EFFECTS OF LYING IN REPUTATION-BASED MULTI-AGENT SYSTEMS

Marek Laskowski
Dept. ECE
University of Manitoba,
Winnipeg, Manitoba, R3T 5V6
email: mlaskows@ee.umanitoba.ca

Sara McGrath
Dept. Computer Science
University of Manitoba,
Winnipeg, Manitoba, R3T 2N2
email: ummcgrat@cs.umanitoba.ca

Abstract

As an increasing number of tasks on the Internet become automated using autonomous agents, it will become increasingly important for these agents to be able to discern which agents can be trusted and which cannot. This is especially true where interacting agents may have divergent goals, an example being Peer to Peer applications. Passing on reputation information about other agents is a strong way to encourage cooperation. This paper points out weaknesses in both a general reputation scheme as well as a framework which was previously proposed. These weaknesses could leave the door open for exploitation by malicious agents.

Keywords: *Multi Agent, Internet, Evolutionary Computing*

1. Introduction

Peer to Peer (P2P) systems inherently rely on the cooperation of members for stability. Studies have shown [1] that only a small percentage of P2P members contribute (cooperate) usefully to network services, while the majority freeload or are even malicious agents (defection). The next generation of P2P applications such as distributed caching of the World Wide Web [2] may require even more rigorous cooperation in the sense that a member may be required to cache and offer for download data which may not be of any direct interest to themselves. Clearly, self-interest alone may not be enough incentive to keep enough cooperating agents given the propensity for freeloaders or malicious agents to infiltrate the network.

In the past, centralized as well as distributed approaches have been proposed to encourage cooperation among members [3][4][5]. In the centralized case, relying on a few important nodes leaves the network vulnerable to attack, because malicious agents need only take down a few key nodes to cripple the network. A centralized approach may also result in network bottlenecks. On the other hand, distributed models can be compromised (primarily by freeloaders) by members lying about their level of cooperation or participation. The aim of this paper is to investigate potential weakpoints in a hypothetical P2P network using a reputation scheme to promote cooperation. We will use a multi-agent approach in describing and simulating the proposed network.

2. Model

We model our next generation P2P network as a collection of agents. An agent interacting with another agent has the choice of cooperating with the other agent (presumably by serving whatever data request the other has) or by defecting (artificially throttling one's bandwidth, offering corrupt data, or any other means of deception). The same options are available to the other agent as well and the results of their respective choices are contained in the following payoff table.

Table 1: The Prisoner's Dilemma

Payoffs	Cooperate	Defect
Cooperate	3,3	0,5
Defect	5,0	1,1

Table 1 is the classical Prisoner's Dilemma seen in many previous studies concerning cooperation [6]. The Prisoner's Dilemma can be used to model complex interactions because the payoffs do not correspond to real life quantities, only the ordering of preferences is important here (higher is better). This captures the idea that if both agents cooperate they receive a moderate benefit, but not as much as one does at the expense of another if the first defects and the latter cooperates. When both defect they both receive almost no benefit but at the same time neither is totally exploited. Note that the average payoff for both cooperating is higher than in any other case. It is presumed that an agent can differentiate between the cooperation and defection of another agent.

Due to the nature of our application, pieces of a given file (or web page, or whatever it is the network is intended to share) may be distributed over any number of agents. As a result, an agent may potentially be called upon to interact with any number of random agents in the system to complete a single higher level data request. Agents have the ability to remember agents they have previously dealt with, and thus will be able to recognize agents that have cooperated or cheated them previously. Agents will then be able to make future decisions on whether to cooperate with a given agent accordingly. As with Armstrong and Durfee [7], the behavior of agents is governed by a set of parameters. For example parameter “i” specifies the base probability for an agent cooperating with an agent it has never encountered. We do not, however, consider any spatial aspect as they do. In our system in addition to exchanging commodity data agents can also share information about the trustworthiness or reputation of other agents. Agents can use this reputation information in conjunction with firsthand information (history) to form a primitive world model when dealing with other agents. In addition to using the stored history, the parameter “u” dictates how much weight is given to stored reputation information when calculating the probability of cooperating with a particular agent. Thus, an agent can learn about the reputation of another agent before having to deal with it for the first time. In our system agents will exchange reputation information immediately after exchanging data during a given interaction.

3. System Evolution

Our initial investigation uses a custom simulator written in C++. We submit that the evolutionary stability and robustness of the following strategies will demonstrate the stability of a P2P network using those strategies.

During a simulation run, payoffs from each agent's interactions with the rest of the population during a time period T (measured in number of interactions) are tallied. For our experiments T was 10,000 games or interactions. Each set of such games is called a generation. At the end of T the top scoring 10% of the population are probabilistically chosen to replicate themselves and replace the lowest scoring 10% of the population, also chosen probabilistically. In addition, small changes to agent’s parameters will be made at random to a relatively small percentage of the population.

Although a Genetic Algorithm approach was considered, the chosen approach is less intensive computationally. It also does not have the drawback, as some have argued [8], of crossover being basically a macro-mutation operator. However, it does rob the system of an ability to innovate. That is why we also include a mutation-like operator. After each generation it chooses 10% of the population at random. For each of these, a parameter is randomly chosen to be modified by an absolute amount of +/- 0.05. Care is taken to ensure that a parameter does not take an invalid value as a result.

Agents which have their parameters replaced by stronger agents are “forgotten” by all other agents in the system. Similarly, their history and reputation tables are wiped clean. Effectively they are new to the system. After each generation, 5% of the population are chosen at random and “forgotten” by all other agents. Both these processes allow agents to over time redeem themselves if they have poor reputations. This also captures the effects of agents entering and leaving our simulated P2P network, or perhaps agents having to interact with new agents because of a particular rare commodity they are seeking.

We will only simulate 100 agents to represent the agents that any one agent will typically interact with (for example the 99 closest agents geographically). Also, reputation table entries will probabilistically be deleted over time to reflect the effects of limited cache space, new agents entering the system, as well as having to interact with an existing agent for the first time because they have a rare piece of needed data. This also allows agents with changing parameters to shed over time a reputation which perhaps no longer applies

In a real implementation however, some kind of request mechanism would be in place to eliminate unnecessary transfer of reputation information, especially if there are potentially a large number of agents.

4. Strategies

Many strategies from other cooperation studies [6][7], can be represented using our notation. The agent types used are as follows, including the equivalent type (if applicable) in Sen et al.[5].

- Tit for Tat or TFT for short (approximates Sen's Reciprocative strategy): This strategy starts by cooperating and then reciprocates whatever its opponent did the last time they interacted. It does not care about the reputation of others when deciding whether or not to cooperate. Because it's a “nice” cooperative agent it always tells the truth about other's reputations.
- Tit for Tat with Reputation (approximates Sen's Earned-Trust Based Reciprocative): TFT w/ Rep for short. Works like TFT but also accepts reputation information from agents which have cooperated with it in the past. This allows it to theoretically predict whether an agent it has never encountered is likely to cooperate or not.
- Always Defect or ALLD for short (approximates Sen's Collaborative Lying Selfish): Not only does this strategy never cooperate, it also tries to make selfish agents like itself look cooperative and make cooperative agents look selfish by spreading false reputation information.
- Lying Reciprocative or TFT Lying: Is cooperative like TFT, but ignores reputation information. This strategy can be described as “two faced” as it lies about all other agents making them look selfish.
- RANDOM: Cooperates exactly 50% of the time. Since this is not intended to be a malicious strategy, it always tells the truth about other agents.

5. Experiments & Results

We ran a number of simulations to investigate the evolutionary stability of the mentioned strategies. First we compared the time it takes for TFT without reputation to dominate a population of ALLD agents versus the time it takes for TFT with reputation to dominate a population of ALLD agents.

At the beginning of each trial we begin with 16% TFT agents and 84% ALLD. Axelrod [6] suggested that as few as 5% TFT agents are required for cooperation to eventually dominate, but in this system we found it to be about 15%.This is likely due to agents being forgotten at the end of each generation, which gives ALLD a slight advantage since all cooperative agents will be “nice” and attempt to cooperate with it at least once. Figure 1 demonstrates the advantage of using a reputation based system; TFT w/ Rep takes about five generations fewer to dominate ALLD. Standard deviation is shown with the error bars.

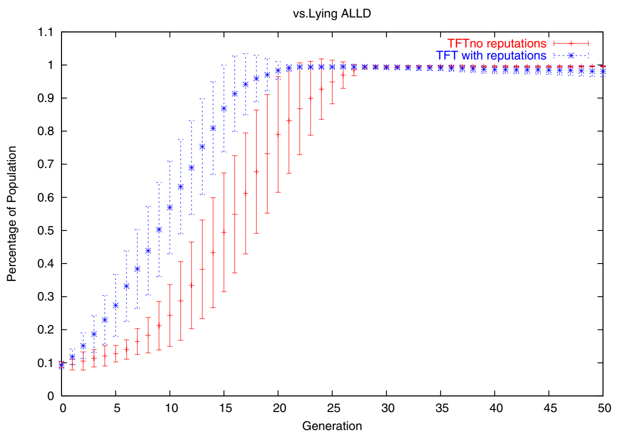

Fig. 1. Number of generations before TFT variants dominate the population, initially 16% TFT & 84% ALLD

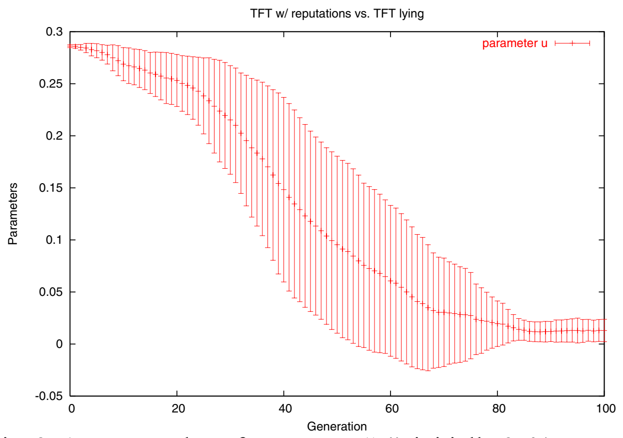

Fig. 2. Average Value of Parameter “u”, initially 95% TFT w/ Rep (u = 0.3) versus 5% TFT Lying (u = 0)

The advantage demonstrated in Figure 1 does come at a price. Figure 2 shows that when TFT w/ Rep is paired against TFT Lying, the latter invades and dominates a population of the former within 80 generations. TFT Lying accomplishes this so quickly by turning the reputation trait against itself. By always giving low reputation information TFT Lying over time subverts TFT w/ Rep into behaving effectively like ALLD. This allows TFT Lying to eliminate TFT w/ Rep, just as ALLD was eliminated in Figure 1.

Using reputations has at least one more weak point which we will illustrate. That is, agents that employ different strategies may have a different frame of reference or world view resulting in incompatible reputation information. Figure 3 shows the results of running basic TFT in a population of RANDOM agents compared to TFT w/ Rep in a population of RANDOM agents.

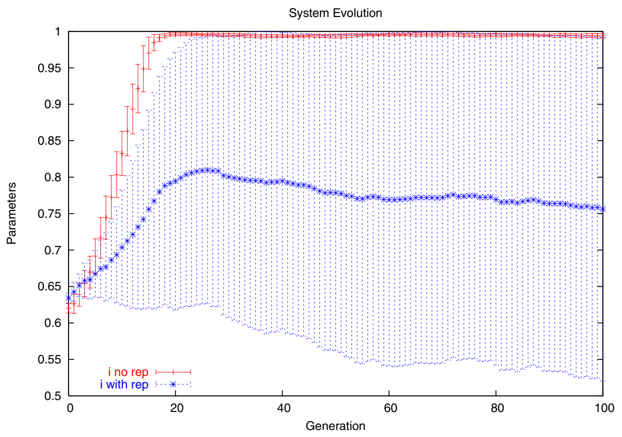

Fig. 3. TFT vs. RANDOM and TFT w/ Rep vs. RANDOM, initially 30% TFT (i = 1) and 70% RANDOM (i = 0.5)

Interestingly enough, TFT w/ Rep does considerably worse on average, and sometimes fails to dominate a RANDOM population after 100 generations. This is all the more intriguing considering that RANDOM is by default passing on the true reputation. To explain, consider what would happen over a number of interactions between RANDOM and TFT w/ Rep agents, ignoring for the moment the possibility of erroneous reputation information. Over time, a TFT w/ Rep agent should be able to correctly model a RANDOM agent resulting in an approximately 50% chance of cooperating with the RANDOM agent. At the same time, the RANDOM agent is also modeling the TFT w/ Rep agent, not that it uses that model in deciding whether to cooperate or not. The model the RANDOM agent forms of the TFT w/ Rep agent is basically a RANDOM agent since the TFT w/ Rep agent will cooperate approximately 50% of the time, as mentioned above. Now, when this RANDOM agent cooperates with another TFT w/ Rep agent, it passes on that the first TFT w/ Rep agent is a RANDOM agent. Over time the TFT w/ Rep agents treat each other like RANDOM agents because of the reputation information passed on by RANDOM agents. In this way the TFT w/ Rep agents lose any advantage they had over RANDOM, all without any agents lying. This is what we mean when we say that RANDOM has a different frame of reference. It views TFT w/ Rep agents differently, because it is treated differently by TFT w/ Rep, because at the core they behave differently than TFT w/ Rep agents.

6. Other Reputation Frameworks

To demonstrate that the above weakness is not limited to the system we presented, we also implemented Sen's information processing domain [5] using Swarm [9], and used Sen's equivalent of TFT w/ Rep, ALLD, and TFT Lying. Instead of playing the Prisoner's Dilemma, agents are experts in one of three task types. At each time step an agent is assigned one of the three task types at random. The quality of the completed task will be greater if an agent is an expert in that particular task type. An agent may ask another agent to do the task, provided that the first thinks that the second can do a better job (for example if the second is an expert in that task and the first is not). An agent which helps another incurs no cost besides improving a competing agent's score with respect to its own.

If agent A helps B with a task then A is said to have a positive balance with B (B owes A help), and B has a negative balance with A until B helps A with some task. Note that A's balance with B and B's balance with A are not necessarily equal. In this domain TFT type agents reciprocate probabilistically, with the probability of helping an agent increases as the balance with that agent increases. When agents that use reputation are deciding whether to help another agent, not only do they look at their own balance with that agent, but they also consider the balances (with the agent requesting help) of agents they owe help to. So if A owes B, and A is deciding whether to help C, A will consider B's balance with C, in addition to A's own balance with C. An important difference here is that TFT Lying agents report a huge negative balance when asked for balances with other agents regardless of actual balance. TFT Lying will only look at its own balances when deciding whether or not to help another agent. ALLD will request help but will never help others. As before ALLD distorts balances by reporting that agents with negative balances have positive balances and likewise that agents with positive balances have negative balances. For further details please see [5], note that the naming convention for the agents will be different, as explained earlier.

A performance comparison between ALLD and TFT w/ Rep (lower curves) when either interacts with TFT Lying (upper curves) is shown in Figure 4, below. What it suggests is that TFT Lying causes TFT w/ Rep to behave like ALLD. TFT Lying accomplishes this by providing balances which fool TFT w/ Rep into believing all other agents are ALLD.

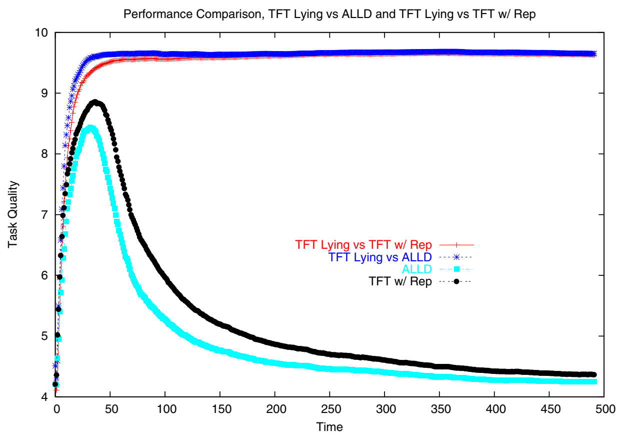

Fig. 4. Performance comparison for the information domain

To see exactly how this is occurs, consider Figure 5. Agents A & B are Reciprocative (TFT), however they lie and report a huge negative balance with any given agent when queried about their balances. C, D, and E being TFT w/ Rep will consider the balances of agents who have helped them in the past. So, when agent A helps C (in blue), C will query A when deciding to help anyone else. A will report that any other agent owes it a huge amount, effectively infinity, causing C to behave as an ALLD agent towards all agents besides A. At some point B will also help out C (in red) and convince C to not help A either. Over time C will have a negative balance with all other agents (since it only accepts help and never offers) large enough so that all other agents will no longer help C out. Effectively C is cut off from the rest of the agents in the system.

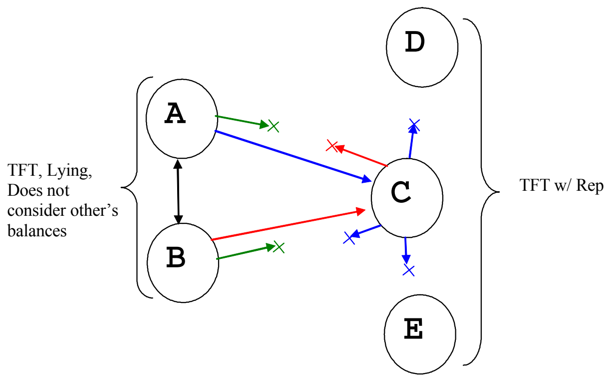

Fig. 5. Information domain example - how a TFT w/ Rep agent is made to behave as ALLD

Over time, A and B will convert all other TFT w/ Rep agents in the system to behave as ALLD agents in the same manner. A and B will continue to behave Reciprocatively towards one another since they do not consider any balances besides their own. Since Sen demonstrated that Reciprocative or TFT agents will eventually dominate ALLD agents, so will the Reciprocative agents dominate the TFT w/ Rep agents over time. Thus, we have shown how TFT w/ Rep agents are also not evolutionarily stable in Sen's system.

7. Conclusions

We have demonstrated that because one can trust an agent to cooperate, one should not trust the agents it trusts nor out of hand mistrust the agents it mistrusts. Not only is lying a possibility, but a difference in perspective could cause incompatable reputation information. The good news is that a simple Tit-for-Tat strategy is quite robust, especially when future interactions with the same agent are likely. The bad news is that in cases where the temptation to both defect and lie is large, and the relative probability of future interactions is low, a method to detect defectors before interacting is desirable. Unfortunately, the reputation schemes we presented may not be enough. Suggestions for detecting lying exist [10], however these do not into take account that other agents may have different yet legitimate frames of reference, so a grounding [11] approach may be necessary as well.

8. Acknowledgements

We would like to thank the Internet Innovation Centre at the University of Manitoba for the use of their facilities as well as assistance in attending this conference.

9. References

[1] S. Saroiu, P. Gumma, and S. Gribble. "A Measurement Study of Peer-to-Peer File Sharing Systems," *In Proceedings of Multimedia Computing and Networking*, 2002.

[2] S. Iyer, A. Rowston, and P. Druschel. "Squirrel: A decentralized peer-to-peer web cache," *In 21th ACM Symposium on Principles of Distributed Computing*, pp. 213-222, 2002.

[3] P. Resnick and R. Zeckhauser. "Trust among strangers in internet transactions: Empirical analysis of ebay's reputation system," Working Paper for the *NBER workshop on empirical studies of electronic commerce*. 2002.

[4] Bram Cohen. "Incentives build robustness in bittorrent," Proceedings of the *First Workshop on the Economics of Peer-to-Peer Systems*, June 2003.

[5] S. Sen and P. Dutta,``The evolution and stability of cooperative traits," in the Proceedings of the *First Intenational Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1114-1120, July 2002.

[6] R. Axelrod. *THE EVOLUTION OF COOPERATION*. Basic Books, 1984.

[7] A.A. Armstrong, E.H. Durfee. "Mixing and memory: Emergent cooperation in an information marketplace," *Third International Conference on Multi Agent Systems*. 1998.

[8] P. Angeline. "Subtree Crossover: Building Block Engine or macromutation?," *Proceedings of Second Annual Conference in Genetic Programming*, pp. 34-41, 1997.

[9] http://www.swarm.org

[10] S. Buchegger and J. LeBoudec. "A Robust Reputation System for P2P and Mobile Ad-hoc Networks," Proceedings of *P2PEcon*, 2004.

[11] D. Jung and A. Zelinsky. "Grounded Symbolic Communication Between Heterogenious Cooperating Robots," *Autonomous Robots*, pp. 269-292, 2000.