Strategies for Repeated Games with Subsystem Takeovers Implementable by Deterministic and Self-Stabilizing Automata †

Shlomi Dolev ‡
Department of Computer Science, Ben-Gurion University of the Negev, Israel 84105
E-mail: dolev@cs.bgu.ac.il

Elad M. Schiller §
Department of Computing Science Chalmers University of Technology and Göteborg University Rännvägen 6*B* Göteborg, Sweden S-412 96.
Fax: +46-31-772 3663
E-mail: elad@chalmers.se
Corresponding author

Paul G. Spirakis ‡
Research Academic Computer Computer Technology Institute N. Kazantzakis str., University Campus, 265 00 Rio, Patras, Greece
E-mail: spirakis@cti.gr

Philippas Tsigas §
Department of Computing Science Chalmers University of Technology and Göteborg University Rännvägen 6*B* Göteborg, Sweden S-412 96.
E-mail: tsigas@chalmers.se

Abstract: Systems of selfish-computers, such as the Internet, are subject to transient faults due to hardware/software temporal malfunctions; just as the society is subjected to human mistakes due to a moment of weakness. Game theory uses punishment for deterring improper behavior. Due to faults, selfish-computers may punish well-behaved ones. This is one of the key motivations for forgiveness that follows any effective and credible punishment. Therefore, unplanned punishments must be proven to have ceased in order to avoid infinite cycles of unsynchronized behavior of "tit for tat".

We investigate another aspect of selfish-computer systems. We consider the possibility of subsystem takeover, say, by the use of hostile malware. The takeover may lead to joint deviations coordinated by an arbitrary selfish-computer that controls an unknown group of subordinate computers.

We present strategies that deter the coordinator (and its subordinates) from deviating in infinitely repeated games. We construct deterministic and finite automata that implement these strategies with optimal complexity. Moreover, we prove that all unplanned punishments eventually cease by showing that the automata can recover from transient faults.

Keywords: Game Theory; Folk-Theorem; Joint Deviations; Finite-Automata; Self-Stabilization; Repeated Games; Subsystem Takeover; Autonomous Systems; Adaptive Communications.

. .

1 Introduction

Systems of selfish-computers, such as the Internet, introduce new challenges in distributed computing, game theory, and computational complexity. They exhibit both cooperative and uncooperative interactions. [1] While cooperative and uncooperative interactions have been extensively studied as the two extremes, the study of joint deviations in uncooperative repeated games has been neglected so far. In systems of selfish-computers (where out-of-band communication is possible), it is unlikely that selfish-computers cannot conspire. Therefore, it is imperative to consider joint deviations. New models of distributed systems and uncooperative games are needed to consider joint deviations. This paper presents one such new model of subordinates' deviations in infinitely repeated games. We find a simple and optimal strategy for deterring subordinates' deviations. Moreover, the strategy allows selfish-computers to recover from involuntary misbehaver and unplanned punishments.

1.1 Subsystem takeovers

Stability and self-enforcement are two of the most attractive properties that equilibria offer. We consider equilibrium of strategies that autonomous agents have devised, and all possible joint deviations by a group of at most D deviators. Stability and self-enforcement are achieved when the autonomous agents deter the deviation; if any one of all possible joint deviations happens, then the deviating group will lose payoff, compared to what they would get by obeying the equilibrium strategy.

Many noncooperative games follow the assumption of unilateral deviation (e.g., Nash equilibrium in noncooperative games [37]). In practice, it is unlikely that agents cannot conspire and coordinate joint deviations. Alternatively, joint deviations are considered in cooperative games as a competition among coalitions of agents, rather than among individual agents (e.g., strong Nash equilibrium in cooperative games [6, 41]). Unfortunately, cooperative games enforce cooperative behavior among agents using mechanisms that do not exist in selfish-computer systems.

We study the crossover between noncooperative and cooperative games. We consider noncooperative games in which every joint deviation is coordinated by an arbitrary agent – the *coordinator*. The coordinator selects the actions of its *subordinates*, and has no control over the *autonomous* agents. The coordinator and autonomous agents maximize their individual payoffs by a deliberate and optimized selection of actions. The autonomous agents cannot enforce coordinated behavior, and have no a priori knowledge about the identities of the coordinator and its subordinates. Stability and self-enforcement are achieved when the autonomous agents deter the coordinator (and its subordinates) from deviating.

Subsystem takeovers can model scenarios in which users abuse their access privileges to remote machines. (Such privileges might be gained by hostile malware.) The abuser (i.e., the coordinator) deprives the individual benefit of an arbitrary subset of agents (i.e., the remote machines). We assume that the coordinator does not compensate its subordinates for their losses. Therefore, the notion of subordinates' deviation should be modeled by games that have no side payments or transferable utilities (similar to the definitions in [7]). Hence, neither the sum nor the maximum of the deviators' payoffs should be considered (our approach is different than [32]).

Corollary 1 (Lemma 1 of Section 5)**.** *Autonomous and selfish agents can deter joint deviations of the subordinate groups using deterministic and finite automata.*

1.2 Complexity issues of games with subsystem takeovers

Computational game theory has several ways of measuring complexity (see [38]). The two most related to games with subordinates' deviations are:

[†] An extended abstract of this paper appeared in [26]

[‡] Partially supported by the ICT Programme of the European Union under contract number ICT-2008-215270 (*FRONTS*)

[§] Partially supported by the European Science Foundation (ESF) Scientific Programme of (*MiNEMA*)

[1] Cooperative interactions refers to scenarios in which contracts among selfish-computers are usually held on to and can be made legally binding; uncooperative interactions refers to scenarios in which there is mistrust and no external enforcement mechanisms are available.

• Costs of finding strategies

The computational complexity of a game model describes the asymptotic difficulty of finding a solution as the game grows arbitrarily. We give the *shared responsibility game* as an example for which it is possible to efficiently find strategies that deter subordinates' deviations. Unfortunately, this is not the general case; finding strategies that deter joint deviations is at least as hard as finding a Nash equilibrium, which is known to also be computationally intractable for infinitely repeated games, see [20, 13].

• Costs of implementing strategies

What is the minimal amount of memory required for implementing a given strategy? Kalai and Stanford [33] answer this question in the context of finite-state machines, and show that it is the size of the smallest automaton that can implement the strategy. [2] The difficulty that Corollary 2 raises is that selfish-computers that try to deter subordinates' deviations may exhaust their resources.

Corollary 2 (Lemma 2 of Section 6)**.** *Strategies for deterring subordinates' deviations have the complexity of* $\Theta(D\binom{n}{D})$*, where* n *is the number of agents, and* D *is an upper bound on the size of the subordinate group.*

1.3 Tolerating transient faults

When designing a distributed system of selfish-computers, it is unsuitable to assume that failures never occur (see [24, 23, 25]). Most of the existing literature on repeated games considers agents that have identical views on the history of actions. In practice, it is unlikely that all selfish-computers never fail to observe an action in an infinite system run. Once a single selfish-computer misinterprets an action, the outcome of the game cannot be predicted by the current theory.

Transient faults are regarded as faults that temporarily violate the assumptions made by the system designer about the game model and the system settings. For example, the system designer may assume the existence of a constant upper bound, D, on the size of the subordinate group. In this case, a transient fault could be a joint deviation of more than D agents. (Recall that Corollary 2 implies a possible failure in allocating sufficient memory as D grows.) Actions that some agents misinterpret are transient faults as well. Thus, a transient fault is defined as an arbitrary violation of the designer's assumptions for a finite period. Transient faults imply an arbitrary starting state after the system has returned to obey the designer's assumptions for an infinitely long period.

[2] The size of an automaton is the cardinality of its state set.

1.4 Self-stabilization

Self-stabilizing systems [21, 22] can recover after the occurrence of transient faults. These systems are designed to automatically regain their consistency from any starting state of the automaton. The arbitrary state may be the result of violating the assumptions about the game model or the system settings. The correctness of a self-stabilizing system is demonstrated by considering every sequence of actions that follows the last transient fault and is, therefore, proved assuming an arbitrary starting state of the automaton. Corollary 3 implies that there are self-stabilizing systems of selfish-computers that deter subordinates' deviations.

Corollary 3 (Lemma 3 of Section 7)**.** *Self-stabilizing automata can satisfy the assertion of Corollary 1.*

1.5 Our contribution

We present deterministic self-stabilizing finite automata for deterring subsystem takeovers. We show how to deter subsystem takeovers in uncooperative games using a simple strategy that can be implemented by finite automata.

• Costs of games with subsystem takeovers

We analyze the complexity of our strategy and demonstrate a lower bound that asymptotically matches the costs of our implementation. We note that prior work, such as [41], does not explicitly bound the complexity of their strategies.

• Self-stabilization

The automaton is self-stabilizing and provides a strategy that deals with deviations, transient faults, and mistakes that are done at a moment of weakness. After the occurrence of such mistakes, the system punishes the deviators for a bounded number of periods. Moreover, after the occurrence of an unexpected combination of deviations (or transient faults), the system is guaranteed to recover within

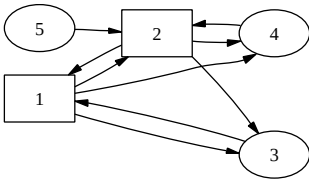
We describe an example of a system with n selfish-computers. We denote by $N = \{1, \ldots, n\}$ the set of agents; each represents a selfish-computer. Every agent decides whether to be a Server or a Client. An agent receives a payoff of -1 for every period in which it decides to be a Server (independently of the other agents' choices). The payoff of a Client depends on the decisions of other agents.

In every period, every Server i, reveals its access list s_i of agents that can access its services. Moreover, every Client j, writes a single agent, say i, in s_j (possibly not matching the access list of agent i). Namely, for Client j, $s_j = \{i\}$ means that j would like to access i as a server Server. In case that i indeed chooses to be a Server, then Client j can access i if $j \in s_i$.

Let $G = (V, E)$ be a directed graph that is induced by the access lists. The set V is the set of agents, N. Let $i \in N$ be a Server and $j \in N$ be an agent (that is either a Client or a Server), then $(i, j) \in E$ if, and only if, $i \in s_j \wedge j \in s_i$. See the example of the right.

The Client j receives the payoff of $+1$ (or 0) if the strongly connected component that contains j in the induced graph G includes (respectively, does not include) the majority of agents in N (i.e., more than $\frac{|N|}{2}$).

An example of the induced graph



Above is an example of the induced directed graph. Servers 1 and 2 (boxes) receive the payoff of -1 each. Clients 3 and 4 (ellipses) receive the payoff of $+1$ each. Client 5 is not connected to a strongly connected component that includes a majority of agents. Therefore, 0 is the payoff of client 5.

Figure 1: The shared responsibility n-agent game.

a bounded number of periods. We believe that requiring a bounded number of periods of punishment and recovery is essential within the scope of self-stabilization, because the system can be started with agents being punished in spite their excellent past behavior.

• Autonomous systems of selfish-computers

While we do not claim to be the first to bridge game theory and fault tolerance, we believe that our work provides an important insight to self-stabilizing distributed systems. On one hand, we consider joint deviations that are harder to deal with than deviations in which all deviators are rational (as in [6, 1, 5]) because we assume that not all deviators are rational. On the other hand, we offer equilibria that are more credible than known fault tolerant equilibrium because we consider new realistic system settings of infinitely repeated games in which not all deviators are faulty (as in [28, 17, 3, 35, 18, 4]). [3]

[3] One may think about a subordinate agent as a faulty one. The reason is that a subordinate agent does not selfishly promote its own benefit, because it is controlled by another selfish (non-faulty) agent, i.e., the coordinator. However, subordinate agents do not present an arbitrary behavior (as in [28, 17, 3, 35, 18, 4]).

This work facilitates the design of autonomous systems that are required to deter subsystems takeovers. Subsystem takeovers can model scenarios in which users abuse their access privileges to remote machines. (Such privileges might be gained by hostile malware.) We show that a simple strategy can deter subsystem takeovers. Moreover, we are the first to show that the simple strategy guarantees system recovery after the occurrence of an unexpected combination of deviations.

1.6 Document structure

We illustrate the problem at hand (Section 3) and list the preliminaries before we define subsystem takeovers (Section 4). Then, we explain the proofs of Corollary 1 (Section 5), Corollary 2 (Section 6), and Corollary 3 (Section 7). (For brevity, some parts of the complete proof of Corollary 3 appear in Section 9 of the Appendix and in [27].) Lastly, we draw our concluding remarks.

2 Preliminaries

Throughout we follow the definitions and notations of [39]. Throughout we use N to denote the set of agents, A_i the set of action, and $\succsim_i$ the preference relation (where $i \in N$ is an agent). We represent single stage games, G, in their strategic form as $\langle N, A = (A_i), \succsim = (\succsim_i)\rangle$ and in their extensive form as $\langle N, H, \succsim = (\succsim_i)\rangle$. We refer to solution concepts such as Nash equilibrium, and feasible and enforceable payoff profiles.

2.1 Profiles

We refer to a collection of values of some variable, one for each agent, as a *profile*. Similar to the single element profile notation (i.e., $x = (x_i)_{i\in N}$, (x_{-i}, x_i), and X_{-i} of [39]), we consider profile notation for subsets of elements $s \subseteq N$. We define profiles x_s, x_{-s} to be the list of elements $(x_i)_{i\in s}$, respectively $(x_i)_{i\in N\setminus s}$ for all $s \subseteq N$. Given a list of elements $x_{-s} = (x_i)_{i\in N\setminus s}$ and a profile $x_s = (x_i)_{i\in s}$, we denote by (x_{-s}, x_s) the profile $(x_i)_{i\in N}$. We denote by X_s, X_{-s} the sets $\times_{j\in s} X_j$, respectively $\times_{j\in N\setminus s} X_j$, where the set of elements is defined as X_i for each $i \in N$.

2.2 Repeated games

Throughout we consider the game $\Gamma = \langle N, H, \succsim = (\succsim_i)\rangle$, in which the constituent game $G = \langle N, A = (A_i), \succsim = (\succsim_i)\rangle$ is repeated an infinite number of times. We assume that all periods (plays) are synchronous, i.e., all agents make simultaneous moves. Moreover, by the end of each round, all agents have observed the actions taken by all other agents.

2.3 Preference relations for repeated games

A preference relation expresses the desire of the individual for one particular outcome over another. For the constituent game, G, the relation $\succsim_i$ refers to agent i's preferences. Suppose that $\succsim_i$ can be represented by a *payoff/utility function* $u_i : A \to \mathbb{R}$, for which $u_i(a) \geq u_i(b)$ whenever $a \succsim_i b$. We assume that in Γ, agent i's preference relation $\succsim_i^\star$ is based on a payoff function u_i. Namely, $(a^t) \succsim_i^\star (b^t)$ depends only on the relation between the corresponding sequences $(u_i(a^t))$ and $(u_i(b^t))$.

2.4 The limit of means criterion

The limit of the means criterion [9, 42] treats the future as no more important than the present. The sequence v_i^t of real numbers is preferred to the sequence w_i^t if and only if $\lim_{T\to\infty} \Sigma_{t=1}^T (v_i^t - w_i^t)/T > 0$. $\lim_{T\to\infty} \Sigma_{t=1}^T (v_i^t - w_i^t)/T > 0$.

2.5 Games in extensive form

The *extensive form* of a game describes the game as a decision tree, which is directed from the root downwards. Each node of the tree represents every reachable stage of the game. Starting from the initial node, agents take synchronous (simultaneous) choices of actions. Given any internal node in the tree, each possible action profile leads from that node to another node. A node is said to be terminal if it has no outgoing action profiles.

A history is a sequence of action profiles that corresponds to a directed path from the root of the decision tree. The set of all histories is denoted by H. We note that history $(a^k)_{k=[1,K]} \in H$ is terminal if it is infinite or if there is no $(a^k)^{K+1}$ such that $(a^k)_{k=[1,K+1]} \in H$. The set of terminal histories is denoted Z. Moreover, for each agent $i \in N$ a preference relation $\succsim_i$ is defined on Z. Let h be a history of length k; we denote by (h, a) the history of length $k+1$ consisting of h followed by a. We denote an extensive game with perfect information and synchronous (simultaneous) moves as $\Gamma = \langle N, H, \succsim = (\succsim_i)\rangle$.

2.6 Subgames

In large (or infinite) decision trees, it is useful to isolate parts of the tree in order to establish simpler games. When the initial node of a subgame is reached in a larger game, agents can concentrate on only that subgame; they can ignore the history of the rest of the game.

Let $\Gamma = \langle N, H, \succsim\rangle$ be an extensive game with perfect information and synchronous (simultaneous) moves. Let $H|_h$ be the set of sequences h' of actions for which $(h, h') \in H$. We define $\succsim_i |_h$ as $h' \succsim_i |_h h''$ if, and only if, $(h, h') \succsim_i (h, h'')$. The *subgame* $\Gamma(h)$ of game Γ that follows the history h is the extensive game $\langle N, H|_h, \succsim\rangle$. By defining a new decision tree, $H|_h$ in the subgame, agents can concentrate on only the subgame $\Gamma(h)$; they can ignore the history of the rest of the game.

2.7 Strategies for individuals

Agents protect their benefits by following a long-term plan (or program) of action selection. We call this plan the *(individual) strategy* st_i of agent $i \in N$. We define st_i as a function that assigns an action in A_i to every finite sequence of outcomes in the game $\Gamma = \langle N, H, \succsim = (\succsim_i) \rangle$.

3 Background of the problem

We illustrate basic notions in game theory and elements of the problem at hand using an example (a more detailed tutorial appears in [31]).

Our example of a system of selfish-computers considers the *shared responsibility* service, which is presented in Figure 1. We model the service as an uncooperative game among n agents. An agent decides whether it would participate as a server or as a client. Servers specify their access list; the list restricts the access of other agents (clients or servers). Clients benefit the most whenever they can access a majority of selfish-computers via a server that relays the communications.

3.1 Single stage games

The payoff matrix that considers a 3-agent instance of the game is presented in Figure 2. The matrix describes the payoff that agent 1 gets for every possible combination of actions that the agents may take. We note that the payoff of any server is less than the payoff of any client (in the single stage game). Therefore, all agents decide to be clients and thus receive the payoff of 0. [4] This is *Nash equilibrium*.

3.2 Infinitely repeated games

If the single stage game is repeated infinitely, the agents can benefit from a *sequence of cooperation* steps in which the agents take turns for responsibility. A possible cooperation sequence is presented in Figure 3(a). In this sequence of cooperation, every agent is supposed to eventually receive the average payoff of $(|N|-2)/|N|$. [5] In that sense, if all agents "play along" then $(|N|-2)/|N|$ is a *feasible* payoff. Unfortunately, the selfish agent j might deviate from the sequence of cooperation. Suppose that j knows that all other agents would always allow j to access their services. Then agent j can decide to deviate and be a client whenever it is the turn of j to be a server.

3.3 Punishment

In uncooperative games, all agents must monitor the actions of agent j, and deter j from deviation by punishment. The punishment should hold j to the minimal payoff value that the punishing agent can enforce. In the shared responsibility game, j receives a minimal enforceable payoff when the punishing agents take a sequence of steps in which: (1) they exclude j from their access list, and (2) they "play along" among themselves. A punishing sequence in which the payoff of 0 is enforced on agent 3 is shown in Figure 3(b). In that sense, 0 is an *enforceable* payoff.

3.4 Grim trigger strategies

One can consider the following strategy. Initially, agent i follows the sequence of cooperation. However, as soon as agent j defects, agent i follows the punishment scheme that *forever* holds j down to its minimal enforceable payoff. In the shared responsibility game, agent j cannot benefit from defecting, because the punishment eventually reduces j's average payoff from $(|N|-2)/|N|$ to 0. Thus, agent j would prefer to cooperate. Thus, the grim trigger strategy is Nash equilibrium for infinitely repeated games.

There is a clear disadvantage to the grim trigger strategy; while the agents hold down j to its minimal payoff, their payoff might be reduced as well. The equilibrium will continue to be played forever, even if j defects only once. Thus, the threatened response may seem unreasonable, especially when it is too costly to the punishing agents. In other words, the knowledge that the punishing agents will respond to j's defection by an unrelenting punishment is what keeps j from defecting. However, if j does in fact defect, it may no longer be beneficial for the punishers to punish. That is what makes the grim trigger strategy unbelievable.

[4] Starting from any entry of the matrix, consider a sequence of (unilateral) changes to the agents' choice of action. Once every agent was able to change its choice, all agents choose to be clients.

[5] In every $|N|$ periods of the cooperative sequence there are $N-1$ periods in which agent $i \in N$ is a Client (that is served by others) and a single period in which the payoff of agent i is -1.

Agent 1 \ Agent 2	⟨Server, {1}⟩	⟨Server, {3}⟩	⟨Server, {1, 3}⟩	⟨Client, {1}⟩	⟨Client, {3}⟩
⟨Server, {2}⟩	−1	−1	−1	−1	−1
⟨Server, {3}⟩	−1	−1	−1	−1	−1
⟨Server, {2, 3}⟩	−1	−1	−1	−1	−1
⟨Client, {2}⟩	+1	0	+1	0	0
⟨Client, {3}⟩	+1	+1	+1	+1	+1

(a) Agent 3: ⟨Server, {1}⟩ or ⟨Server, {1, 2}⟩

Agent 1 \ Agent 2	⟨Server, {1}⟩	⟨Server, {3}⟩	⟨Server, {1, 3}⟩	⟨Client, {1}⟩	⟨Client, {3}⟩
⟨Server, {2}⟩	−1	−1	−1	−1	−1
⟨Server, {3}⟩	−1	−1	−1	−1	−1
⟨Server, {2, 3}⟩	−1	−1	−1	−1	−1
⟨Client, {2}⟩	+1	0	+1	0	0
⟨Client, {3}⟩	0	0	0	0	0

(b) Agent 3: ⟨Server, {2}⟩, ⟨Client, {1}⟩, or ⟨Client, {2}⟩

Figure 2: The payoff matrices of the shared responsibility game with 3-agents. The headings of tables, columns and rows are in the form of $\langle a, s\rangle$, where a is the action, and s is the access list of agent 1. The matrix is symmetrical and thus the payoffs of agents 2 and 3 are, in fact, described as well.

3.5 The perfect folk theorem

In the context of repeated games, Nash equilibrium can be refined to exclude strategy such as the grim trigger strategy (see subgame perfect equilibrium [43, 39]). Roughly speaking, the "refined equilibrium" represents Nash equilibrium in every "stage" of the repeated game. And thus, if agent j defects only once, then the punishing agents would punish j merely for a finite period. At the end of the punishment period, all agents return to cooperate.

The perfect Nash Folk theorem provides strategies that can deter an agent from deviating unilaterally (see [39], and references therein). A sketch of a strategy is presented in Figure 4. The strategy is a deterministic and finite automaton that plays the sequence of cooperation as long as there are no deviations. The automaton retaliates to the deviation of an agent with a punishment for a sufficiently long (but finite) period.

4 Subsystem Takeovers

The perfection property is a key feature of the notion of subgames. This property specifies that a strategy profile is Nash equilibrium in every subgame. Given a game $\Gamma = \langle N, H, \succsim\rangle$, we define the strategy profile $st = (st_i)_{i\in N}$ as a *subgame perfect equilibrium that is t-defendable* from joint deviations of any subordinate group $s \in S(t)$, where $t \in [1, |N|)$ is a constant and $S(t) = \{s : s \in 2^N \setminus \{\emptyset, N\} \wedge |s| \leq t\}$ is the set of all possible subordinate groups. [6]

4.1 Joint strategies

Let $s \subseteq N$ be any group of agents, $st_s = (st_i)_{i\in s}$ their joint strategies and $h \in H$ a history of the extensive game $\Gamma = \langle N, H, \succsim\rangle$. We denote by $\Gamma(h)$ the subgame that follows the history h. Moreover, denote by $st_s|_h$ the joint strategies that st_s induces in the subgame $\Gamma(h)$ (i.e., $st_s|_h(h') = st_s(h, h')$ for each $h' \in H|_h$). We denote by O_h the outcome function of $\Gamma(h)$. Namely, $O_h(st_{-s}|_h, st_s|_h)$ is the outcome of the subgame $\Gamma(h)$ when the agents take the strategy profile $(st_{-s}|_h, st_s|_h)$.

4.2 Perfect and t-defendable subgame equilibria

Given a number $t \in [1, |N|)$, we say that the subgame $st = (st_i)_{i\in N}$ cannot recover from a joint deviation of a subordinate group $s \in S(t)$, if there is a joint deviation st'_s of the agents in s, such that for any $h \in H \setminus Z$ it holds that for an arbitrary agent $i_{coord} \in s$ (the coordinator) we have $O_h(st_{-s}|_h, st'_s|_h) \succ_{i_{coord}}|_h O_h(st_{-s}|_h, st_s|_h)$. When the subgame st can recover from any joint deviation of the subordinate groups, $s \in S(t)$, we say that st is a t-defendable equilibrium.

We note that while the joint deviation st'_s is required to guarantee the benefit of coordinator, there are no guarantees for the benefits of the subordinates. In more detail, there could possibly exist a subordinate agent $j_{subor} \in s \setminus \{i_{coord}\}$, such that

[6] We do not consider the case of $s = N$, because it refers to a system that is controlled by a single agent.

Period	Agent 1	Agent 2	Agent 3
1	$\langle \mathsf{Server}, \{2,3\}\rangle$	$\langle \mathsf{Client}, \{1\}\rangle$	$\langle \mathsf{Client}, \{1\}\rangle$
2	$\langle \mathsf{Client}, \{2\}\rangle$	$\langle \mathsf{Server}, \{1,3\}\rangle$	$\langle \mathsf{Client}, \{2\}\rangle$
3	$\langle \mathsf{Client}, \{3\}\rangle$	$\langle \mathsf{Client}, \{3\}\rangle$	$\langle \mathsf{Server}, \{1,2\}\rangle$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

(a) A sequence of cooperation for 3 agents.

Period	Agent 1	Agent 2
1	$\langle \mathsf{Server}, \{2\}\rangle$	$\langle \mathsf{Client}, \{1\}\rangle$
2	$\langle \mathsf{Client}, \{2\}\rangle$	$\langle \mathsf{Server}, \{1\}\rangle$
$\vdots$	$\vdots$	$\vdots$

(b) A scheme for punishing agent 3.

Figure 3: Two examples of cooperation sequences. The entry format follows that of Figure 2.

$O_h(st_{-s}|_h, st'_s|_h) \quad \prec_{j_{subor}} |_h \quad O_h(st_{-s}|_h, st_s|_h)$. We mention that joint deviations in which agent j_{subor} may exist are not considered by [5, 41, 6, 1, 12, 36] (see the discussion in Section 8).

4.3 s-enforceable payoff profiles

To support a feasible outcome, each subordinate group and its coordinator must be deterred from deviating by being "punished". The concept of enforceable payoff profiles considers a single agent that may deviate (see [39]). We extend that concept to consider the deviation of subordinate groups.

Define minmax payoff in game Γ of a subordinate group $s \in S$, denoted $v_i|_s$, to be the lowest payoff that the autonomous agents $N' = N \setminus s$ can force upon the coordinator $i \in s$:

$$v_i|_s = \min_{a_{-s} \in A_{-s}} \max_{a_s \in A_s} u_i(a_{-s}, a_s). \tag{1}$$

Given a minmax payoff profile $v_i|_s$, the payoff profile $w|_s$ is called strictly *s-enforceable* if $w_i|_s > v_i|_s$ for all $i \in s$. Denote by $p_{-s} \in A_{-s}$ one of the solutions of the minimization problem on the right-hand side of equation 1.

5 Folk Theorem of Subsystem Takeovers

The folk theorem is a class of proofs which show that every feasible and enforceable profile of payoffs can be achieved by a subgame perfect equilibrium (see [39], and references therein). In this section, we present Lemma 1, which is a folk theorem for games with subsystem takeovers.

Joint deviations are more complex than unilateral ones. The coordinator of a subordinate group can synchronize its subordinates' deviations and divide them in groups: a group of provoking agents, and a group of "fifth column" agents. [7]

For example, suppose that in the shared responsibility game the subordinate group is $s = \{j_1, j_2\}$. The coordinator can synchronize the following deviation: Agent j_1 provokes the autonomous agents by not following its duty to be a Server. The autonomous agents retaliate by punishing agent j_1. We note that the deviation of the provoking agents does not reveal the fact that the "fifth column" agent, j_2, is the coordinators' subordinate. Therefore, the autonomous agents expect j_2 to participate in the punishment of its fellow member j_1. Alas, the agent j_2 betrays the autonomous agents; while the autonomous group is punishing, agent j_2 deviates from punishing and enters j_1 in its access list. Hence, the synchronized deviation can protect j_1's profit.

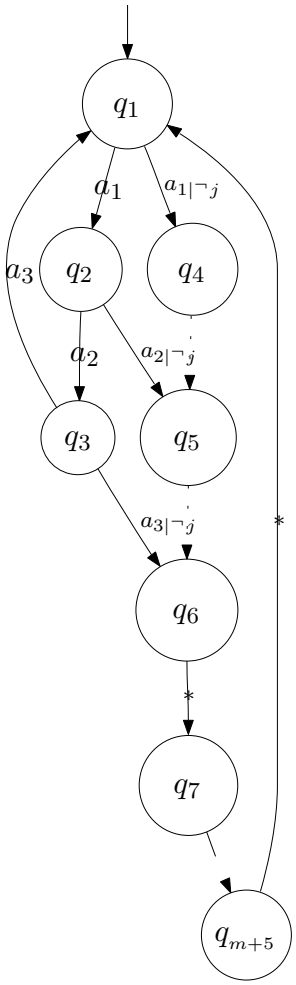
Lemma 1 considers the payoff profiles that the autonomous group can guarantee in the presence of subsystem takeovers. A payoff profile w that is strictly s-enforceable $\forall s \in S(D)$ is called strictly *D-defendable*. If $a \in A$ is an outcome of Γ for which $u(a)$ is strictly D-defendable in Γ, then we refer to a as a *strictly D-defendable outcome of* Γ.

Lemma 1 (Corollary 1)**.** *Let Γ be an infinitely repeated game of $G = \langle N, (A_i), (u_i)\rangle$ with the limit of means criterion. Every feasible and strictly D-defendable payoff profile of Γ has a subgame perfect equilibrium payoff profile that is D-defendable.*

Proof outline The strategy profile of the automata, $(atm_i)_{i \in N}$, is illustrated in Figure 5. We use the constant m^* that we now turn to estimate. After the first deviation, the sequence of punishment starts, during which, the coordinator might increase its benefit for ϱ periods of betrayal, where $0 \le \varrho \le |s' \setminus s|$. However, a suffix of the punishment sequence is guaranteed to

[7] Fifth column [14]: Clandestine group of subversive agents who attempt to undermine a nation's solidarity by any means.

On the right side of this figure, we sketch an automaton for agent $i \in \{1,2,3\}$. For brevity, the sketch merely considers a specific deviating agent $j \in \{1,2,3\}$, such that $i \neq j$. †



• *Norm* **(normal) states**

The set *Norm* includes the states q_1, q_2, and q_3. The *Norm* states emulate the cooperation sequence of Figure 3(a). The automaton starts from state q_1 and chooses its action according to the first row of the table in Figure 3(a). As long as agent j does not deviate, the automaton stays in the *Norm* states. Namely, if all agents choose their actions according to the sequence of cooperation, then from *Norm* state q_k the automaton moves to state $q_{k+1 \mod 3}$. ‡

• **The** *Norm-d* **(deviated) states**

Suppose that agent j deviates while the automaton is in a *Norm* state, q_k, where $k \in \{1,2,3\}$. In this case, the automaton moves to state q_{k+3}. The set *Norm-d* includes the states q_4 and q_5. The *Norm-d* states emulate the cooperation sequence of Figure 3(a). However, all of the *Norm-d* states lead to the punishment periods. We use $*$ to denote an arbitrary output of the automaton. Namely, regardless of the choice of the other agents, the automaton moves from state q_4 to q_5 and then to q_6, which is the starting state of the punishment.

• **The** P **(punishing) states**

The state that emulates the punishment scheme q_6, q_7, q_{m+5}. A scheme for punishing agent 3 appears in Figure 3(b). (It is easy to describe similar schemes that punish any other agent.) The automaton starts punishing agent j in state q_6 and chooses its action according to the first row of the table in Figure 3(b). Regardless of the choice of the other agents, the automaton moves from state q_k to the state $q_{k'}$, where $k \in [6, m+5]$ and $k' = k+1 \mod m+5$.

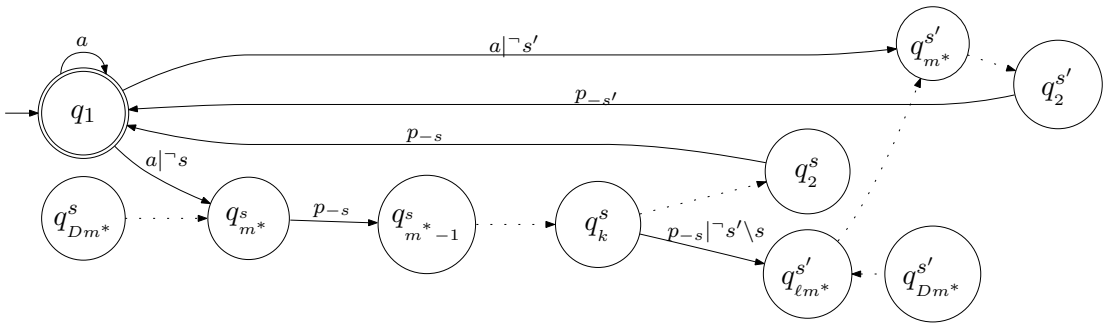
• **Estimating the constant** m^*

We note that the maximum benefit of agent j from a single deviation is 1. Moreover, a complete cycle of the cooperation sequence provides the payoff of $frac13$ (see Figure 3(a)), whereas in every period of punishment j's payoff is 0 (see Figure 3(b)). Therefore, agent j receives a benefit of $1 - fracx3$ whenever j is being punished for x periods. Thus, $m \geq 3$ is any integer that is an integral multiple of 3.

† We note that this sketch can be completed easily by considering any deviating agent. Moreover, we consider all the transitions that the figure on the right does not describe. The state q_6 (double circled) is the state to which all non-described transitions go. ‡ We define $m(\mod g)$ to be the integer q with $1 \leq q \leq \gamma$ satisfying $m = \ell\gamma + q$ for some integer ℓ (e.g., $\gamma(\mod \gamma) = \gamma$).

Figure 4: A sketch of a strategy for the shared responsibility game with 3 agents.

include $|s'|m^*$ periods in which there are no further betrayals and the automaton plays $v_i|_{s'}$. Thus, the coordinator's potential benefit is $(\gamma + \varrho)g^*$, where g^* is the maximal amount that any coordinator $j \in N$



On the right, an automaton for agent $i \in N$ is sketched. For brevity, the automaton considers: a single subordinate group, $s' \in S(D)$, † and a sequence of cooperation that consists of the repetition of a single outcome, i.e., $\gamma = 1$. ‡

• $Norm$ **(normal) states**

The automaton starts from $Norm$ state, q_1, and chooses its action according to the profile of actions, $a = (a_i)_{i \in N}$, that guarantees the strictly D-defendable outcome, w. As long as no subordinate group deviates, the automaton stays in the $Norm$ state q_1. In the case where the subordinate group s deviates from a, then the automaton moves to state $q^s_{m^*}$ from which the punishment of s begins.

• P **(punishing) states**

For each subordinate group, $d \in S(D)$, the set P includes the punishing states $\{q^d_k\}_{d \in S(D), k \in [2, Dm^*]}$. The punishment of the subordinate group $s \in S(D)$ uses a payoff profile, p_{-s} that is s-enforceable. § As long as the agents in $N' = N \setminus \{s\}$ do not deviate from the punishment scheme p_{-s}, the automaton moves from q^s_k to $q^d_{k'}$, where $k \in [2, Dm^*]$ and $k' = k + 1 \mod m^*$. ‡ Suppose that while the automaton is in state q^s_k, the agents $s' \setminus s$ betray the autonomous group N' and deviate from the punishment scheme p_{-s}. In this case, the automaton moves from state q^s_k to state $q^{s'}_{\ell m^*}$, where s' is the set of exposed deviators, and $\ell = |s'|$.

We note that: † This sketch can be easily completed by considering any subordinate group. Moreover, we consider all the transitions that the figure above does not describe. The state q_1 (double circled) is the state to which all such non-described transitions go. ‡ The more general case appears in Figure 4. § Use $m(\mod g)$ as defined in Figure 4.

Figure 5: A strategy sketch for a repeated game with n agents and the limit of means criterion.

can gain when any subordinate group $s \in S(D)$ deviates from any action profile in G. (Namely, g^* is the maximum of $u_j(a_{-s}, a'_s) - u_j(a)$ over all $j \in N$, $s \in S(D)$, $a'_s \in A_s$ and $a \in A$. Moreover, we assume that g^* is given.)

The coordinator cannot increase its benefit during the punishment suffix, which has no further betrayals. We explain how to choose a large enough m^* so that the punishment is effective. The alternative payoff of the coordinator is at least the sum of $w_j - v_j|_{s'}$ taking over all $|s'|m^*$ periods of the punishment suffix. Since w is strictly D-defendable and $s \in S(D)$, then $w|_s$ is s-enforceable and $w_j|_s > v_j|_s$ (recall $v_j|_s$ from Equation 1). Therefore, there exists an integer $m^* \geq 1$ that is an integral multiple of γ, such that for all $j \in N$ and $s' \in S(D)$:

$$g^*(\gamma + D - 1) < m^*(w_j - v_j|_{s'}). \qquad (2)$$

The proof specifies the strategy described above. Moreover the proof verifies that in the case where there are no deviations, the automaton follows the sequence of cooperation. In addition, for any non-

empty subordinate group that deviates, the automaton follows a finite and effective sequence of punishment.

We note that existing work on joint deviation in repeated games, such as [41, 1], does not bound the costs that are related to the strategy complexity. The finite automaton that is considered by Lemma 1 allows us to present such bounds (see Section 6).

5.1 Proof of Lemma 1

Proof. The strategy follows the sequences of cooperation as long as no agent deviates. In case agents deviate, the strategy follows a scheme of punishment for a finite number of periods. We start by specifying the sequences of cooperation and punishment before constructing an automaton that implements strategy. We present a sketch of the strategy that allows us to discuss key issues of concern. Then, in the full description of the strategy, we address these issues, before proving the strategy's properties.

Sequence of cooperation that yields a profile of w of average payoffs By definition, a vector of payoff profile, w, is feasible in Γ if it is a convex combination $\Sigma_{a\in A}\alpha_a u(a)$ for which the coefficients α_a are rational. (See [9] for the generalization to the case of vector with irrational numbers.) Suppose that w is strictly D-defendable payoff profile, such that $w = \Sigma_{a\in A}\alpha_a u(a)$. Moreover, suppose that $\alpha_a = \beta_a/\gamma$ for each $a \in A$, where every β_a is an integer and $\gamma = \Sigma_{a\in A}\beta_a$. We define the cooperative sequence, W, as the sequence of outcomes $(a^k)_{k=1}^{\gamma}$ in the repeated game that consists of an indefinite repetition of a cooperation sequence of length γ in which each $a \in A$ is played for β_a periods. We note that W yields an average payoff profile over the cooperation sequence, and hence in the entire repeated game, of w.

Let $s \in S(D)$ be the subordinate set, and $N' = N \setminus s$ the autonomous group that take the strategy of $(atm_i)_{i\in N'}$. We denote by $W|_s$ the set of sequences $((a_i^k)_{i\in N'})_{k=1}^{\gamma}$ in which the automata $(atm_i)_{i\in N'}$ follow the sequence of cooperation.

The sequence of punishment, p_{-s}, where $s \in S(D)$ Since w is D-dependable, then for every $s \in S(D)$ there is a s-enforceable payoff profile $v|_s$. For the sake of simplicity, we assume that $v|_s$ can be achieved using any finite number of repetitions of the profile of actions p_{-s}. We note that the generalization is straightforward; construct a sequence of action profiles, P_{-s}, in a similar manner to W. We denote by $P|_s$ the set of sequences, $p_{-s'}$, in which all of the automata $(atm_i)_{i\in N'}$ punish the subset $s' \subseteq s$ of agents $s \in S(D)$.

The strategy and its sketch The strategy is presented in Figure 6. The sketch in Figure 5 depicts the strategy as an automaton. For the reader's connivance, we illustrate keys issues of the strategy, before presenting a formal proof of these properties.
∘ *Cooperation and punishment* The strategy generates indefinite repetition of the cooperation sequence W as long as no agent deviates. Each agent punishes any deviation for a limited number of periods. We construct the strategy such that the punishment sequence begins after the last period in the sequence of cooperation in which a deviation occurred (similar to the example in Figure 4). Subsequently, the punishers return to the beginning of a cooperation sequence after the deviators have been punished.
∘ *Groups of provoking agents, and "fifth column" agents* In the definition of D-defendable payoff profiles (Section 4), we describe a social scenario in which the coordinator divides the subordinate group, $s' \in S(D)$, in two groups: a group of provoking agents, $s \in S(D)$, and a group of "fifth column" agents, $s' \setminus s \in S(D)$. The provoking agents s, deviate during cooperation sequence W, while "fifth column" agents, $s' \setminus s$ deviate not during cooperation sequence. Rather, the "fifth column" agents aim at conspiring with s and against the punishers $N \setminus (s \cup s')$. Namely, suppose that $s \cup s' \in S(D)$ and let $a_{-(s\cup s')} = b_{-(s\cup s')}(((p_{-s})_j)_{j\in N\setminus(s\cup s')})$ be an action profile of agents $(s \cup s')$ take as a best response to $((p_{-s})_j)_{j\in N\setminus(s\cup s')}$. If $\exists j \in s : u_j(a_{-(s\cup s')}) < u_j(p_{-s})$, then the conspires, $s\cup s'$, may deviate during s's sequence of punishment and make j's punishment "sweet".

We note that the coordinator may divide the subordinate group, s', to at most D groups. Namely, all agents, $j \in s$, deviate for the "first time" at different periods, $k_{j_1}, \ldots k_{j_{|s|}}$, where $k_j \in s$. At each of these "first time" deviations, the strategy is caught "unguarded". Namely, the non-deviator agents may lose some of their payoff. Nevertheless, the strategy effectively punishes the agents of s' for all deviations.

The perfection property of the strategy We show that after all the deviators have deviated at least once, the automaton take an s-enforceable payoff profile for $|s|m^*$ periods. During these periods, the deviator's benefit is washed off.

Let $atm = (atm_i)_{i\in N}$ be automata that are constructed as specified in Figure 6. Let $s \in S(D)$ be the

subordinate set and $N' = N \backslash s$ the autonomous group that take the strategy of $(atm_i)_{i \in N'}$. Let $\langle N, H, \succsim \rangle$ be the extensive form of the game Γ. Claims 1 and 2 show that atm is a perfect Nash equilibrium that is D-defendable. Claim 1 shows that any sequence of punishments has at most $m^*|s|(|s|-1)/2$ periods.

Claim 1. *Let $s' \in S(D)$ be the subordinate group. Let $h \in H$ be the history that has a suffix $h' \in W|_s$ (in which the automata $(atm_i)_{i \in N'}$ follow the sequence of cooperation). Moreover, h' is a cooperation sequence in which a non-empty set of agents $s \subseteq s'$ deviates. Then, within γ periods of h' the automata $(atm_i)_{i \in N'}$ reach the state $P(s, |s|m^*)$. Moreover, within $m^*|s'|(|s'|-1)/2$ periods from $P(s, |s|m^*)$, the automata $(atm_i)_{i \in N'}$ start a sequence of punishment $h'' \in P|_{s'}$. Furthermore, h'' is immediately followed by a sequence of cooperation.*

Proof. Without the loss of generality, suppose that the first agent to deviate does it during the first cycle of the cooperation sequence. Moreover, we assume that h starts on the first period of the cooperation sequence. We denote by k_j the index of the first period in which agent $j \in s$ deviates in h'. Moreover, without the loss of generality, we assume that $j_1 < j_2$ implies that $k_{j_1} \leq k_{j_2}$ for every $j_1, j_2 \in s'$.

The proof is followed by showing that:

Assertion 1 (The sequence of cooperation ends)**.** *On the γth period of h', automata $(atm_i)_{i \in N'}$ reach the state $P(s, |s|m^*)$.*

Let $k \geq \gamma$ be the (minimal) index of a period in h' that is before the first period in which all automata $(atm_i)_{i \in N'}$ are in state $(Norm^1, \emptyset)$. Let $d_k = \{j \in s' | k_j < k\}$ be the set of agents that have deviated before period k.

Assertion 2 (Reaching an effective sequence of punishment)**.** *Within $m^*|d_k|(|d_k|-1)/2$ periods from $P(s, |s|m^*)$ in h, all automata $(atm_i)_{i \in N'}$ are in state $P(d_k, |d_k|m^*)$.*

We note that Assertion 2 implies that once all agents in s' have deviated, then all automata $(atm_i)_{i \in N'}$ reach the state $P(s', |s'|m^*)$. By the definition of the subordinate group s' (see Section 4), we can assume, without the loss of generality, that all agents in s' deviate and all automata $(atm_i)_{i \in N'}$ reach the state $P(s', |s'|m^*)$. (In other words, we assume that $k_j - k_{j-1} \leq m^*(j-1)(j-2)/2$.)

Assertion 3 (Returning to the sequence of cooperation)**.** *Within $m^*|s|$ periods (that are after $P(s', |s'|m^*)$) in which all deviators in s' are held to their minimal payoff, all automata $(atm_i)_{i \in N'}$ are in state $(Norm^1, \emptyset)$.*

In other words, a sequence of cooperation starts after a finite sequence of punishment that is in $P|_s$.

• The sequence of cooperation ends – verifying Assertion 1.

Let $k' \leq k$ be the index in h' of any period that precedes the first deviation of agent j on the k_j-th period. By induction on k', it is possible to show that on that k'th period, atm_i's state, q_i, is $(Norm^{k'}, d)$ for any $i \in N'$, where $d_0 = \emptyset$, $d_j = d_{j-1} \cup \{j\}$ and $d \subsetneq d_j$. The arguments comes from the assumption that j deviates on the k_j-th round and the definition of $\tau_i()$ 1a. Therefore, all automata $(atm_i)_{i \in N'}$ move to state $(Norm^k, d)$, where $d \in S(D)$ and $j \in d$.

Let s be the set of agents that deviate during the (first) cooperation sequence in h'. The proof of this assertion is continued by similar arguments for each of the periods in which any agent $j \in s$ deviates for the first time in h'. The rest of the proof of this assertion is followed by the definition of $\tau_i()$; It is easy to see that item 1a of $\tau_i()$'s definition holds not before the state $P(s, |s|m^*)$ is reached (and only item 1b is applied).

• Reaching an effective sequence of punishment – verifying Assertion 2.

By the definition of $\tau_i()$, the condition in item 2a of $\tau_i()$'s definition holds only when the period index, k, equals to k_j for some $j \in s$. In all other periods k of h in which $k_j < k < k_{j+1} : j \in s' \setminus s$, the automata $(atm_i)_{i \in N'}$ move from state $P(d_j, t)$ to state $P(d_j, t-1)$ until $\tau_i()$'s item 2b holds. Once item 2b holds, all automata $(atm_i)_{i \in N'}$ move to state $(Norm^1, \emptyset)$ and the sequence of cooperation starts.

• The sequence of cooperation is reached – verifying Assertion 3.

The proof of this assertion is by the fact that the number $|s'|(|s'|-1)/2m^*$ bounds the maximal number of successive periods in which the condition in item 2a of $\tau_i()$'s definition holds. Once item 2b of $\tau_i()$'s definition holds, all automata $(atm_i)_{i \in N'}$ move to state $(Norm^1, \emptyset)$ and the cooperation sequence starts. □

Let h'' be the maximal prefix of h' that includes at most one cooperation sequence from the set $W|_s$ (i.e., periods in which any subset of s is punished). Without the loss of generality, we assume that all of the agents in s' deviate in h''. Claim 2 shows that any sequence of punishments is effective.

Claim 2. *Any coordinator $j_{coord} \in s'$ of the subordinate group $s' \in S(D)$ does not benefit from any deviation of s' from the strategy atm.*

Proof. Let k_j be the first period in h'' in which agent $j \in s'$ deviates. Without the loss of generality, we assume that $j_1 < j_2$ implies that $k_{j_1} \leq k_{j_2}$ for every $j_1, j_2 \in s'$. Let x, y, and z be the sums of j's payoffs for the different periods of h'', which we define as follows: The sum x considers the periods between the first and γth one. The sums y considers the periods between the $(\gamma+1)$th one and the periods in which all automata move to the state $P(s', |s'|m^*)$. The sum z considers all the periods in which all automata are in the states $P(s', t)$, where $t \in [1, |s'|]$. We show that for any coordinator $j_{coord} \in s'$, the total sum $x_{j_{coord}} + y_{j_{coord}} + z_{j_{coord}}$ is smaller than the sum of payoffs $w_{j_{coord}}$ taking over the period of punishment.

The proof is followed by showing that for agent $j \in s$, it holds that:

Assertion 4 (Potential benefit during the cooperation sequence)**.** $x_j \leq g^*\gamma$.

Let $k \geq \gamma$ be the index of a period in h'' that is before the first period in which all automata $(atm_i)_{i\in N'}$ are in state $(Norm^1, \emptyset)$. Let $d_k = \{j \in s' | k_j < k\}$ be the set of agents that have deviated before period k. We denote by $\xi(d_k)$ the number of periods in h'' in which the automata $(atm_i)_{i\in N'}$ follows the strategy $v_j|_{d_k}$. We denote by $\varrho = \{k_j : \gamma < k_j \leq |h''| \wedge j \in s'\}$ the number of periods in h'' in which there is an agent $j \in s' \setminus s$ that deviates for the first time while the automata $(atm_i)_{i\in N'}$ are in any of the punishment states.

Assertion 5 (Benefits and detriments between the states $P(s, |s|m^*)$ and $P(s', |s'|m^*)$)**.** $y_j \leq y_j^+ + y_j^-$, *where $y_j^+ = \varrho g^*$, and $y_j^- = \Sigma(\xi|_{d_k} - 1)v_j|_{d_k}$ (recall the definition of $v_i|_s$ from Equation 1).*

We note that by definition of g^*, we have that $y_j^+ \geq w_j\gamma$. We would like to explain why $y_j^- \leq w_j\Sigma(\xi|_{d_k} - 1)$. Since w is D-defendable, we have that $w_j \geq w_j|_{s'}$ for all $s' \in S(D)$ and $j \in s'$. Moreover, for all $s \subseteq s'$ it holds that $w_j|_s \geq w_j|_{s'}$. We note that by Equation 1, we have that $w_j \geq w_j|_{d_k} \geq v_j|_{d'_k}$ for all $d_k \subseteq d'_k$. Therefore, $y_j^- \leq \Sigma(\xi|_{d_k} - 1)v_j|_{d_k} \leq \Sigma(\xi|_{d_k} - 1)w_j$.

Assertion 6 (Profits wipeout)**.** $z_j < v_j|_{s'}m^*|s'|$.

Then, by Assertion 7, any coordinator $i_{coord} \in s'$ does not profit from a deviation of s' from the strategy profile atm.

Assertion 7 (The total sum)**.**

$$x_j + y_j + z_j < m^* w_j|s'| + w_j\Sigma\xi(d_k) \qquad (3)$$

We would like to note that the left side of Equality 3 represents the coordinators' choice to deviate. Moreover, the right side of Equality 3 represents the alternative sum of payoffs. Namely, we sum up the coordinator's payoffs during a sequence of cooperation that is in the length of the sequence of punishment.

• Potential benefit during the cooperation sequence – verifying Assertion 4.

The proof of this assertion is followed by Assertion 1 of Claim 1 and the definition of g^*.

• Benefits and detriments between the states $P(s, |s|m^*)$ and $P(s', |s'|m^*)$ – verifying Assertion 5.

The proof of this assertion is followed by Assertion 2 of Claim 1. We note that Figure 5 illustrates the sequence of punishment, which we now turn to look at. Let k be an index of a period in the sequence of punishment of h''. The coordinator, $j_{coord} \in s'$, decides if to order the subordinates to deviate in k. We note that after the subordinate, $j \in s'$, deviates, the automata $(atm_i)_{i\in N'}$ punish j. Namely, playing the strategy $v_j|_{d_k}$, where $j \in d_k$. Thus, the agents $j \in d_k$ may increase their payoff to at most g^* if the coordinator j_{coord} order a deviation. However, when coordinator j_{coord} does not order a deviation, the agents $j \in d_k$ receive the payoff $v_j|_{d_k}$.

The proof of this assertion is completed by summing in y_j^+ and y_j^- the payoffs of the periods in which the coordinator j_{coord} decide, and respectively decides not, to order the deviation of its subordinates.
The profits wipeout – verifying Assertion 6. By the definition of $\tau_i()$, we have that starting from state $P(s', |s'|m^*)$, the automata $(atm_i)_{i\in N'}$ take $|s'|m^*$ steps from state $P(s', |s'|m^*)$ to state $(Norm^1, 1)$ in which the automaton $i \in N'$ play $v_i|_{s'}$.

• **The total sum – verifying Assertion 7.**

By assertion 4, 5, and 6, Equality 3 can be rewritten as:

$$g^*\gamma + \Sigma(\xi|_{d_k} - 1)v_j|_{d_k} + \varrho g^* + m^* v_j|_{s'}|s'| < m^* w_j|s'| + \Sigma\xi(d_k)w_j, \quad (4)$$

which implies

$$g^*\gamma + \varrho g^* + m^* v_j|_{s'}|s'| < m^* w_j|s'| + \Sigma\xi(d_k)(w_j - v_j|_{d_k}) + \varrho w_j. \quad (5)$$

We wish to find the minimal value of m^*, and therefore we write:

$$g^*\gamma + \varrho g^* - \Sigma\xi|_{d_k}(w_j - v_j|_{d_k}) - \varrho w_j < m^*|s'|(w_j - v_j|_{s'}). \quad (6)$$

Since $w_j > v_j|_{s'}$, then there exists an integer $m^* \geq 1$ that is an integral multiple of γ and satisfies Equation 6. Moreover, since $\varrho \leq D - 1$ and $|s'| = D$, then we can take an integer $m^* \geq 1$ that is an integral multiple of γ and satisfies Equation 2.

Hence, the lemma. □

□

6 Strategy Complexity

Computational game theory has several ways of measuring complexity of games (see [38]). In Section 1, we mentioned the computational complexity of games with subsystem takeovers. We now turn to consider the strategy complexity of these games. Kalai and Stanford [33] define the complexity of an individual strategy as follows. Let $(st_i)_{i\in N}$ be a subgame perfect equilibrium. Then, the complexity of an individual strategy, st_i, is the number of distinct strategies, $|\{st_i|_h : h \in H\}|$, induced by st_i in all possible subgames. The size of an automaton is the cardinality of its state set. Kalai and Stanford [33] show that the complexity of a strategy equals the size of the smallest automaton that can implement the strategy.

Existing work on joint deviation in repeated games, such as [41, 1], does not bound the costs that are related to the strategy complexity. Rubinstein [41] adds to Aumann's n-strong Nash equilibria (see [6]) the perfection property for the limit of means criterion. Rubinstein [41] presents an inductive definition of the strategy that, on the t-th period of the game, considers all the deviations during the last T periods of the game. We note that T is not explicitly bounded. Moreover, in order to decide on the next action of the strategy, the definition of the strategy in [41] considers all the payoffs of all the subsets, $s \in 2^N$, of agents during the last T periods. Our automaton uses a (constant time) transition function for deciding on the next action that the strategy should take. Moreover, we explicitly bound the number of periods in which the agents in s are punished.

Abraham et al. [1] consider t-resilient Nash equilibrium (where $1 \leq t < n$) and presents mechanisms for finitely repeated games that use poly-time (or probabilistic) Turing machine, which can be emulated by finite automata. The reduction increases the number of states that the automaton uses by a non-polynomial factor. We consider simpler implementations.

Lemma 2 (Corollary 2)**.** *The complexity of a strategy that deters subordinates' deviations is in* $\Theta(D\binom{n}{D})$*, where n is the number of agents, and D is an upper bound on the size of the subordinate group.*

Proof outline A strategy that deters subordinates' deviations is presented in Section 5. The automaton that implements these strategies requires $O(D\binom{n}{D})$ states (for the case of $D \leq \frac{n}{2}$). The lower bound part of this lemma is demonstrated by considering every subordinate group, $s \in S(D)$, and all the possible sequences of deviations. There are at least $\binom{n}{D} - 1$ subordinate groups. The proof verifies the existence of at least D different periods in which the deviators may deviate before all of them deviate. Only after the last deviation, can the strategy complete the punishment of the subordinate. Therefore, there are at least $D(\binom{n}{D} - 1)$ different subgames in which a subordinate group deviates. Thus, by [33] the strategy complexity is in $\Theta(D\binom{n}{D})$.

6.1 Proof of Lemma 2

The proof of Lemma 2 is followed by Observation 1 and claims 3, 4, and 5.

Claim 3. *Let us consider the automaton that is presented in Figure 6. The number of states that the automaton has is in* $O((\gamma + m^*D)\binom{n}{D})$.

• **Set of states Q_i:**

$\{(Norm^k, d)$: either $k = 1$ and $d = \emptyset$ or $2 \leq k \leq \gamma$ and $d \in \{\emptyset\} \cup S(D)\} \cup \{P(d,t) : d \in \{\emptyset\} \cup S(D)$ and $1 \leq t \leq Dm^*\}$. [†] (The state $(Norm^k, \emptyset)$ means that we are in the kth period of the cooperation sequence and no agent deserves punishment. The state $(Norm^k, s)$ means that we are in the kth period of the cooperation sequence and the agents in s deserves punishment. The state $P(s,t)$ (where $s \neq \emptyset$) means that the agents in s is being punished and there are t periods left in which he has to be punished. The state $P(\emptyset, t)$ means that a transient fault have happened and there are t periods left to recovery.)

• **Initial state:**

$q_i^0 = (Norm^1, \emptyset)$.

• **Output function $f_i()$:**

In $(Norm^k, s)$ for any $s \in \{\emptyset\} \cup S(D)$ choose $(a^k)_i$; in $P(s,t)$ choose $(p_{-s})_i$ if $i \notin s \wedge s \neq \emptyset$, and $b_i(p_{-s})$ if $i \in s \wedge s \neq \emptyset$. For the case of $s = \emptyset$, we define $p_{-\emptyset}$ to be t-defendable Nash equilibrium in G, where $t \geq D$ is maximal. [‡] We note that $\forall i \in N \wedge s \in S(D)$ it holds that $u_i(p_{-s}) \geq u_i(p_{-\emptyset})$ and therefore no agent can profit by deviating (and getting punished) by having $(p_{-\emptyset})_i$ played for Dm^* periods.

• **Transition function $\tau_i()$:**

To simplify the presentation, we define $\mathtt{@s}_i \in \{\emptyset\} \cup S(D)$ to be the set of deviating agents in automaton atm_i's point-of-view when the automata output the action profile $a \in A$ and atm_i's state is $q_i \in Q_i$. Namely, we require that $\mathtt{@s}_i$ is a maximal subset of N, such that $((f_j(q_j))_j)_{j \in N \setminus \mathtt{@s}_i} = (a_j)_{j \in N \setminus \mathtt{@s}_i}$.

1. From $(Norm^k, d)$ move to $(Norm^{k+1(\mod \gamma)}, d)$ unless: [§]
   (a) When $\mathtt{@s}_i \cup d \notin S(D)$, or agent $j \in N$ plays $(p_{-d'})_j$ (where $d' \in \{\emptyset\} \cup S(D)$) move to $P(\emptyset, Dm^*)$.
   (b) When $\mathtt{@s}_i \cup d \in S(D)$ move to $(Norm^{k+1(\mod \gamma)}, \mathtt{@s}_i \cup d)$ if $k < \gamma$, to $(Norm^1, \emptyset)$ if $k = \gamma \wedge \mathtt{@s}_i \cup d = \emptyset$, and to $P(\mathtt{@s}_i \cup d, |\mathtt{@s}_i \cup d| m^*)$ if $k = \gamma \wedge \mathtt{@s}_i \cup d \neq \emptyset$.
2. From $P(d,t)$ move to $P(d, t-1)$ unless:
   (a) When $\mathtt{@s}_i \not\subseteq d$ move to $P(\emptyset, Dm^*)$ if $\mathtt{@s}_i \cup d \notin S(D) \vee d = \emptyset$, and to $P(\mathtt{@s}_i \cup d, |\mathtt{@s}_i \cup d| m^*)$ if $\mathtt{@s}_i \cup d \in S(D)$.
   (b) When $t = 1 \wedge (d = \emptyset \vee \mathtt{@s}_i \subseteq d)$ move to $(Norm^1, \emptyset)$.

[†] We use m^* as defined in Figure 5. [‡] For example, in the shared responsibility game we have that $t = n$ (when all autonomous agents choose to be Clients). [§] We use $m(\mod g)$ as defined in Figure 4.

Figure 6: A strategy for a repeated game with n agents and the limit of means criterion.

Proof. There are γ states of the type $Norm$. The states of the type $Norm - d$ can be divided into the groups $\{(Norm^k, d) : d \in S(D) \wedge 1 < k \leq \gamma\}$. I.e., there are $O(\binom{n}{D})$ groups and $\gamma - 1$ members in each. The states of the type P can be divided into the groups $\{P(d,k) : d \in S(D) \wedge 1 < k \leq Dm^*\}$. I.e., there are $O(\binom{n}{D})$ groups and Dm^* members in each. The proof is completed by a summery of the number of states in every type. □

Observation 1 explains some aspect of strategies'

perfection (see [43]). Let $s \in S(D)$ be the group of deviators. We consider the different sequences of punishment for s. We say that a sequence of punishment $P'|_s$ is maximal, if there is no sequence of punishment $P|_s$, such that $\forall i \in N' = N \setminus s$ we have that $P'|_s \prec_i^* P|_s$ in Γ.

Observation 1. *Let st_i be a perfect strategy for deterring subordinate deviations in the game Γ. Then, st_i induces only maximal sequences of punishment.*

Proof. The perfect strategy punishes s for a finite number of punishment periods. We note that there are many possible punishments schemes: There are punishment schemes in which "only" the agents in s are punished. Alternatively, there are schemes that take the form of a "collective punishment". Namely, the non-deviating agents can punish $j \in s$ by punishing a group of non-deviating agents $d \not\subseteq s$, such that $j \in d$.

Let us consider the example of the shared responsibility game. The punishment can be a finite number of periods in which all non-deviating agents decide to be clients. This way s's minimal payoff is guaranteed and the automata size is reduced, because the scheme employs merely one sequence of punishment. However, from the perspective of an agent that does not deviate, the cost of any "collective punishment" is higher than the cost of the punishment in which just the agents in s are excluded. This is what makes the "collective punishment", $P|_s$, unbelievable. □

Claim 4. *Consider the shared responsibility game. Let s and s' be two different subordinate groups in $S(D)$, and let their corresponding maximal sequences of punishments be $P|_s$ and $P'|_s$. Then, $P|_s$ and $P|_{s'}$ are different.*

Proof. Without the loss of generality, suppose there exists agent $j \in s \setminus s'$. By the effectiveness of the punishment scheme $P|_s$, the agents of $N' = N \setminus s$ disconnect agent j from the communication graph in $P|_s$. By the maximality of $P|_{s'}$, the agents of $N' = N \setminus s$ do not disconnect j in $P|_{s'}$. Hence, the claim. □

Claim 5. *Let G be a game with symmetrical payoff matrix, and Γ a game that infinitely repeats G. The complexity of perfect strategies that deter subordinates' deviations in Γ is in at least $D(\binom{n}{D} - 1)$.*

Proof. The proof is followed by counting the number of possible subgames that starts at the first period of the cooperation sequence, present a single sequence of punishment, and then returns to the first period in the sequence of cooperation.

There are at leat $\binom{n}{D} - 1$ different subordinate groups. Then, by Observation 1 and Claim 4, we have that each subordinate group can induce a different sequence of punishment. Given a subordinate group, s, we show that there are $|s|$ different subgames in which any non-empty subset of s deviates. Therefore, by Kalai and Stanford [33], we have that the strategy complexity is at leat $D(\binom{n}{D} - 1)$.

Suppose that agent $j_0 \in s$ deviates during the sequence of cooperation. Moreover, suppose that agents $j_1, j_2 \in s$ deviate in different periods of the (first) sequence of punishment that follows j_0's deviation. By Observation 1, only maximal sequence of punishment are used. Therefore, we have that the subgame in which j_1 deviates before j_2 does, is different than the subgame in which j_1 deviates after j_2 does.

We can generalize the above arguments to any non-empty subset of agents in $s \setminus \{j_0\}$ and every permutation of that subset by induction on the cardinality of subset. Therefore, every permutation of s induces a different subgame.

We note that by the symmetry of G's payoff matrixes, we have that the order in which the deviators $j_i \in s$ join the punishment set, $d \subseteq s$ does not change the sequence of punishment. Namely, for all $s \in S(D)$, we have that all punishment sequences $p_{\{j_1\}}, p_{\{j_1,j_2\}}, p_{\{j_1,j_2,\ldots\}}, \ldots$ are identical for all permutations $j_1, j_2, \ldots$ of $s \setminus \{j_0\}$. Hence, there are at least $|s|$ different subgames that corresponds to s. □

7 Self(ish)-Stabilization

Lemma 3 extends Lemma 1 by showing that the automata can be made to recover from transient faults.

Lemma 3 (Corollary 3)**.** *Let Γ be an infinitely repeated game of G under the limit of means criterion. Then, there are self(ish)-stabilizing automata that implement subgame perfect equilibria that are D-defendable in Γ.*

We consider two proofs of Lemma 3: a simple one the uses known techniques of clock (state) synchronization and a more general one. Dolev [22] presets clock synchronization algorithms that within a finite

number of steps after the last transient fault synchronize the clock values. One can view the clock values as the state of the automata. Namely, requiring identical state values of the autonomous automata is the same as requiring the same time values in all clocks. This simple proof assumes that all states are distinguishable, e.g., all automata send their state number in every period. This assumption implies that during the punishment period all automata have to communicate and that there is no "punishment by silence". Sometime this assumption is too restrictive. For example, in some applications it might be required that the autonomous agents do not communicate or interact during the sequence of punishment. In this case, no communication implies that the states are indistinguishable. Therefore, we also consider a more general proof in which only the first state of any punishment sequence is distinguishable. For the sake of brevity, we present here the proof outline, and the complete and general proof of Lemma 3 can be found in [27] (and in Section 9 of the Appendix).

Proof outline Let us construct an additional sequence of punishment using the states $P(\emptyset, k)$, where $k \in [1, Dm^*]$. In these states the automaton plays according to a D-defendable Nash equilibrium. Without the loss of generality, suppose that the states $P(d, Dm^*) : d \in S(D) \cup \{\emptyset\}$ are distinguishable from all the other states. [8]

Self-stabilization requires the properties of closure and convergence that can be verified by a variant function (see [22]). Every step of the automata monotonically decreases the value of the variant function until it reaches zero, which implies the end of the stabilization period. In order to define the potential function, we represent the automata as directed graphs.

• The automaton as a graph

The graph $\Phi = (V, E)$ has the set of states as the set of vertices, V. Moreover, the transitions function, $\tau()$, induces directed edges of E, i.e., $(v, u) \in E \leftrightarrow \exists a \in A : \tau(v, a) = u$.

[8] This assumption can be implemented, for example, by letting all selfish-computers to broadcast the indices of their current states at the end of every period. We note that any additional costs that the broadcast induces can be compensated by selecting a larger m^*.

• The variant function

We define $\mathsf{LongDistance}(q_j)$ to be the length of the maximal simple and directed path in graph Φ, from state q_j to state $P(\emptyset, Dm^*)$. (A simple and directed path is one without any directed cycles.) We define the variant function $\Phi()$ to be 0 if all automata $(atm_i)_{i \in N'}$ are in the same state, where $s \in S(D)$ is the subordinate group and $N' = N \setminus s$. For all other cases, we define $\Phi(c) = \max_{j \in N'} \mathsf{LongDistance}(q_j)$. It can be observed in Figure 5 that $0 \leq \Phi(c) \leq (\gamma + m^* D^2)$.

• Closure

Suppose that $\Phi() = 0$, which mean that automata $(atm_i)_{i \in N'}$ are in the same state. Since the automata are deterministic they all move to the same state, and $\Phi() = 0$ holds.

• Convergence

The proof verifies this property by showing that all steps decrease the value of $\Phi()$. Let us construct the automaton, such that all undefined transitions move to state $P(\emptyset, Dm^*)$. In particular, the automaton moves to state $P(\emptyset, Dm^*)$ when more than D deviators are observed. The proof verifies that if automaton $atm_i : i \in N'$ is in any punishing state, then all automata $(atm_i)_{i \in N'}$ move to state $P(\emptyset, Dm^*)$, and stay in $P(\emptyset, Dm^*)$, until all automata $(atm_i)_{i \in N'}$ move to state $P(\emptyset, Dm^*)$.

We follow the spirit of Kalai and Stanford [33] and define the strategy complexity of a self-stabilizing strategy, st_i, as the number of distinct strategies induced by st_i in all possible subgames that start *after* stabilization. In that sense, Lemma 3 shows a self-stabilizing strategy that has asymptotically the same complexity as the non-self-stabilizing one (presented in Lemma 1).

8 Related work

The solution concepts of strong Nash equilibrium [5, 41, 6] aims at deterring a coalition of deviators that may all benefit from their joint deviations. Moreover, the solution concepts of resilient Nash equilibrium [1] aims at deterring a coalition of deviators that may

increase the payoff of at least one deviator, but committed to keep the benefits of all the other deviators. We mention that coalition-proof strategies consider agents that can communicate prior to play, but cannot reach binding agreements (see [12, 36]). In the context of repeated games, the collective dynamic consistency (of coalition-proof strategies) considers equilibria for which agents do not wish to jointly renegotiate throughout the course of the game (see [11]). This work considers harder deviations, in which the coordinator benefit and the subordinates may lose payoff. Therefore, our strategy can deter the deviations that are mentioned above.

Self(ish)-stabilization [24, 23, 19, 25, 16] was earlier considered for single stage games. The game authority [24, 23, 25] verifies that no agent violates the game model of the stage game. Spanning trees among selfish parties are studied by [19]. Reactive systems that are inspired by game theory appear in [16].

The research path of BAR fault tolerance systems [4] studies cooperative services that span multiple administrative domains, such as: a backup service [3], a peer-to-peer data streaming application [35], and Synchronous Terminating Reliable Broadcast [17]. BAR fault tolerance systems consider a minority of Byzantine computers that deviate arbitrarily and a single selfish deviator (out of the set of all selfish-computers). Between every pair of selfish-computers, the grim trigger strategy is used, which suffers primarily from the inability to recover from transient faults (see [10]). In other words, an agent that (involuntarily) deviates once is punished forever. We consider the more realistic model of infinitely repeated games, in which any group of D agents can always deviate. We offer a more sensible solution; the system punishes the deviators for a bounded period after the last betrayal. This type punishment better fits the cases of non-deliberate misbehavior of selfish-computers and transient faults.

8.1 Discussion

• Why the model of repeated games is considered?

In distributed systems, single stage games reflect tasks that are less common compared to settings of infinitely repeated games. Repeated games are best-known for their ability to model cooperation among selfish agents. For example, the perfect folk theorem (see [9, 42]) presents a strategy where its payoff in infinitely repeated games is better than the payoff of the single stage Nash equilibrium. The theorem can explain periods of war and peace among selfish agents that can deviate unilaterally. For this reason, the model of repeated games is regarded as more realistic than the model of single stage games.

• Why using DFA and not Turing machines?

Deterministic and finite automata (DFA) can implement the strategies of the folk theorem (see [39], and references therein). The literature considers strategies that can be implemented by deterministic and finite automata as a separate and "simpler" class of strategies (see [8, 40]). In fact, there is evidence that this class is strictly weaker than the class of strategies that Turing machines can implement (see the survey [30] and references therein).

We note that some of the existing results (such as [1, 2]) consider poly-time (or probabilistic) Turing machine, which can be emulated by finite (or probabilistic) automata. The reduction increases the number of states that the automaton uses by a non-polynomial factor. We present simpler implementations.

• Why not to consider coalitions in which all agents are faulty? [3]

Eliaz [28] and later [17, 3, 35, 18, 4] consider coalitions in which all of the deviators may possibly be faulty. [3] The inherent difficulty is that no punishment deters a coalition in which all agents are Byzantine. In this case, the literature proposes either to use strategies for single stage games, or grim trigger strategies.

In distributed systems, single stage games reflect tasks that are less common compared to settings of infinitely repeated games. Lack of credibility is the Achilles' heel of grim trigger strategies; deviating agents are forever punished due to mistakes that are made at the moment of weakness. Furthermore, the system cannot recover from transient faults in these settings.

We assume that a single rational agent controls a set of deviators and propose a perfect strategy that deters the deviators with a finite period of punishment. Thus, in the context of self-stabilization it is essential to require that not all deviators are faulty. [3]

• Why not to consider coalitions in which all agents are rational?

A coalition in which all deviators are rational is required to promote (or at least protect) the benefit of its members (see [5, 41, 6, 1, 12, 36], and references therein). This is not the case with subsystem takeovers; here the coordinator dictates the actions of its subordinates and ignores their benefits. Therefore, by assuming that not all deviators are rational, it is "harder" for the autonomous (non-deviating yet selfish) agents to protect their benefits, because the requirements regarding joint deviations are explicitly less restrictive.

We do not claim to be the first to consider strategies for protecting the benefit of the autonomous (non-deviating yet selfish) agents (see [1, 5]). However, we present strategies for protecting the benefit of autonomous agents in the presence of deviating coalitions that do not protect the social benefit of all deviators. It is important to see that previous works [1, 5] consider strategies for protecting the benefit of autonomous agents in the presence of deviating coalitions that indeed protect the social benefit of all deviators.

• Are there strategies for coping with more than one rational deviator within the subordinate group?

Our definition of subsystem takeovers has a straightforward extension that considers collations of k rational agents that collectively and totally control t subordinate agents. For example, the rational agent 1 controls the subordinating agents 1_1, 2_1 and 3_1, and the rational agent 2 controls subordinating agents 1_2, 2_2 and 3_2. Another example is when agents 1 and 2 reach an agreement about the behavior of their subordinates. Our strategies can deter such deviations because we consider an arbitrary coordinator and punish the entire subordinate group.

Generally speaking, given an integer $t \in [1, |N|]$, we have that a t-*defendable* Nash equilibrium is a t-*resilient* Nash equilibrium, and a t-*resilient* Nash equilibrium is a t-*strong* Nash equilibrium. Also, let $\mathcal{X}$ be any of the properties *defendable*, *resilient*, and *strong*. Then, for any $t \in [2, |N|]$, we have that a $(t+1)$-$\mathcal{X}$ Nash equilibrium is also a t-$\mathcal{X}$ Nash equilibrium. Therefore, a 1-$\mathcal{X}$ Nash equilibrium [37] is the conventional Nash equilibrium, and n-$\mathcal{X}$ Nash equilibrium is the conventional strong Nash equilibrium [6, 41].

• Are the assumptions on synchrony and observable actions holds in distributed system?

These are well-known settings that can be realized; every period can be defined to be sufficiently long to allow the stabilization of the underlying protocols (i.e., the actions' implementation). This behavior can be facilitated by game authority [24, 23, 25] in which a self-stabilizing Byzantine clock synchronization protocol periodically restarts a Byzantine agreement protocol. The agreement explicitly facilitates observable actions, and the synchronization protocol overcomes timing failures.

8.2 Conclusions

Decentralized systems consisting of selfish-computers are becoming part of reality; new aspects of these systems need to be exposed and studied. One such an example is subsystem takeover of a selfish-computer over other computers by the use of hostile malware. Game theory does not consider this type of joint deviation.

We investigated infinitely repeated games in the presence of an arbitrary deviator that controls an unknown group of subordinates. We consider infinitely repeated games with the limit of the mean criterion. Interestingly, the strategy for deterring subordinates' deviations, in such games, is simple; it can be described by deterministic and finite automaton. We discover that the number of states of the automaton is in $\Theta(D\binom{n}{D})$, where n is the number of agents, and D is an upper bound on the size of the subordinate group.

In practice, high performance communication networks process communication by dedicated fast hardware. The hardware is essentially an implementation of a deterministic and finite automaton. Therefore, the hardware may cope with a predefined number of subordinates, D, which is a bound on the size of a subordinate group. In the very rare cases in which D exceeds the designed bound, the automata will not act as desired; due to the loss of synchronization, the automata may never recover.

Therefore, we must consider the case in which the system resources are eventually exhausted, e.g., when an unexpected number of computers deviate. We address this problem by designing self-stabilizing automata that recover once the system returns to follow the designer's original assumptions. Interestingly, the self-stabilization design criteria provide an ele-

gant way for designing decentralized autonomous systems.

REFERENCES

[1] I. Abraham, D. Dolev, R. Gonen, and J. Y. Halpern. Distributed computing meets game theory: robust mechanisms for rational secret sharing and multiparty computation. In E. Ruppert and D. Malkhi, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing, PODC 2006, Denver, CO, USA, July 23-26, 2006*, pages 53–62. ACM, 2006.

[2] I. Abraham, D. Dolev, and J. Y. Halpern. Lower bounds on implementing robust and resilient mediators. In R. Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 302–319. Springer, 2008.

[3] A. S. Aiyer, L. Alvisi, A. Clement, M. Dahlin, J.-P. Martin, and C. Porth. BAR fault tolerance for cooperative services. In A. Herbert and K. P. Birman, editors, *Proceedings of the 20th ACM Symposium on Operating Systems Principles 2005, SOSP 2005, Brighton, UK, October 23-26, 2005*, pages 45–58. ACM, 2005.

[4] L. Alvisi. BAR - where distributed computing meets game theory. In A. Bondavalli, F. V. Brasileiro, and S. Rajsbaum, editors, *Dependable Computing, Third Latin-American Symposium, LADC 2007, Morella, Mexico, September 26-28, 2007, Proceedings*, volume 4746 of *Lecture Notes in Computer Science*, pages 235–236. Springer, 2007.

[5] N. Andelman, M. Feldman, and Y. Mansour. Strong price of anarchy. In N. Bansal, K. Pruhs, and C. Stein, editors, *SODA*, pages 189–198. SIAM, 2007.

[6] R. J. Aumann. Acceptable points in general cooperative n-person games. *Contributions to the Theory of Games*, 4:287–324, 1959.

[7] R. J. Aumann. The core of a cooperative game without side payments. *Transactions of the American Mathematical Society*, 98(3):539–552, 1961.

[8] R. J. Aumann et al. Survey of Repeated Games. *Essays in Game Theory and Mathematical Economics in Honor of Oskar Morgenstern*, 4, 1981.

[9] R. J. Aumann and L. S. Shapley. *Long-term Competition: A Game-theoretic Analysis*. Dept. of Economics, Los Angeles University of California, 1992.

[10] R. Axelrod. On Six Advances in Cooperation Theory. *Analyse and Kritik*, 22(1):130–151, 2000.

[11] B. Bernheim and D. Ray. Collective Dynamic Consistency in Repeated Games. *Games and Economic Behavior*, 1(4):295–326, 1989.

[12] B. Bernheim and M. Whinston. Coalition-proof Nash equilibria. II. Applications. *Journal of Economic Theory*, 42(1):13–29, 1987.

[13] C. Borgs, J. Chayes, N. Immorlica, A. Kalai, V. Mirrokni, and C. Papadimitriou. The Myth of the Folk Theorem. Report 07-082, Electronic Colloquium on Computational Complexity (ECCC), 2007. ISSN 1433-8092, 14th Year, 82nd Report.

[14] E. Britannica. *Encyclopaedia Britannica Online*. Encyclopaedia Britannica, 2004.

[15] J. E. Burns, M. G. Gouda, and R. E. Miller. Stabilization and pseudo-stabilization. *Distrib. Comput.*, 7(1):35–42, 1993.

[16] H. Cao and A. Arora. Stabilization in dynamic systems with varying equilibrium. In T. Masuzawa and S. Tixeuil, editors, *Stabilization, Safety, and Security of Distributed Systems, 9th International Symposium, SSS 2007, Paris, France, November 14-16, 2007*, volume 4838 of *Lecture Notes in Computer Science*, pages 67–81. Springer, 2007.

[17] A. Clement, J. Napper, H. Li, J.-P. Martin, L. Alvisi, and M. Dahlin. Theory of BAR games architecture. technical report TR-06-63, The University of Texas at Austin, Department of Computer Sciences., November 29 2006.

[18] A. Clement, J. Napper, H. C. Li, J.-P. Martin, L. Alvisi, and M. Dahlin. Theory of BAR games. In Gupta and Wattenhofer [29], pages 358–359.

[19] A. Dasgupta, S. Ghosh, and S. Tixeuil. Selfish stabilization. In A. K. Datta and M. Gradinariu, editors, *Stabilization, Safety, and Security of Distributed Systems, 8th International Symposium, SSS 2006, Dallas, TX, USA, November 17-19, 2006*, volume 4280 of *Lecture Notes in Computer Science*, pages 231–243. Springer, 2006.

[20] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a nash equilibrium. In Kleinberg [34], pages 71–78.

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

[22] S. Dolev. *Self-stabilization*. MIT Press, Cambridge, MA, USA, 2000.

[23] S. Dolev, E. M. Schiller, and P. Spirakis. Game authority: for robust distributed selfish-computer systems. Technical report, Dynamically Evolving, Large-scale Information Systems (DELIS), 2006. Accessible via `http://delis.upb.de/docs/`.

[24] S. Dolev, E. M. Schiller, P. G. Spirakis, and P. Tsigas. Game authority: for robust distributed selfish-computer systems. Technical report, Department of Computer Science and Engineering, Chalmers University of Technology and Goteborg University, March 2006.

[25] S. Dolev, E. M. Schiller, P. G. Spirakis, and P. Tsigas. Game authority for robust andscalable distributed selfish-computer systems. In Gupta and Wattenhofer [29], pages 356–357.

[26] S. Dolev, E. M. Schiller, P. G. Spirakis, and P. Tsigas. Strategies for repeated games with subsystem takeovers implantable by deterministic and self-stabilizing automata. In A. Manzalini, editor, *Autonomics*, volume 302 of *ACM International Conference Proceeding Series*, page 3. ACM, 2008.

[27] S. Dolev, E. M. Schiller, P. G. Spirakis, and P. Tsigas. Strategies for repeated games with subsystem takeovers implantable by deterministic and self-stabilizing automata (preliminary version). Technical Report 2008:11, Department of Computer Science and Engineering, Chalmers University of Technology and Göteborg University, April 2008.

[28] K. Eliaz. Fault Tolerant Implementation. *Review of Economic Studies*, 69(3):589–610, 2002.

[29] I. Gupta and R. Wattenhofer, editors. *Proceedings of the Twenty-Sixth Annual ACM Symposium on Principles of Distributed Computing, PODC 2007, Portland, Oregon, USA, August 12-15, 2007*. ACM, 2007.

[30] J. Y. Halpern. Computer science and game theory: A brief survey. *CoRR*, abs/cs/0703148, 2007.

[31] S. Hart. Robert Aumann's Game and Economic Theory. *Scandinavian Journal of Economics*, 108(2):185–211, 2006.

[32] A. Hayrapetyan, É. Tardos, and T. Wexler. The effect of collusion in congestion games. In Kleinberg [34], pages 89–98.

[33] E. Kalai and W. Stanford. Finite Rationality and Interpersonal Complexity in Repeated Games. *Econometrica*, 56(2):397–410, 1988.

[34] J. M. Kleinberg, editor. *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21-23, 2006*. ACM, 2006.

[35] H. C. Li, A. Clement, E. L. Wong, J. Napper, I. Roy, L. Alvisi, and M. Dahlin. BAR gossip. In *7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6-8, Seattle, WA, USA*, pages 191–204. USENIX Association, 2006.

[36] D. Moreno and J. Wooders. Coalition-Proof Equilibrium. *Games and Economic Behavior*, 17(1):80–112, 1996.

[37] J. Nash. Equilibrium point in n-person games. *Proc. Nat. Acad. Sci. USA*, 36:48–49, 1950.

[38] N. Nisan, T. Roughgarden, É. Tardos, and V. V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press New York, NY, USA, 2007.

[39] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[40] D. Pearce. Repeated games: cooperation and rationality. *Advances in Economic Theory: Sixth World Congress*, 1992.

[41] A. Rubinstein. Strong perfect equilibrium in supergames. *International Journal of Game Theory*, 9(1):1–12, 1980.

[42] A. Rubinstein. *Essays in Game Theory in Honor of Michael Maschler*, chapter Equilibrium in supergames. Springer-Verlag, 1994. N. Meggido, editor.

[43] R. Selten. Spieltheoretische Behandlung eines Oligopolmodells mit Nachfragetragheit. *Zeitschrift fur die gesamte Staatswissenschaft*, 12:301–324, 1965.

Appendix

9 Proof of Lemma 3

We start by formally defining self-stabilizing systems before presenting complementary definitions. Then, we turn to prove Lemma 4.

9.1 Self-stabilizing systems

System executions We describe the global state of the system, the system *configuration*, by the vector of the state of the automata $\langle q_1, q_2, \cdots q_n \rangle$. We describe the system configuration in the instance in which the pulse is triggered, when there are no messages in transit. We define an *execution* (or run) $R = c_0, c_1, \ldots$ as a sequence of system configurations c_i, such that each configuration c_{i+1} (except the initial configuration c_0) is reached from the preceding configuration c_i by an execution of steps by all the automata. [9]

The task of self(ish)-stabilizing automata The *self(ish)-stabilization* tasks are characterized by maximizing the payoffs of selfish-computers in self-stabilizing systems (see [19, 16, 23]). In this work, we consider the payoff of subordinate groups in the repeated game $\Gamma = \langle N, H, \succsim^\star = (\succsim_i^\star) \rangle$.

[9] **Histories vs. executions:** We terms histories and executions are different. Recall that a history is a sequence of action profiles, whereas an execution is a sequence of configurations.

The *task* τ of a distributed system is defined by a set of executions LE called *legal* executions. We define legal executions in which the automata $(atm_i)_{i \in N}$ implement subgame perfect equilibria that are D-defendable in Γ. The definition of the set $LE = \cup_{s \in S(D)} LE(s)$ includes the executions from the sets $LE(s)$, where $s \in S(D)$ is a subordinate group. The definition of $LE(s)$ considers all possible executions in which the autonomous group of automata $(atm_i)_{i \in N'}$ (where $N' = N \setminus s$) implements a profile of strategies, such that the profile of strategies $(atm_i)_{i \in N}$ is a subgame perfect equilibrium that is D-defendable, and $(atm_i)_{i \in s}$ is a profile of arbitrary strategies.

Namely, $R \in LE(s)$ if, and only if, for every two consecutive configurations, $c, c' \in R$ (where $c = \langle q_1, \cdots q_n \rangle$ and $c' = \langle q'_1, \cdots q'_n \rangle$), and every action profile $(a_{-s}, a_s) \in A$ (which are the steps that all the automata take from c to c'), we have that: (1) $(f_i(q_i))_{i \in s} = a_{-s}$ is the output of automaton $(atm_i)_{i \in N'}$, and (2) $\tau_i(q_i, (a_{-s}, a_s)) = (q'_i)_{i \in N'}$. [10]

9.2 Correctness proofs

The correctness of a self-stabilizing system is demonstrated by considering every execution that follows the last transient fault and is, therefore, proved assuming an arbitrary starting configuration. The system should exhibit the desired behavior (of task τ) in an infinitely long period after a finite convergence period.

A configuration c is *safe* with regard to a task τ and to the distributed system if every nice execution that starts from c belongs to LE. We say that the system satisfies its task τ when its execution is in LE. A system is *(strongly) self-stabilizing* if, starting from an arbitrary configuration, it reaches a safe configuration within a finite convergence period.

A system is *(pseudo) self-stabilizing* [15, 22] if, starting from an arbitrary configuration, it reaches a safe configuration within a finite convergence period, and exhibit the desired behavior (of task τ) in an infinitely long period of legal behavior; but may stray from this legal behavior a finite number of times.

We note that the proof of a self-stabilizing system demonstrates both convergence property and the closure property.

[10] **Minority groups with arbitrary strategies:** We note that $a_s \in A_s$ is an arbitrary output that the subordinate group, $(atm_i)_{i \in s}$, takes from c to c'. Moreover, $(q'_i)_{i \in s}$ is a profile of arbitrary states of the subordinate group.

9.3 Complementary definitions

Distinguishable states One way to regain system consistency after a transient fault is to perform a global reset. The global reset technique requires all automata to set their state to a unique state and to stay in that state until system consistency is achieved. We note that assuming that all of the actions of the automata encode the states of the automata implies that the game model should consider the communication costs of the state related information. We wish not to unnecessarily restrict the possible set of game models and therefore we specify our assumptions regarding to the state related information.

We say that the set of states $Q'_i \subset Q_i$ is distinguishable (from $Q_i \setminus Q'_i$) if, and only if, $\forall q'_i \in Q'_i \wedge q_i \in Q_i \setminus Q'_i$ it holds that $f_i(q'_i) \neq f_i(q_i)$. We note that any set of states can be made distinguishable by letting the automata communicate the states' indices. However, additional communication costs may change the game model. We add the following assumptions to the automata presented in Figure 6. We assume that each of the states $P(d, m^*)$ (where $d \in \{\emptyset\} \cup S(D)$) is a distinguished one, and that the set $\{P(d,t) | d \in \{\emptyset\} \cup S(D) \wedge 1 \leq t \leq m^*\}$ is distinguishable as well.

We note that in case that additional communications costs are require for allowing the punishment states to be distinguishable, then larger values of m^* could compensate these overheads. Moreover,

$$\forall i, j \in N \wedge q_i \in Q_i \wedge q_j \in Q_j, \text{ and } q_i \text{ is distinguishable from } q_j, \text{ we have that } i \in \mathtt{@s}_j. \quad (7)$$

9.4 The proof

Lemma 4 (Self(ish)-Stabilization)**.** *Let Γ be an infinitely repeated game of G. Then, there exists self(ish)-stabilizing automata that implement a subgame perfect equilibrium that is D-defendable of the limit of means criterion of Γ.*

Proof. We show that the automata constructed in Figure 6 are self-stabilizing. Self-stabilization requires the properties of closure and convergence that can be verified by a variant function (see [22]). Every step of the automata monotonically decreases the value of the variant function until it reaches zero, which implies the end of the stabilization period. In order to define the potential function, we represent the automata as directed graphs.

The automaton as a graph The graph $\Phi = (V, E)$ has the set of states as the set of vertices, V. Moreover, the transitions function, $\tau()$, induces directed edges of E, i.e., $(v, u) \in E \leftrightarrow \exists a \in A : \tau(v, a) = u$.

The variant function

A simple and directed path is one without any directed cycles. We define $\mathsf{LongDistance}(q_j)$ to be the length of the maximal simple and directed path in graph Φ, from state q_j to state $P(\emptyset, Dm^*)$.

Let R be an execution of a system that execute the automata $(atm_i)_{i \in N'}$ as depicted in Figure 6, and $c \in R$ be a configuration, where $N' = N \setminus s$ and $s \in S(D)$ be the subordinate group. We define $dis(c, j)$ to be the value of $\mathsf{LongDistance}(q_j)$ in configuration c. We define the potential function $\Phi(c)$ to be 0 if all automata $(atm_i)_{i \in N'}$ are in the same state in c (i.e., $\exists q \in Q : \forall i \in N' \rightarrow q_i = q$), and $\Phi(c) = \max_{j \in N'} dis(c, j)$ for all other cases. By looking at Figure 5 and Figure 6, we can easily conclude Observation 2.

Observation 2. $0 \leq \Phi(c) \leq \gamma + m^* D(D+1)/2$.

Claim 6 demonstrates that once $\Phi(c) = 0$ holds, the system never leaves the legal execution.

Claim 6 (Closure)**.** *Let c' be the configuration that immediately follows c. Then, $\Phi(c) = 0 \rightarrow \Phi(c') = 0$.*

Proof. By the assumption that $\Phi(c) = 0$, we have that $\exists q \in Q : \forall i \in N' \rightarrow q_i = q$. Recall that we assume that all non-deviating automata $(atm_i)_{i \in N'}$ are deterministic, and therefore $(\mathtt{@s}_i)_{i \in N'}$ are deterministic. Since automata $(atm_i)_{i \in N'}$ observe the same output a, then automata $(atm_i)_{i \in N'}$ move from q to $\tau_i(q, a)$. Hence, $\Phi(c') = 0$. □

Let c be an arbitrary starting configuration of system execution R. Claims 7, 8, 9, and 10 show that within $O(\gamma + D^2 m^*)$ steps, the system reaches a safe configuration, c'', in which $\Phi(c'') = 0$.

Claim 7. *Suppose that $\exists i \in N' : q_i = P(\emptyset, Dm^*)$ in c. Then, $c'' \in R : \Phi(c'') = 0$ within $O(\gamma + Dm^*)$ steps from c.*

Proof. This proof shows that $q_i = P(\emptyset, Dm^*) \wedge \Phi(c) \neq 0$ in c implies that:

Invariant 1. $q_i = P(\emptyset, Dm^*) \wedge \Phi(c) \neq 0$ *in* c'

Invariant 2. $j \in \mathtt{@s}_i$ *(and* $i \in \mathtt{@s}_j$*)*

Invariant 3. $\Phi(c) > \Phi(c')$

Therefore, by Observation 2, we have that the system reaches configuration $c'' \in R$ within $O(\gamma + m^*D)$ steps from c, in which $\Phi(c'') = 0$.

• Verifying invariants 1 and 2

Suppose that $\Phi(c) \neq 0$, which implies (by definition of $\Phi()$) that $\exists j \in N'$, such that $q_i \neq q_j$. Let c' be the configuration that immediately follows c. By definition of f_i, the automaton atm_i's state, $P(\emptyset, Dm^*)$, is distinguishable from atm_j's state. Therefore, by Equation 7, we have invariant (2). This implies that $q_i = P(\emptyset, Dm^*)$ in c', by definition of $\tau_i()$ 2a. Moreover, as long as $\Phi(c') \neq 0$ the invariant (1) holds.

• Verifying Invariant 3

We show that $\forall j \in N' - \{i\}$, we have that $dis(c', j) - dis(c, j) \geq 1 \vee dis(c', j) = 0$, and thus $\Phi(c) > \Phi(c')$. Let us look at the atm_j's state in c:

1. $(Norm^k, d)$. Then $q_j = P(\emptyset, Dm^*)$ in c' (by definition of $\tau_i()$ 1a). I.e. $dis(c, j) \geq 1 > dis(c', j) = 0$.
2. $P(\emptyset, t)$. Then $q_j = P(\emptyset, Dm^*)$ in c' (by definition of $\tau_i()$ 2a and the fact that $i \in \texttt{@s}_j$). I.e. $dis(c, j) \geq 1 > dis(c', j)$.
3. $P(d, t) \wedge d \neq \emptyset$. Then, by definition of $\tau_i()$ 2a and 2b, the value of q_j in c' is:
   (a) $P(\emptyset, Dm^*)$. Then $dis(c', j) = 0$.
   (b) $P(d, t-1)$ (where $t > 1$) or $P(\texttt{@s}_i \cup d, t)$ (where $\texttt{@s}_i \not\subseteq d$). Then $dis(c, j) > dis(c', j)$.
   (c) $(Norm^1, \emptyset)$. Then $dis(c, j) > dis(c', j)$. Moreover, by case 1 within one step from c', case 3a holds.

□

Claim 8. *Suppose that $q_i = P(d_i, t_i)$ in c, where $d_i \in \{\emptyset\} \cup S(D) \wedge t_i \in [1, Dm^*]$. Then, the system reaches configuration $c'' \in R$ within m^*D steps from c, where $\Phi(c'') = 0$ or the assertion of Claim 7 holds in c''.*

Proof. We denote by c the configuration that immediately precedes c'. The proof show that the following invariant. For every pair of configuration c and c', either one of the following hold:
(1) the assertion of Claim 7 holds in c',
(2) $\Phi(c') = 0$, or
(3) $dis(c, j) > dis(c', j)$.
Thus, by Claim 7 and Observation 2, the system reaches configuration c'' within $O(\gamma + m^*D)$ steps from the starting configuration.

Let $q_j : j \in N'$ be atm_j's state in c. We show that for every possible value of q_j in c, we have that either (1) the assertion of Claim 7 holds in c', (2) $\Phi(c') = 0$, or (3) $dis(c, j) > dis(c', j)$.

1. $(Norm^{t_j}, d_j)$, where $d_j \in \{\emptyset\} \cup S(D) \wedge t_j \in [1, \gamma]$. Then by this claim assumption that $q_i = P(d_j, t_j)$, the definition of $f()$ (the assumption that the set $\{P(d, t) | d \in \{\emptyset\} \cup S(D) \wedge 1 \leq t \leq Dm^*\}$ is distinguishable), and definition of $\tau()$ 1a, we have that $q_j = P(\emptyset, Dm^*)$ in c'. I.e. the assertion of Claim 7 holds in c'.
2. $P(d_j, 1)$, where $d_j \in \{\emptyset\} \cup S(D)$. By $\tau_i()$ (2a and 2b) definition, q_j in c' is:
   (a) $P(\emptyset, Dm^*)$. Then the assertion of Claim 7 holds in c.
   (b) $(Norm^1, \emptyset)$. Then $dis(c, j) > dis(c', j)$. Moreover, by case 1 $\nexists k \in N'$, such that $q_k = (Norm^{t_k}, d_k)$ in c', where $d_k \in \{\emptyset\} \cup S(D) \wedge t_k \in [2, \gamma]$. Therefore, in c' either: (1) $\forall k \in N' : q_k = (Norm^1, \emptyset)$ (i.e., $\Phi(c') = 0$), or (2) within one step $q_j = P(\emptyset, Dm^*)$ (i.e., the assertion of Claim 7 holds).
   (c) $P(\texttt{@s}_i \cup d, t)$ (where $\texttt{@s}_i \not\subseteq d$). Then $dis(c, j) > dis(c', j)$.
3. $P(d_j, t_j)$, where $d_j \in \{\emptyset\} \cup S(D) \wedge t_j \in [2, Dm^*]$. By $\tau_i()$ (2a and 2b) definition, q_j in c' is:
   (a) $P(\emptyset, Dm^*)$. Then the assertion of Claim 7 holds in c.
   (b) $P(d_j, t_j - 1)$ (where $t_j > 1$) or $P(\texttt{@s}_i \cup d_j, t_j)$ (where $\texttt{@s}_i \not\subseteq d_j$). Then $dis(c, j) > dis(c', j)$.
4. $P(\emptyset, Dm^*)$. Then the assertion of Claim 7 holds in c.

□

Claim 9 (Strong convergence)**.** *Suppose that $w = \Sigma_{a \in A} \alpha_a u(a)$ is a payoff profile for which $\exists a_1, a_2 \in A : \alpha_{a_1}, \alpha_{a_2} > 0 \wedge a_1 \neq a_2$. Then $\exists c'' \in R : \Phi(c'') = 0$ within $O(\gamma + m^*D)$ steps from execution's R starting configuration c.*

Proof. By Claim 7 and Claim 8, it is sufficient to show that $c' \in R$ within $O(\gamma + m^*D)$ steps from c, such that $\exists i \in N' : q_i = P(d,t)$, where $d \in \{\emptyset\} \cup S(D) \wedge t \in [1, D\gamma m^*]$. By the definition of $\tau_i()$ 1b, if $\exists i \in N : q_i = (Norm^k, d) \wedge d \neq \emptyset$ in c, then the system reaches c' within $O(\gamma)$ steps from c. Therefore, we complete the proof by considering all starting configurations, c, in which all automata

$$\forall i \in N : q_i = (Norm^{k_i}, \emptyset) \wedge k_i \in [1, \gamma]. \quad (8)$$

Suppose, without the loss of generality, that $\forall i \in N \exists k_i \in [1,\gamma] : q_i = (Norm^{k_i}, \emptyset)$ in c , and that $i < j \rightarrow k_i \leq k_j$. We note that $k_1 = k_n$ implies that all automata are in the same state and $\Phi(c'') = 0$. By the definition of $\tau_i()$ 1a and 1b, within at most γ steps from c, there is a configuration c', such that $f_i((Norm^{k_1}, \emptyset)) \neq f_i((Norm^{k_n}, \emptyset))$ (or predicate 8 does not hold). Hence, in c', the states q_1 and q_n are distinguishable and within one step all automata set their state to be $P(\emptyset, Dm^*)$ (by the definition of $\tau_i()$ 1a). Hence, we have shown that the system is self-stabilizing.

Let $c \in R$ be a configuration. We define the potential function $\Psi(c) = \arg\min_{0 \leq x < \gamma} \exists i, j \in N' : f_i((Norm^{k_i+x}, \emptyset)) \neq f_j((Norm^{k_j+x}, \emptyset))$. We note that $\Psi(c) \in [0, \gamma - 1]$ by this claim assumption that $\alpha_{a_1}, \alpha_{a_2} > 0 \wedge a_1 \neq a_2$. The rest of the proof follows by induction on $\Psi(c)$. In every step of the induction, the system takes a step from configuration c to configuration c'. By the definition of $\tau_i()$ 1a and 1b, in c' it holds that the invariant of equation 8 does not hold, or $\Phi(c') - \Phi(c) \geq 1$. □

Suppose that the assertion $\alpha_{a_1}, \alpha_{a_2} > 0 \wedge a_1 \neq a_2$ of Claim 9 does not hold. Then, by the definition of $\tau_i()$, predicate 8 (see Claim 9's proof) holds as long as no automaton $atm_j : j \in N \setminus N'$ deviates. Claim 10 consider this case, and can be demonstrated by the arguments of Claim 9's proof. Hence, the system is pseudo self-stabilizing and may stray from legal behavior at most once (when atm_j deviates).

Claim 10 (Pseudo convergence)**.** *Suppose that* $w = \Sigma_{a \in A} \alpha_a u(a)$ *is a payoff profile for which* $\forall a_1, a_2 \in A : \alpha_{a_1}, \alpha_{a_2} > 0 \rightarrow a_1 = a_2$. *Let* $c \in R$ *be a configuration that immediately follows the first deviation by automaton* $atm_j : j \in N \setminus N'$. *Then* $\exists c'' \in R : \Phi(c'') = 0$ *within* $O(\gamma + m^*D)$ *steps from* c.

Hence the lemma. □

Biographical notes: Shlomi Dolev received his D.Sc. in computer Science in 1992 from the Technion. From 1992 to 1995 he was a research associate at Texas A&M University. In 1995 he joined Ben-Gurion University where he is now a full professor holding the Rita Altura trust chair. He published more than a hundred journal and conference scientific articles, including the book "self-stabilization" published by the MIT Press. Shlomi is an associate editor of the IEEE Transactions on Computers and the AIAA Journal of Aerospace Computing, Information and Communication. His current research interests include distributed computing, security and cryptography.

Elad Michael Schiller received his Ph.D., M.Sc. and B.Sc. in Computer Science in 2004, 2000 and 1998 from the Department of Mathematics and Computer Science at Ben-Gurion University of the Negev. His research excellence has been acknowledged by two highly competitive research fellowships from the Israeli government and the Swedish government. He is now with the Department of Computing Science at Chalmers University of Technology and Gothenburg University. His research interests include distributed computing with special emphasis on self-stabilizing algorithms, wireless communications and game theory.

Paul Spirakis, obtained his PhD from Harvard University, in 1982. He is currently the Director of the RACTI and a Full Professor in the Patras University, Greece. Was acknowledged between the top 50 scientists worldwide in Computer Science with respect to "The best Nurturers in Computer Science Research", published by B. Kumar and Y.N. Srikant, ACM Data Mining, 2005. His research interests Algorithms and Complexity and interaction of Complexity and Game Theory. Paul Spirakis has extensively published in most of the important Computer Science journals and most of the significant refereed conferences. He was elected unanimously as one of the two Vice Presidents of the Council of the EATCS.

Philippas Tsigas is an Associate Professor in the Department of Computer Science and Engineering at Chalmers University of Technology in Gothenburg, Sweden. He is the scientific leader of the Distributed Computing and System research group and his research interests include distributed and parallel computing and systems, as well as general information visualization.