Nash Equilibria in Distributed Systems

Mohamed G. Gouda
Department of Computer Science
University of Texas
Austin, TX, USA
Email: gouda@cs.utexas.edu

H. B. Acharya
Department of Computer Science
University of Texas
Austin, TX, USA
Email: acharya@cs.utexas.edu

March 2, 2009

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, 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:

$$\begin{aligned} x.i = & 0 \text{ if prisoner } i \text{ selects the "stay silent" strategy} \\ & 1 \text{ if prisoner } i \text{ selects the "betray other prisoner" strategy} \end{aligned}$$

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:

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

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 can not 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.

2 Distributed 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:

$$< guard > \rightarrow < statement >$$

where $< guard >$ is a boolean expression over the local variables of process i and the local variables of the neighboring processes of process i, and $< statement >$ 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 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.)

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.

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.

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 computation ends at reaches 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 since 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 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.

3 Fixed Points and Nash Equilibria

In this section, we present some theorems that state sufficient conditions under which a fixed point of a stabilizing distributed system is, or is not, a Nash equilibrium. These sufficient conditions are convenient to use in verifying that a fixed point is, or is not, a Nash equilibrium, as illustrated by the system examples in the following sections.

The first theorem states a sufficient condition for a given fixed point, of a given stabilizing distributed system, to be a Nash equilibrium w.r.t. a given set of gain functions.

Theorem 1. *Consider a stabilizing distributed system that has n processes. 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 a Nash equilibrium w.r.t. $\{g.i\}$ if any of the following two conditions holds for every i in the range $0 \ldots (n-1)$:*

(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' yields the system in a fixed point s'' where the value of $g.i$ at s'' is no more than its value at s.

Proof. To prove part (b), note that process i can, by perturbing its local state, cause the global state of the system to become s'.

If s' is a fixed point, then it is given that the value of $g.i$ at s' does not exceed its value at s. Hence i cannot increase the value of its gain by perturbing the system.

If s' is not a fixed point, it is possible that the system will go to fixed point s'' from s' (because process i has an action whose execution starting at s' yields the system in s''). It is given that the value of $g.i$ at s'' does not exceed its value at s. Hence, in this case also, i is not guaranteed to increase the value of its gain by perturbing the system.

As, in all cases, no process in the system has an incentive to perturb it, s is a Nash equilibrium.

The proof of part (a) follows from the fact that the condition of part (a) is sufficient (though not necessary) for the condition of part (b), and they have the same conclusion (the state s is a Nash equilibrium), so part (b) implies part (a). □

The next theorem states a sufficient condition for a given fixed point, of a given stabilizing distributed system, to not be a Nash equilibrium w.r.t. a given set of gain functions.

Theorem 2. *Consider a stabilizing distributed system that has n processes. 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 not a Nash equilibrium w.r.t. $\{g.i\}$ if there exists an i in the range $0 \ldots (n-1)$ such that the following condition holds:*

> *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 the i for which the condition holds be process p. We denote the gain of p at state s by $g.p(s)$.

It is given that the set of gain functions $\{g.i\}$ is subject to the condition $g.p(s) < g.p(s')$. s and s' differ only in the local state of p. If the system is in the fixed point s, p can perturb its own local state and force the system into fixed point s', thereby increasing its gain. Hence the fixed point s is not a Nash equilibrium. □

The next theorem states a sufficient condition for a given fixed point, of a given stabilizing distributed system, to be a Nash equilibrium w.r.t. every set of gain functions.

Theorem 3. *Consider a stabilizing distributed system that has n processes. Let s be a fixed point of this system. The fixed point s is a Nash equilibrium w.r.t. every set of gain functions of this system if the following condition holds for every i in the range $0 \ldots (n-1)$:*

> *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 the fixed point s.*

Proof. Let p be any process in the system. When the system is at the fixed point s, p can perturb its own local state. After perturbation, the system goes to the global state t.

Now it is given that for any 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 the fixed point s. Setting $i = p$ and $s' = t$, there is an action in the action system of p that returns the system to state s.

The gain of p in state s is not greater than its gain in state s (it is exactly the same), for *any* set of gain functions. Hence there is a possibility that, if p perturbs its local state, the system will stabilize in a state where p's gain does not increase.

Hence s is a Nash equilibrium. □

The next theorem states sufficient conditions under which one can construct a set of gain functions to make every fixed point, of a stabilizing distributed system, a Nash equilibrium, or to make one fixed point, of a stabilizing distributed system, not a Nash equilibrium.

Theorem 4. *Consider a stabilizing distributed system that has n processes.*

(a) 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. $\{g.i\}$.

(b) If the system has two fixed points that differ only in the local state of one process, then there is a set of gain functions $\{g.i\}$ for this system such that one of the two fixed points is not a Nash equilibrium w.r.t. $\{g.i\}$.

Proof. For part (a), we consider any set of gain functions that assigns a constant value to the gain of each process (ie for each i, $g.i$ has the same value at all fixed points). At any fixed point s, if a process p perturbs the state of the system, the system will eventually stabilize at a state s' (as it is a stabilizing system). As the gain $g.p$ is the same at s and s', p has no incentive to perturb the system; hence s is a Nash equilibrium. As s is any fixed point, all fixed points in the system are Nash equilibria.

For part (b), we consider the fixed points s and s' which differ only in the local state of process p. Let the set of gain functions $\{g.i\}$ be such that $g.p(s) < g.p(s')$. If the system is in the fixed point s, p can perturb its own local state and force the system into fixed point s', thereby increasing its gain. Hence the fixed point s is not a Nash equilibrium. □

Based on the above definition of a fixed point being or not being a Nash equilibrium, we identify, in the next four sections, four classes of perturbation proof/prone systems:

1. Relatively perturbation-proof systems in Section 4
2. Relatively perturbation-prone systems in Section 5
3. Absolutely perturbation-proof systems in Section 6
4. Absolutely perturbation-prone systems in Section 7

We give nontrivial examples of systems in the first three classes and then show that the fourth class of systems is empty.

4 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. $\{g.i\}$.

In this section, we present 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) in verifying 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 maximal matching bidirectional ring that consists of n processes.

Each process i in this ring has two neighbors: process $i-1$ and process $i+1$. Note that, henceforth, we use $i-1$ and $i+1$ to denote ($i-1 \mod n$) and ($i+1 \mod$ n), respectively. 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
$$\begin{aligned} & m.i = i-1 \wedge m.(i-1) = i-2 \rightarrow m.i := i \\ [] \; & m.i = i+1 \wedge m.(i+1) = i+2 \rightarrow m.i := i \\ [] \; & m.i = i \wedge m.(i-1) \neq i-2 \rightarrow m.i := i-1 \\ [] \; & m.i = i \wedge m.(i+1) \neq i+2 \rightarrow m.i := i+1 \end{aligned}$$
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:

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

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

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

To show that the following predicate P holds at each fixed point of this ring:

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

it is sufficient to argue that the guard of each action in the ring is false when predicate P holds.
Specify the gain function $g.i$ for each process i in this ring 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 where predicate P holds. 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 1 at s, and

process i has no incentive to perturb the value of $m.i$ when the ring is at s. It remains now to consider the case where $m.i$ has the value i at s which implies, 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 causing 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. $\{g.i\}$.

Because each fixed point of this ring is a Nash equilibrium w.r.t. the set of gain functions $\{g.i\}$ specified above, this ring is relatively perturbation-proof.

5 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. $\{g.i\}$.

In this section, we present an example of a relatively perturbation-prone system and use Theorem 2 (above) to verify that the system is indeed relatively perturbation-prone.

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 $\{0, 1, 2\}$. At each fixed point of this ring, every two neighboring processes have distinct colors. 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:

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

Specify the gain function $g.i$ for each process i in this ring as follows:

$$\begin{aligned} g.i = & 0 \text{ if } c.i = 0 \\ & 1 \text{ if } c.i \neq 0 \text{ and } c.(i-1) \neq 0 \text{ or } c.(i+1) \neq 0 \\ & 2 \text{ if } c.i \neq 0 \text{ and } c.(i-1) = 0 \text{ and } c.(i+1) = 0 \end{aligned}$$

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 fixed point s of the ring (assuming that n is even):

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

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 fixed point s' of the ring:

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

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

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. $\{g.i\}$.(Note that a system is absolutely perturbation proof iff it is not 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 this section, we give an example of an absolutely perturbation-proof system and 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
$$\begin{aligned} & c.i \neq 0 \wedge \{c.(i-1), c.(i+1)\} \equiv \{1, 2\} \rightarrow c.i := 0 \\ [] \; & c.i \neq 1 \wedge \{c.(i-1), c.(i+1)\} \equiv \{0, 2\} \rightarrow c.i := 1 \\ [] \; & c.i \neq 1 \wedge \{c.(i-1), c.(i+1)\} \equiv \{2\} \rightarrow c.i := 1 \\ [] \; & c.i \neq 2 \wedge \{c.(i-1), c.(i+1)\} \subseteq \{0, 1\} \rightarrow c.i := 2 \end{aligned}$$
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, if the first action in process 0 is executed starting at a global state where $c.(n-1) = 1$ and $c.0 = 1$ and $c.1 = 2$, then 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, then ensure that process i has an action whose execution starting at s yields the system in s'.

The next theorem identifies an interesting subclass of absolutely perturbation-proof system.

Theorem 5. *Any stabilizing distributed system, that has exactly one fixed point, is absolutely perturbation-proof.*

Proof. Let us consider a stabilizing distributed system which has only one fixed point s. If any process p perturbs its local state, the system will stabilize in the state s: it must stabilize, and s is the only stable state. The gain of p at s, $g.p(s)$, is always equal to (and hence not greater than) the gain of p at s (itself). In other words, no process p is guaranteed to increase the value of its gain function by perturbing the system. This proves that s is a Nash equilibrium.

□

7 Absolutely Perturbation-Prone Systems

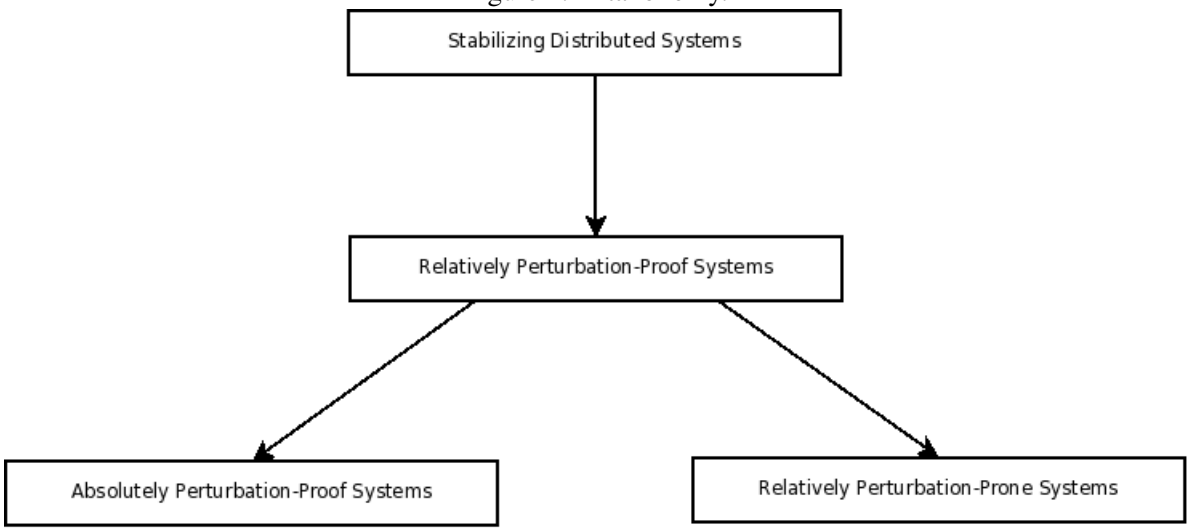
A stabilizing distributed system is called absolutely perturbation-prone iff for every set of gain functions $\{g.i\}$ for this system, there is a fixed point of the system that is not a Nash equilibrium w.r.t. $\{g.i\}$.

The next theorem, which follows directly from Theorem $4(a)$, states, in effect, that the class of absolutely perturbation-prone systems is empty.

Theorem 6. *No stabilizing distributed system is absolutely perturbation-prone. (In other words, every stabilizing distributed system is relatively perturbation proof.)*

Proof. By Theorem 4(a), for every stabilizing distributed system there must exist at least one set of gain functions $\{g.i\}$ such that every fixed point of the system is a Nash equilibrium. If we consider any stabilizing

Figure 1: A taxonomy.



distributed system S which is absolutely perturbation-prone, by definition, no such gain function for S can exist. If S exists, there is a contradiction: $\{g.i\}$ for S must exist and cannot exist. Hence S does not exist; no stabilizing distributed system can be absolutely perturbation-prone. □

Based on Theorem 1, we have the taxonomy of stabilizing distributed systems shown in Figure 1.

8 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 [3].

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'.

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 [4], [5], and [6].) It is interesting to explore how these extensions of Nash equilibrium can impact our theory of Perturbation-Freedom outlined in this paper.

References

[1] John F. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36(1):48–49, 1950.

[2] Merrill M. Flood. Some experimental games. *Management Science*, 5(1):5–26, 1958.

[3] Anish Arora and Mohamed Gouda. Closure and convergence: A foundation for fault-tolerant computing. *IEEE Transactions on Computers*, 19(10):1015–1027, 1993.

[4] Ittai Abraham, Danny Dolev, Rica Gonen, and Joe 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*, pages 53–62, New York, NY, USA, 2006. ACM.

[5] Ittai Abraham, Danny Dolev, and Joe Halpern. Lower bounds on implementing robust and resilient mediators. In *Lecture Notes in Computer Science*, volume Volume 4948/2008, pages 302–319. Springer Berlin / Heidelberg, 2008.

[6] Joseph Y. Halpern. Beyond nash equilibrium: Solution concepts for the 21st century. In *Proceedings of the 19th international conference on Concurrency Theory*, pages 1–1. Springer-Verlag, 2008.

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

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

[9] Mohamed G. Gouda. The triumph and tribulation of system stabilization. In *Distributed Algorithms*, volume Volume 972/1995, pages 1–18. Springer Berlin / Heidelberg, 1995.

[10] Wayne Goddard, Stephen T. Hedetniemi, David P. Jacobs, and Pradip K. Srimani. Self-stabilizing protocols for maximal matching and maximal independent sets for ad hoc networks. In *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, page 162.2, Washington, DC, USA, 2003. IEEE Computer Society.

[11] Shing-Tsaan Huang, Su-Shen Hung, and Chi-Hung Tzeng. Self-stabilizing coloration in anonymous planar networks. *Information Processing Letters*, 95(1):307–312, 2005.