text{ if } m.i = i \\ & 1 \text{ otherwise} \end{aligned}$$

We use Theorem 1 to show that each fixed point of this ring is a Nash equilibrium w.r.t. this set of gain functions $\{g.i\}$.

Consider a fixed point s of this ring. At s, each $m.i$ has one of the three values $i-1$, $i+1$, or i.

If $m.i$ has the value $i-1$ or $i+1$ at s, then $g.i$ has its maximum value (i.e., 1) at s, so process i clearly has no incentive to perturb the value of $m.i$ when the ring is at s.

If $m.i$ has the value i at s, we note that, as s is a fixed point, predicate P holds at s. As $m.i = i$, we conclude from predicate P that $(m.(i-1) = i-2 \wedge m.(i+1) = i+2)$ at s. If process i perturbs the value of its $m.i$ from i to $i-1$, then the first action in process i can be executed, as its guard becomes true. This will cause the ring to return to s, and the value of $g.i$ to remain unchanged. Thus, process i has no incentive to perturb the value of $m.i$ from i to $i-1$. Similarly, we can argue that process i has no incentive to perturb the value of $m.i$ from i to $i+1$. Therefore, s is a Nash equilibrium w.r.t. $\{g.i\}$. (Note that, as each fixed point of this ring is a Nash equilibrium w.r.t. the set of gain functions $\{g.i\}$ specified above, this system is relatively perturbation-proof. This is not surprising, as by Theorem 5, all stabilizing distributed systems are relatively perturbation-proof.)

We now determine the Nash Spread for this system.

We begin by considering the system as a ring with n nodes which represent the n processes. We now color the n edges of the ring blue and red. A blue edge connects matched nodes, and a red edge neighboring non-matched nodes. In other words, nodes i and $i+1$ are connected by a blue edge iff $m.i = i+1 \wedge m.(i+1) = i$; otherwise, they are connected by a red edge. At a fixed point, we have the following invariants:

1. No node has two blue edges incident on it. (This is because a process i cannot be matched with both neighboring processes $i-1$ and $i+1$ at the same time.) In other words, there cannot be two neighboring blue edges.
2. There cannot be two neighboring nodes that are matched to themselves. In other words, there cannot be three neighboring red edges. This is one of the conditions in P:

$$(m.i = i \rightarrow m.(i-1) = i-2 \wedge m.(i+1) = i+2)$$

Furthermore, we note that the global gain is equal to the number of processes that are matched to their neighbors, because these processes have gain 1, and the processes that are matched to themselves have gain 0. As a blue edge in the ring indicates two processes that are matched to each other, the global gain is twice the number of blue edges. Hence, to find the global gain at the best Nash equilibrium, it suffices to find the greatest possible number of blue edges in the ring and double it. Conversely, to find the global gain at the worst Nash equilibrium, we need to find and double the smallest possible number of blue edges according to the above constraints.

From the first condition above, we note that an upper bound on the number of blue edges is $n/2$; if we introduce $n/2+1$ blue edges, by the pigeon-hole principle, we have at least two neighboring blue edges. It is possible to attain this upper bound and have exactly $n/2$ blue edges, by simply coloring alternate edges red and blue. (This is the reason why we chose n to be an even number.) Hence, the global gain at the best Nash equilibrium is $2 \times n/2 = n$.

Similarly, from the second condition above we note that a lower bound on the number of blue edges is $n/3$. If we have more than $2n/3$ red edges, by the pigeon-hole principle, we have at least one series of three consecutive red edges. (To see this, partition the ring into $n/3$ pieces, each of length 3; to accommodate $2n/3+1$ red edges, by the pigeon hole principle, we are forced to make at least one piece have 3 red edges.) Again, this bound can be attained, by coloring edges in the sequence blue–red–red–blue–red–red ... .(This is the reason why we chose n to be a multiple of 3.) Hence, the global gain at the worst Nash equilibrium is $2 \times n/3 = \frac{2n}{3}$.

Therefore, we conclude that the Nash Spread for the above system with respect to the given gain function is $n - \frac{2n}{3} = \frac{n}{3}$. It is interesting to observe that this value is linear in the size of the system, and not upper bounded by any constant.

9. Concluding remarks

A stabilizing distributed system is a system that is guaranteed to return to a fixed point every time a "fault" yields the system in an arbitrary state that is not a fixed point. Thus, the property of stabilization makes systems recover from the effects of a large class of faults [6].

Unfortunately, stabilization does not protect a system from state perturbations caused by system processes that prefer one fixed point over another in accordance with some gain functions for these processes. To ensure that no process in a system has an incentive to intentionally perturb the system state, the system should be designed such that any perturbation caused by any process i in the system is not guaranteed to increase the value of the gain function of process i. This can be accomplished by ensuring that each fixed point of the system is a Nash equilibrium with respect to the given set of gain functions, one for each process in the system. We refer to a system, where each fixed point is a Nash equilibrium, as a perturbation-proof system.

In this paper, we identify two classes of perturbation-proof systems: relatively perturbation-proof systems and absolutely perturbation-proof systems. In a relatively perturbation-proof system, each fixed point is a Nash equilibrium w.r.t. a given set of gain functions (one for each process in the system). In an absolutely perturbation-proof system, each fixed point is a Nash equilibrium w.r.t. each possible set of gain functions (one for each process in the system).

Clearly, the concept of a relatively perturbation-proof system is useful when the set of gain functions for the system processes can be uniquely and completely identified. On the other hand, the concept of an absolutely perturbation-proof system is useful when there are multiple candidates for the set of gain functions.

The fact that one can design an absolutely perturbation-proof system, as indicated by Theorem 3, is the main contribution of this paper. Theorem 3 suggests that an absolutely perturbation-proof system can be designed as follows:

1. Consider each global state gs that is not a fixed point of the system and where changing the local state of one process i from ls to ls' changes the global state of the system from gs to a fixed point gs'.

2. Ensure that process i has an action that can be executed when the global state of the system is gs and when the local state of process i is ls, and ensure that the execution of this action causes the local state of process i to become ls'.

In this paper, we have used a slightly restricted definition of stabilizing systems; we require that every computation of the system be finite, so every computation of the system causes it to reach a fixed point and stop. We conjecture that this class of stabilizing systems may be equivalent to the class of silent stabilizing systems. It is clear that, in our model, the only "legitimate" states (where gain functions are defined) are fixed points. In a general self-stabilizing system, this may not hold; for instance, a system may cycle through a loop of legitimate states forever and still be self-stabilizing, because it exhibits closure and convergence. In our future work, we will attempt to develop the theory of Nash equilibria for all self-stabilizing systems.

It has been suggested recently that the definition of a Nash equilibrium can be extended to make it more relevant to Computer Science. (See for instance [7–9].) It is interesting to explore how such extensions of Nash equilibrium can impact the theory outlined in this paper.

Acknowledgements

The authors would like to express their gratitude to Dr. A. Ebnenasir of Michigan Technological University for bringing to their attention a correction to the bidirectional maximal ring matching system, and to Dr. S. Ghosh of the University of Iowa and Dr. S. Tixeuil of Universite Paris-Sud for pioneering the concept of selfish stabilization.

References

[1] J.F. Nash, Equilibrium points in n-person games, Proceedings of the National Academy of Sciences of the United States of America 36 (1) (1950) 48–49.
[2] M.M. Flood, Some experimental games, Management Science 5 (1) (1958) 5–26.
[3] A. Bhattacharya, S. Ghosh, Self-optimizing peer-to-peer networks with selfish processes (2007) 340–343.
[4] J. Cohen, A. Dasgupta, S. Ghosh, S. Tixeuil, An exercise in selfish stabilization, ACM Transactions on Autonomous and Adaptive Systems 3 (4) (2008) 1–12.
[5] E. Koutsoupias, C. Papadimitriou, Worst-case equilibria, STACS 99 (1999) 404–413.
[6] A. Arora, M. Gouda, Closure and convergence: A foundation for fault-tolerant computing, IEEE Transactions on Computers 19 (10) (1993) 1015–1027.
[7] I. Abraham, D. Dolev, R. Gonen, J. Halpern, Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation, in: Proceedings of the Twenty-fifth Annual ACM Symposium on Principles of Distributed Computing, ACM, New York, NY, USA, 2006, pp. 53–62.
[8] I. Abraham, D. Dolev, J. Halpern, Lower bounds on implementing robust and resilient mediators, in: Lecture Notes in Computer Science, vol. 4948/2008, Springer, Berlin, Heidelberg, 2008, pp. 302–319.
[9] J.Y. Halpern, Beyond nash equilibrium: Solution concepts for the 21st century, in: Proceedings of the 19th International Conference on Concurrency Theory, Springer-Verlag, 2008, pp. 1–1.
[10] E.W. Dijkstra, Self-stabilizing systems in spite of distributed control, Communications of the ACM 17 (11) (1974) 643–644.
[11] S. Dolev, Self-stabilization, MIT Press, 2000.
[12] M.G. Gouda, The triumph and tribulation of system stabilization, in: Distributed Algorithms, vol. 972/1995, Springer, Berlin, Heidelberg, 1995, pp. 1–18.