Emergent Social Rationality in a Peer-to-Peer System

Andrea Marcozzi and David Hales

Department of Computer Science, University of Bologna, Mura Anteo Zamboni 7, 40127 Bologna, Italy {marcozzi, hales}@cs.unibo.it

1 Introduction

For many applications peer-to-peer (P2P) systems require their member nodes (or agents) to behave in a socially beneficial (non-egotistical) way. Kalenka and Jennings [4] termed this requirement as the Principle of Social Rationality: if an agent has a choice of actions it should chose the action that maximizes the social utility (sum of all agent utilities in the system). This principle can be contrasted with classical individual rationality that states agents should select actions that maximize their individual utility. However, developing protocols for realistic P2P systems that adhere to the principle of social rationality is very difficult and potentially so costly as to negate the benefits. This is because P2P systems have no central control, are potentially huge (composed of millions of nodes) and have high node turnover (with users continually entering and leaving the system). In addition, selfish or malicious nodes can get into the system via hacked client programs. These factors mean that individual nodes, even if they wish to follow a socially rational principle, often will not have enough information to gauge the effects of their actions on others. Recently, simple locally adaptive protocols have been proposed that claim to produce socially rational outcomes through a process of self-organisation even though nodes only act on their own utility values. In this approach nodes preferentially copy other nodes (by duplicating their behaviour and links) that have higher utilities. However, in these previous works only specific scenarios are considered in which certain plausible utility values are selected. In this paper we introduce a variant (*ResourceWorld*) of one such existing P2P scenario [1] (*SkillWorld*). For both models we explored a large space of different cost / benefit values to check if the protocols maximized the collective utility or not. In ResourceWorld we found that if the collective cost of an action was less than or equal to the collective benefit the protocol self-organized the network to a state where nodes selected this action. For SkillWorld we found a less socially rational rule.

2 The ResourceWorld Scenario

The *ResourceWorld* model, which takes inspiration from a previous model [1], represents the situation in which nodes in a P2P network can store and serve a single resource from a set (R). Each node may have a maximum of 20 links to other nodes (peers). Each link is bidirectional: if node a is connected to node b, even node b is connected to node a. Links are undirected so the entire network can be considered as an undirected graph where each vertex is a node and each edge is a link.

The state variables of each node is shown in table 1.

Table 1. Nodes state

Parameter	**Value**
Altruism flag	$A \in \{0, 1\}$
Resource/Skill type	$R \in \{1, 2, 3, 4, 5\}$
View	$d = 20$ links to other nodes
Utility	$U \in \mathbb{R}$

The Resource, which indicates the ability held by a single node, is the only parameter which does not evolve: it is not copied during the reproduction phase (see section 4), but it may "mutate" (that is change) with a very small probability. Periodically (at each iteration cycle, where a cycle is all nodes firing in a random order) with probability 0.5, nodes receive a *request* or job (J) to be completed. The request is produced by selecting at random a value from a set of 5 elements ($J \in \{1, 2, 3, 4, 5\}$) and the receiving nodes, in order to complete the request, must hold the appropriate resource. Suppose node i having a certain R_i and a certain strategy A_i receives a request J. Three situations can take place:

- $J = R_i$: in this case (cf. figure 1a) i can satisfy the request by itself (doesn't matter if A is set to 0 or 1) and node i increases its utility by a benefit payoff b ($U_i := U_i + b$) ;
- $J \neq R_i \wedge A_i = 1$: in this case (cf. figure 1b) i itself can not satisfy the request but since it is a cooperating node ($A_i = 1$), it can pass the request to its neighbors (given in the node view) in turn. If one of them (say j) has the appropriate resource ($R_j = J$), then the request can be satisfied; in this case node j gets the benefit payoff ($U_j := U_j + b$) while node i pays a cost payoff ($U_i := U_i - c$); if none of neighbors has the requested resource, the request J can not be completed and no payoffs are given;
- $J \neq R_i \wedge A_i = 0$: in this case (cf. figure 1c) since node i is a selfish node, it cannot pass the job to its neighbors, even though they have the needed resource.

This means that each node receiving a request, if not able to satisfy it with its own resource, it is willing to pay a cost in order to have the request completed.

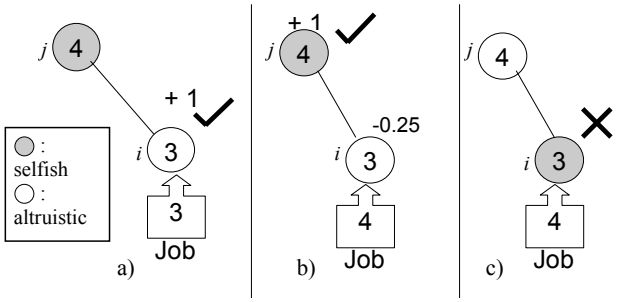

Fig. 1. An illustration of the "ResourceWorld" model. Shading of nodes represents strategy; the number inside the node indicates the *resource / skill*.

This minimal scenario represents a situation in which user-level requests of *resources* (disk space, files, programs, etc..) are supplied to a P2P network composed of nodes holding just one of such needed resources. Nodes should be able to self-organize in a way that makes the most of the created request to be satisfied in the shortest possible amount of time. This organization task is performed by the SLAC protocol [2] which is described in section 4.

3 The SkillWorld Scenario

The *SkillWorld* (SW) model [1] is very similar to the ResourceWorld (RW) model we have just described but it represents a different scenario. Here, what we have previously called resource, is called skill (resource and skill are the same thing, we use different names just to distinguish the two models). This skill represent the ability of a node to work on a certain Job (J). The state variables of the nodes is the same reported in table 1. The difference with RW, is in the way by which the nodes interact.

In SW when a node hasn't the right skill to execute a job (J), it looks for an *altruistic* neighbor to exploit, in order to complete the job and receive a benefit. Once selected, the altruistic neighbor will have to pay a cost. We can see this as a "favour" that the selected node does to its friend, but for this it pays a cost in terms of personal resources (disk space, cpu, ecc..).

4 SLAC in ResourceWorld and SkillWorld

The SLAC protocol (Selfish Link-based Adaptation for Cooperation) has the ability to produce high levels of cooperation in P2P networks while performing some tasks. It has already been covered in details elsewhere [2] so here we give just a brief overview.

The SLAC algorithm specifies how nodes should update their strategies and links. At each cycle of the simulation task, with a certain probability, each node selects a random node from the entire network and their utilities are compared. Suppose node i has an average utility greater than j ($U_i > U_j$): in this case j copies node i strategy (not the resource); j drops all its links and moves to i's neighborhood (copies all i's links and adds a link to i itself). After this, "mutation" is applied with different probabilities, separately, to j's links, to j's strategy and to j's resource / skill. This involves changing the links, strategy and resource / skill. The strategy is flipped, the resource is replaced by uniformly randomly selected new resource value and links are wiped and replaced with a single link to a randomly selected node from the population. Mutation of the resource is for introducing some noise in the system.

5 Experimental Configuration

We performed a massive number of experiments with both models modifying the utilities with the aim to explore a large space of possible values (playing with benefit and cost values). Simulations were carried on *Peersim* [5], an open source P2P systems simulator platform, using the *Newscast* protocol [3] for the management of the overlay topology. The time is divided in cycles and in each cycle each node performs the specific actions described in the previous sections. In each experiment we checked if SLAC maximizes the collective utility or not.

The configuration we adopted in the experiments is shown in table 2.

Table 2. Experiments configuration

Parameter	**Value**
Network size (N)	4000
Network degree (d)	20
Initial topology	*random*
Nodes' Strategy Initialization	$A_i = 0$ (*selfish*)
Resource/Skill initialization	$R_i \in \{1, 2, 3, 4, 5\}$
Links Mutation MRT	0.01
Strategy (MS) and Resource Mutation (MR)	0.0025
Benefit (b) and Cost (c) payoff	$\{0.1 \ldots 2.0\}$ steps by 0.1

For each configuration we performed 10 different runs and we took the average of the results: this means that we performed $20 \times 20 \times 10 = 4000$ different runs.

6 Experiments Results

The performance measure we adopted is the percentage of completed jobs (Pcj). All the results that we show are from an average of ten run per config-

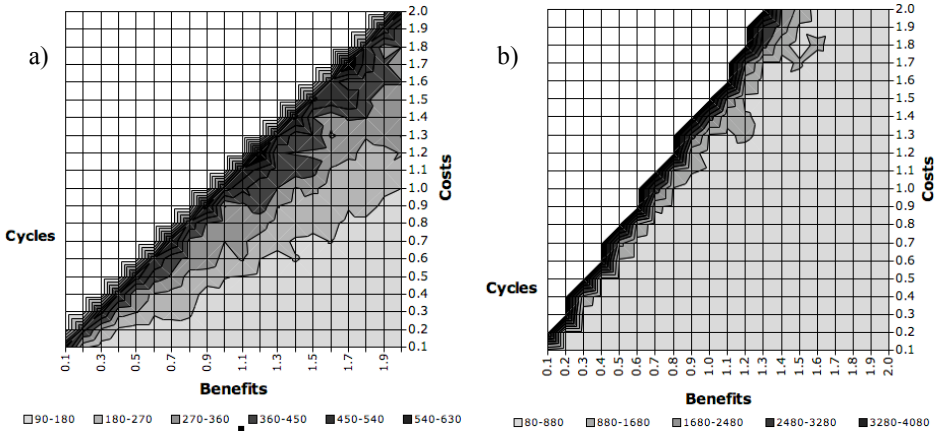

Fig. 2. Map diagram indicating the number of cycles needed to obtain a good level of Pcj (> 80%): (a) relates to the RW model; (b) relates to the SW model. The used parameters for both models are those indicated in table 2. In both diagrams, the top left area indicates that no good results are obtained.

uration. We found that in **ResourceWorld** to obtain a good Pcj (by good Pcj we mean a value greater than 80%), the benefit payoff must be greater or equal than the cost payoff ($b \geq c$). When $b < c$ we obtain a very low Pcj (ranging from 25% to 35%); when $b = c$, a good level of Pcj can be obtained, but the system will take longer to achieve this.

From figure 2a we note that in order to obtain a good level of Pcj in a small amount of time (say less than 180 cycles), the cost payoff must be smaller than the half of the benefit payoff ($c < \frac{1}{2}b$). The bigger is the difference between b and c (of course with $b > c$), the sooner 80% Pcj is reached.

We compared these results from RW with similar experiments using the **SkillWorld** scenario. What we found here is that to obtain a level of Pcj greater than 80%, the cost value can be greater then the benefit. Good levels of Pcj are obtained according to this rule: $c < \frac{3}{2}b$ (see figure 2b). We also found that in order to obtain a good level of Pcj with a small number of cycles, the cost payoff should be smaller than half of the benefit payoff ($c < \frac{1}{2}b$). This also happens in RW.

7 Discussion and Future Works

The SLAC algorithm implements nodes that myopically and greedily attempt to maximize their own utility by copying other random nodes if they perform better. In our simulations even though nodes have no knowledge of the underlying game or of the state of the other nodes, a high number of completed jobs is produced in both models. This requires the self-organisation of both altruism and internal specialized structure.

In this work we took the ResourceWorld (RW) and SkillWorld (SW) models, we varied the cost / benefit values and we found two interesting rules (see table 3).

Table 3. Rules leading to high Pcj in RW and SW.

Model	**Rule**
ResourceWorld	$\frac{b}{c} \geq 1$
SkillWorld	$\frac{b}{c} > \frac{2}{3}$

In RW a high number of jobs can be completed when the cost payoff is smaller than or equal to the benefit payoff. In SW, even when the cost payoff is greater than the benefit (for certain values only), the whole benefit is sustained. We currently don't know why this happens. We think that these results may be influenced by the topology of the network or by the number of skills with which we are playing, so as future work it would be interesting to make tests with a smaller network degree or with different skills and jobs number. More experiments investigating the topology evolution could also illuminate this relationship. We have seen from previous works [1] that with SLAC, a network starting from any initial topology quickly self-organize into a set of disconnected highly clustered components (*tribes*). We think that the same happens here, through some kind of "group-selection" between these tribes leading to the selection of more socially optimal behavior.

It is important to note that although selection appears to operate at the group level it is the result of selection (it is an emergent property) operating on the individual node level. What is significant here is that for the RW model this process is sufficient to select a socially rational behavior in the nodes, maximizing collective or social benefit over individual benefit. The SW results however show that such behavior is not always selected and future work will hopefully allow us to give a general theory which can predict what kinds of scenario will and wont produce social benefit following our simple approach.

References

1. D. Hales. Emergent Group-Level Selection in a Peer-to-Peer Network. *Complexus* 2006; 3:108–118, 2006.
2. D. Hales and S. Arteconi. SLACER: A self-organizing protocol for coordination in peer-to- peer networks. *IEEE Intelligent Systems*, 21(2):29-35, Mar/Apr 2006.
3. M. Jelasity, M. van Steen. Large-Scale Newscast Computing on the Internet. *Internal Report IR-503, Vrije Universiteit Amsterdam* The Netherlands, 2002.
4. S. Kalenka and N. R. Jennings. Socially Responsible Decision Making by Autonomous Agents. Cognition, Agency and Rationality. *(eds. Korta, K., Sosa, E., Arrazola, X.)* Kluwer 135-149, 1999.
5. *http://peersim.sourceforge.net*