Strategies for Repeated Games with Subsystem Takeovers

Implementable by Deterministic and Self-Stabilizing Automata (Extended Abstract)

Shlomi Dolev [†¶] Elad M. Schiller [‡‖] Paul G. Spirakis [§¶] Philippas Tsigas [‡‖]

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.

Categories and Subject Descriptors

F.2 [**Analysis of Algorithms and Problem Complexity**]: Miscellaneous

General Terms: Reliability, Theory

[†]Department of Computer Science, Ben-Gurion University of the Negev (Israel) dolev@cs.bgu.ac.il.[‡]Department of Computer Science, and Engineering, Chalmers University of Technology (Sweden) {elad,tsigas}@chalmers.se.[§]Research Academic Computer Technology Institute, (Greece) spirakis@cti.gr.

[¶]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*).

Autonomics 2008, September 23 - 25, 2008, Turin, Italy

Keywords: Game Theory, Folk-Theorem, Joint Deviations, Finite-Automata, Self-Stabilization

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.

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 [34]). 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, 38]). 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 coop-

[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.

erative 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 [29]).

COROLLARY 1 (LEMMA 1 OF SECTION 4). *Autonomous and selfish agents can deter joint deviations of the subordinate groups using deterministic and finite automata.*

Complexity issues of games with subsystem takeovers

Computational game theory has several ways of measuring complexity (see [35]). The two most related to games with subordinates' deviations are:

∘ *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 [19, 13].

∘ *Costs of implementing strategies* What is the minimal amount of memory required for implementing a given strategy? Kalai and Stanford [30] 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 5). *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.*

Tolerating transient faults

When designing a distributed system of selfish-computers, it is unsuitable to assume that failures never occur (see [23, 22, 24]). 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.

Self-stabilization

Self-stabilizing systems [20, 21] 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 6). *Self-stabilizing automata can satisfy the assertion of Corollary 1.*

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 [38], 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 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

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

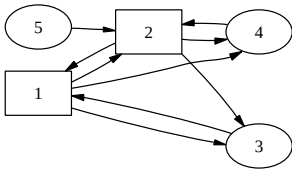
We describe an example of a system with n selfish-computers. We denote by $N = \{1, \ldots, n\}$ the set of agents; each represents a selfish-computer. Every agent decides whether to be a Server or a Client. An agent receives a payoff of -1 for every period in which it decides to be a Server (independently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

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 [25, 16, 3, 32, 17, 4]). [3]

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.

Document structure

We illustrate the problem at hand (Section 2) before we define subsystem takeovers (Section 3). Then, we explain the proofs of Corollary 1 (Section 4), Corollary 2 (Section 5), and Corollary 3 (Section 6). Lastly, we draw our concluding remarks. Throughout we follow the definitions and notations of [36] and for the reader's convenience, the Appendix presents a glossary.

2. 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 [28]).

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.

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*.

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.

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 responsibil-

[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 [25, 16, 3, 32, 17, 4]).

[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.

ity 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.

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.

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 [40, 36]). 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 [36], 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.

3. 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] (Recall that throughout we follow the definitions and notations of [36] and that the Appendix presents a glossary.)

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 $(st_{-s}|_h, st_s|_h)$.

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 $O_h(st_{-s}|_h, st'_s|_h) \prec_{j_{subor}} |_h \; O_h(st_{-s}|_h, st_s|_h)$. We mention that joint deviations in which agent j_{subor} may exist are not considered by [5, 38, 6, 1, 12, 33] (see the discussion in Section 7).

[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	⟨Server, {2, 3}⟩	⟨Client, {1}⟩	⟨Client, {1}⟩
2	⟨Client, {2}⟩	⟨Server, {1, 3}⟩	⟨Client, {2}⟩
3	⟨Client, {3}⟩	⟨Client, {3}⟩	⟨Server, {1, 2}⟩
⋮	⋮	⋮	⋮

(a) A sequence of cooperation for 3 agents.

Period	Agent 1	Agent 2
1	⟨Server, {2}⟩	⟨Client, {1}⟩
2	⟨Client, {2}⟩	⟨Server, {1}⟩
⋮	⋮	⋮

(b) A scheme for punishing agent 3.

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

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 [36]). 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 $i \in s$:

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

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.

4. FOLK THEOREM FOR GAMES WITH 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 [36], 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 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.

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

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 $0 \leq \varrho \leq |s' \setminus s|$. However, a suffix of the punishment sequence is guaranteed to 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 $j \in N$ 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 $s' \in S(D)$:

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

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 [38, 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 5).

5. STRATEGY COMPLEXITY OF GAMES WITH SUBSYSTEM TAKEOVERS

Computational game theory has several ways of measuring complexity of games (see [35]). In Section 1, we mentioned the computational complexity of games with subsys-

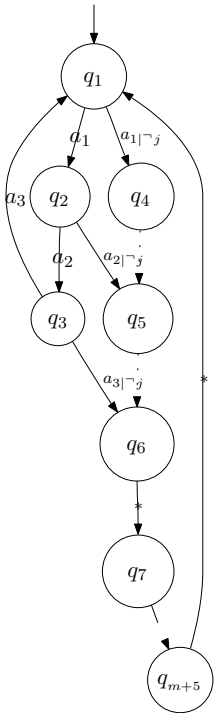
On the right side of this figure, we sketch an automaton for agent $i \in \{1,2,3\}$. For brevity, the sketch merely considers a specific deviating agent $j \in \{1,2,3\}$, such that $i \neq j$. †

• $Norm$ **(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.

tem takeovers. We now turn to consider the strategy complexity of these games. Kalai and Stanford [30] 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 [30] show that the complexity of a strategy equals the size of the smallest automaton that can implement the strategy.

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.* [8]

Proof outline A strategy that deters subordinates' deviations is presented in Section 4. The automaton that implements these strategies requires $O(D\binom{n}{D})$ states. 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 [30] the strategy complexity is in $\Theta(D\binom{n}{D})$. □

[8]The lower bound holds when considering t-strong [5] or t-resilient [1] equilibria, because the t-defendable property implies the t-strong and t-resilient properties (see Section 7).

6. SELF(ISH)-STABILIZATION

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

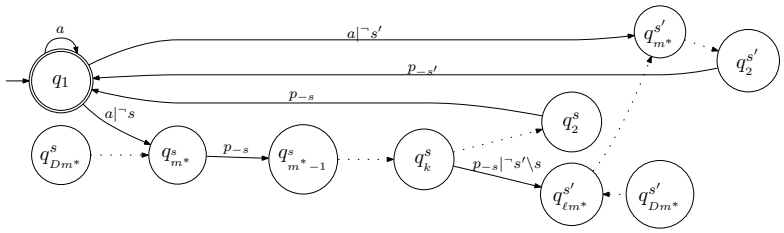
LEMMA 3 (COROLLARY 3). *Let* Γ *be an infinitely repeated game of* G *under the limit of means criterion. Then, there are self(ish)-stabilizing automata that implement subgame perfect equilibria that are* D*-defendable in* Γ.

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. [9]

Self-stabilization requires the properties of closure and convergence that can be verified by a variant function (see [21]). Every step of the automata monotonically decreases the value of the variant function until it reaches zero,

[9]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^*.

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.

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* 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 [30] 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).

7. RELATED WORK

The solution concepts of strong Nash equilibrium [5, 38, 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, 33]). 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 [23, 22, 18, 24, 15] was earlier considered for single stage games. The game authority [23, 22, 24] verifies that no agent violates the game model of the stage game. Spanning trees among selfish parties are studied by [18]. Reactive systems that are inspired by game theory appear in [15].

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 [32], and Synchronous Terminating Reliable Broadcast [16]. 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.

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, 39]) 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 [36], 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, 37]). In fact, there is evidence that this class is strictly weaker than the class of strategies that Turing machines can implement (see the survey [27] 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 [25] and later [16, 3, 32, 17, 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, 38, 6, 1, 12, 33], 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 [34] is the conventional Nash equilibrium, and n-$\mathcal{X}$ Nash equilibrium is the conventional strong Nash equilibrium [6, 38].

• **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 [23, 22, 24] 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.

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 elegant way for designing decentralized autonomous systems.

8. 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] 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.

[16] 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.

[17] A. Clement, J. Napper, H. C. Li, J.-P. Martin, L. Alvisi, and M. Dahlin. Theory of BAR games. In Gupta and Wattenhofer [26], pages 358–359.

[18] 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.

[19] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a nash equilibrium. In Kleinberg [31], pages 71–78.

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

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

[22] 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/`.

[23] 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.

[24] S. Dolev, E. M. Schiller, P. G. Spirakis, and P. Tsigas. Game authority for robust andscalable distributed selfish-computer systems. In Gupta and Wattenhofer [26], pages 356–357.

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

[26] 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.

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

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

[29] A. Hayrapetyan, É. Tardos, and T. Wexler. The effect of collusion in congestion games. In Kleinberg [31], pages 89–98.

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

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

[32] 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.

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

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

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

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

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

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

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

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

Appendix – Glossary

For the reader's convenience, we present a glossary. 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.

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 [36]), 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 $i \in N$.

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.

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 $(u_i(b^t))$.

The limit of means criterion The limit of the means criterion [9, 39] 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 $\lim_{T\to\infty} \Sigma_{t=1}^T (v_i^t - w_i^t)/T > 0$. $\lim_{T\to\infty} \Sigma_{t=1}^T (v_i^t - w_i^t)/T > 0$.

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 $\Gamma = \langle N, H, \succsim = (\succsim_i)\rangle$.

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.

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 $\Gamma = \langle N, H, \succsim = (\succsim_i)\rangle$.