Emergent Social Rationality in a Peer-to-Peer System.

Marcozzi, A. and Hales, D.

2006
Emergent Social Rationality in a Peer-to-Peer System

Andrea Marcozzi       David Hales

Technical Report UBLCS-2006-23
October 2006

Department of Computer Science
University of Bologna
Mura Anteo Zamboni 7
40127 Bologna (Italy)
Recent Titles from the UBLCS Technical Report Series

2006-10 Fast and Fair Event Delivery in Large Scale Online Games over Heterogeneous Networks (PhD Thesis), Palazzi, C.E., April 2006.
2006-14 Performative Patterns for Designing Verifiable ACLs, Dragoni, N., Gaspari, M., April 2006.
2006-17 A Secure Peer Sampling Service as a Hub attack countermeasure, Jesi, G. P., Gavidio, D., Gamage, C., van Steen, M.
2006-18 Modified Realizability and Inductive Types, Asperti, A.; Tassi, E., June 2006
Emergent Social Rationality in a Peer-to-Peer System

Andrea Marcozzi

David Hales

Technical Report UBLCS-2006-23

October 2006

Abstract

Many peer-to-peer (P2P) applications require that nodes behave altruistically in order to perform tasks collectively. Here we examine a class of simple protocols that aim to self-organise P2P networks into clusters of altruistic nodes that help each other to complete jobs requiring diverse skills. We introduce a variant (called ResourceWorld) of an existing model (called SkillWorld) and compare results obtained in extensive (ten billion interactions) simulation experiments. It was found that for both model variants altruistic behavior was selected when certain cost/benefit constraints were met. Specifically, ResourceWorld selects for altruism only when the collective benefit of an action is at least as high as the individual cost. Interestingly, this gives a minimal method for realizing so-called “social rationality” where nodes select behaviors for the good of the collective even though actions are based on individual greedy utility maximization.

1. This work is partially supported by the EU within the 6th Framework Programme under contract 001907 (DELIS).

2. Department of Computer Science, University of Bologna, Mura Anteo Zamboni 7, 40126 Bologna, Italy.
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 [7] 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. It would be too costly for every node to ask every other node to report its state before every individual action, even if it was possible. 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 [3] (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 [4], 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: a strategy bit (or altruism flag, \( A \)) which indicates if it is an altruistic \( (A = 1) \) or a selfish \( (A = 0) \) node; a resource type \( (R) \) which is selected within a set of 5 elements \( R \in \{ 1, 2, 3, 4, 5 \} \) indicating the ability held by the single node; a utility \( (U) \), generated by each node according to some interaction with its neighbors; and a list of neighbors (local view).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Altruism flag</td>
<td>( A \in {0, 1} )</td>
</tr>
<tr>
<td>Resource/Skill type</td>
<td>( R \in {1, 2, 3, 4, 5} )</td>
</tr>
<tr>
<td>View</td>
<td>( d = 20 ) links to other nodes</td>
</tr>
<tr>
<td>Utility</td>
<td>( U \in \mathbb{R} )</td>
</tr>
</tbody>
</table>

Table 1. Nodes state

The Resource (or Skill) is the only parameter which doesn’t evolve: it is not copied during the reproduction phase (see later), 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
At each cycle with probability 0.5:
  each node \( i \) receives a certain job
  if \( (S_i = \text{job}) \) then \( U_i += 1; \) completed++; 
  else
    if \( (\text{strat}, = \text{true}) \) then for all neighbors \( j \)
      if \( (S_j = \text{job}) \) then
        \( U_i += -0.25; \)
        \( U_j += 1; \)
        completed++; 
  With probability 0.2 the SLAC algorithm takes place

Figure 1. Pseudo-code of the “ResourceWorld” model: \( S_i \) represents the resource \( i \) skill of node \( i \), \( U_i \) represents the utility of node \( i \) and \( \text{strat}, \) represents node \( i \) strategy (altruism flag).

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 (figure 1 shows the pseudo-code of the used algorithm):

- \( J = R_i; \) in this case (cf. figure 2a) \( i \) 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 \( b \) \( (U_i = U_i + b) \);

- \( J \neq R_i \land A_i = 1 \): in this case (cf. figure 2b) \( i \) 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 added or subtracted to anybody;

- \( J \neq R_i \land A_i = 0 \): in this case (cf. figure 2c) since node \( i \) is a selfish node, it can not 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.

This minimal scenario represents a situation in which user-level requests of resources (disk space, files, programs, certain devices, 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 [3] which is described in section 4.

3 The SkillWorld Scenario

The SkillWorld (SW) model [4] is very similar to the ResourceWorld (RW) model we have just described but it represents a different scenario. Here each node is marked with a certain skill \( S \), instead of a resource. 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
Figure 2. An illustration of the “ResourceWorld” model. Shading of nodes represents strategy; the number inside the node indicates the resource. In (a) node $i$ receives a certain request $J$: since it has the appropriate resource to satisfy it ($R_i = J$) it gains the benefit payoff ($b = 1$). In (b) $R_i \neq J$: since node $i$ is an altruistic node ($A_i = 1$), it can pass $J$ to its neighbor $j$ which holds the right resource ($R_j = J$); in this case $j$ earns the $b$ payoff and $i$ pays the $c$ payoff ($c = 0.25$). In (c) since node $i$ is a selfish node, it can’t pass the request to its neighbor $j$.

Figure 3. An illustration of the “SkillWorld” model. Shading of nodes represents strategy; the number inside the node indicates the skill. In (a) node $i$ receives a certain job $J$: since it has the appropriate resource to satisfy it ($S_i = J$) it gains the benefit payoff ($b = 1$). In this case we don’t care about the altruism flag. In (b) $S_i \neq J$: node $i$ will look among its neighbor list for an altruistic node; in this case $j$ holds the required skill and is altruistic, so node $i$ earns the $b$ payoff and $j$ pays the $c$ payoff ($c = 0.25$). In (c) since node $i$ has just one selfish neighbor, the job is not executed.

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.). Figure 3 gives an idea of what happens in this model.

4 SLAC in ResourceWorld and SkillWorld

We have already seen from previous works [9, 2] that SLAC (Selfish Link-based Adaptation for Cooperation) has the ability to produce high levels of cooperation in P2P networks while performing some tasks. In this work we apply the SLAC protocol to a simulated task which requires both altruism and specialization between nodes.

The SLAC algorithm specifies how nodes should update their strategies and links under the assumption that they are involved in some on-going interaction with neighbors from which they can derive utility measures. Figure 4 shows the pseudo-code of the relating algorithm.

At each cycle of the simulation task, with a certain probability ($0.2$), the SLAC algorithm is invoked. The algorithm is executed by each node: it periodically compares its own average utility
At each cycle with probability $rp$ for node $i$:
select a random node $j$ from the population:
if ($U_i \leq U_j$) then
  copy strategy from $j$
drop each link from $i$
copy each link from $j$
link $i$ to $j$
with prob(M) mutate strategy of $i$
with prob(MR) mutate links of $i$
reset utility $U = 0$;

Figure 4. The generic SLAC algorithm. Each node executes this algorithm.

(say $U_i$) with the utility of another node (say $U_j$) randomly chosen from the network. 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). As already seen, each node has a maximum view size: if a new node has to be added in an already full view, a randomly selected node is discarded to make place for the new neighbor.

The rewiring operations are symmetric: if node $i$ makes a link to node $j$ then node $j$ make a link to $i$; if node $i$ drops a link to $j$, the link from $j$ to $i$ is dropped as well. After a copying (or replication) event, “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.

To evaluate the performances of SLAC within these models we adopted a simple measure called $Pcj$ (percentage of completed jobs). We may infer that a P2P network in which most of the requests are completed, is sustaining internally cooperative and specialized groups, since in order to have a request satisfied, a receiving node which does not have the required resource must be altruistic if it wants to pass it to one of its neighbors (in the ResourceWorld case). This may be inferred even in SkillWorld, where neighbors must be altruistic to execute a job.

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. Simulations were carried on Peersim [10], an open source P2P systems simulator platform, using the Newscast protocol [6] for the management of the overlay topology. The time is divide 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 the following:

- Network size ($N$): 4000;
- Maximum degree ($d$): 20;
- Initial topology: random;
- Strategy initialization: all the nodes were initialized with a defecting strategy ($A_i = 0$);
• Resource/Skill initialization: all the nodes were initialized with a skill resource taken form a set of 5 elements \( R_i \in \{1, 2, 3, 4, 5\}, S_i \in \{1, 2, 3, 4, 5\} \);

• Links mutation (MRT): performed with probability 0.01;

• Strategy (MS) and Resource (MR) mutation: both strategy and resource mutation rate were set to 0.0025;

• Payoffs: both benefit \( b \) and cost \( c \) payoffs were ranging from 0.1 to 2.0 by steps of 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.

![Diagram](image)

**Figure 5. ResourceWorld: Map diagram indicating the number of cycles needed to obtain a good level of Pcj (> 80%).** The used parameters, are those indicated in section 5. The top left half of the map indicates no results (very low Pcj). The bottom-right half of the map indicates that good results can be obtained according to the rule \( c \leq b \). When \( c < \frac{1}{2} b \), good results are obtained in a small amount of time.

### 5.1 ResourceWorld results

As we said, the performance measure we adopted is the percentage of completed jobs (Pcj). We found that 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 5 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.
Figure 6. ResourceWorld: Average number of cycles needed to obtain a good level of Pcj exploring several cost payoffs: b = 1, c = 0.1...1. The other parameters are the same used for results in figure 5. We note that when $c \geq \frac{1}{2}b$, both average and standard deviation increase.

The above mentioned results are from an average of 10 cycles per configuration. The standard deviation ($\sigma$) depends on the ratio between the benefit and the cost payoff: if $c \geq \frac{1}{2}b$ then the standard deviation is approximately half of the average number of cycles ($\sigma \approx \frac{1}{2}n$, where $n$ is the number of cycles); if $c < \frac{1}{2}b$, then $\sigma \approx \frac{1}{6}n$. Figure 6 gives an idea of this.

5.2 SkillWorld results

We compared results from ResourceWorld 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: $c < \frac{1}{2}b$ (see figure 7). Similar to the ResourceWorld model is the result given in figure 8: 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 ($c < \frac{1}{4}b$).

6 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 2).

In RW a high number of completed jobs is obtained when the cost payoff is smaller then 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
Figure 7. SkillWorld: Map diagram indicating the number of cycles needed to obtain a good level of $P_{cj}$ (> 80%). The used parameters, are those indicated in section 5. The top left part of the map indicates no results (very low $P_{cj}$). The bottom right part of the map indicates that good results can be obtained according to the rule $c < \frac{3}{2}b$.

Figure 8. SkillWorld: from this map we can note that even here (as for the RW model) when $c < \frac{1}{2}b$, good results can be obtained in a small time.
the topology evolution could also illuminate this relationship. We have seen from previous works [3] 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.

From the point of view of actually implementing such protocols in P2P systems there are still some potential problems concerning possible malicious behavior. We have assumed that nodes follow the protocols correctly but malicious nodes could attempt to subvert the protocol by lying about their utilities and / or strategies. Although such attacks do damage system performance experiments with similar protocols have indicated that in many cases the damage may be acceptable (i.e. not completely destroy the functioning of the network) [1].

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 won’t produce social benefit following our simple approach.

Acknowledgments

Thanks go to Ozalp Babaoglu, Stefano Arteconi, Gian Paolo Jesi, Edoardo Mollona and Giovanni Rossi, all from the Department of Computer Science at the University of Bologna, for many discussions relating to aspects that influenced the paper.

References


