# SLACER: Randomness to Cooperation in Peer-to-Peer Networks*

David Hales, Stefano Arteconi, Ozalp Babaoglu
Department of Computer Science
University of Bologna, Italy
{hales, arteconi, babaoglu}@cs.unibo.it

## Abstract

*Peer-to-peer applications can benefit from human friendship networks (e.g., e-mail contacts or instant message buddy lists). However these are not always available. We propose an algorithm, called SLACER, that allows peer nodes to create their own friendship networks, through randomized interactions, producing an artificial social network (ASN) where nodes share high trust with their neighbors.*

## 1. Introduction

Human social relationships form a social network that spans the entire globe forming a "small-world" topology in which highly connected clusters of mutual friends are linked to other clusters by individuals that form "social hubs". Such friendship networks have desirable properties: they tend to be cooperative and they support a number of social functions like solving difficult problems jointly. Furthermore, they are constructed and maintained in a completely distributed manner. Cooperation between nodes is a fundamental property in peer-to-peer (P2P) networks. In fact, given the total lack of centralized control, users can *free-ride* by not respecting the rules (e.g., leeching in a file sharing system) with the risk of performance degradation or even system destruction.

For these reasons social networks are a very good candidate topology for many P2P applications. In fact, some P2P systems import existing social networks built and maintained by users in everyday life [6, 5].

Real social networks, however, are not always available, and even when they are, relate to users' social goals that usually have nothing to do with those of the P2P application.

Our approach in this work is to import *network formation processes*. We let applications maintain their own social networks tuned to their goals and we build an artificial social network (ASN) promoting cooperation between nodes from scratch.

In Section 2 the SLACER algorithm will be described; in section 3 some experimental results will be presented and discussed; finally in section 4 conclusions are drawn.

## 2. The SLACER Algorithm

SLACER (Selfish Link-based Adaptation for Cooperation, Excluding Rewiring) is an evolutionary algorithm based on previous tag models [8, 1]. Here tags are intended as nodes' neighborhoods and strategy as an application-level behavior (e.g., in a file sharing system, the Upload/Download ratio). A simple but significant application to evaluate SLACER is the famous *Prisoner's Dilemma* which we describe later.

### 2.1. SLACER

In the SLACER algorithm, each node engaged in a specific application task generates a *utility* measure ($U$). Such measures are strictly related to the application domain and can be defined in very different ways according to the application's goals. For example, they could be defined as a function of download speed in a file sharing scenario; as the number of jobs processed in a distributed computing environment; as the latency and ratio of delivered packets in a routing protocol, etc.

The higher the value of $U$ at a node, the more it believes to be performing better. Each node $i$ periodically compares its utility $U_i$ with another node $j$, chosen randomly from the network. If $U_i \leq U_j$ then node $i$ drops all of his current links with (high) probability $W$ and copies all of $j$'s links (including a link to $j$ itself) and $j$'s strategy (see Figure 1).

In SLACER, all rewiring is performed in a symmetric fashion — if node $i$ makes a link to $j$ then node $j$ makes a link to $i$, and on the other hand, if $i$ drops a link to $j$ the link from $j$ to $i$ will be dropped as well. Each node can maintain a number of neighbors that is bounded (called the *view*). If

*Partially supported by the EU within the 6th Framework Programme under contract 001907 "Dynamically Evolving, Large Scale Information Systems (DELIS)".

a new node has to be added to a view that is already full, a randomly selected neighbor has to be removed.

```
// Active thread
i=this node
do forever:
  periodically:
    j=GetRandomNode()
    if i.Utility <= j.Utility
      CopyStatePartial(j)
      Mutate(i)

// Function CopyStatePartial(j)
i.Strategy=j.Strategy
drop each link from i with probability W
i.addLink(j)
for each link in j.Links:
  i.addLink(link)

//Function Mutate(i)
with probability M mutate i.Strategy
with probability MR mutate i.Links:
  drop each link with probability W
  i.addLink(SelectRandomNode())
```

**Figure 1. The SLACER algorithm.**

The idea behind the algorithm is that a node copies the behavior of others performing better than itself and moves from lower to higher utility zones. What arises from this is a sort of *tribalism* [2] in which the network is structured into highly clustered groups (tribes) with nodes moving to tribes with better performance. To avoid the tribalism from becoming too strong, hence leading the network to partition, each node does not drop completely all of its links when moving, but keeps them with (low) probability $1 - W$. No specific initial topology is required: random, small-world as well as disconnected graphs all produce the same result — a connected and cooperative small-world topology. We assume a random node can be sampled from the entire population when utility comparisons are made. This is feasible using a lower level peer sampling service such as *Newscast* [3] that maintains a connected random topology robust to high network dynamicity.

### 2.2. The Prisoner's Dilemma

In the Prisoner's Dilemma (PD) two players interact by selecting a move — to "cooperate" (C) or to "defect" (D) — without knowing in advance the opponent's move. For each of the four possible outcomes of the game (illustrated in Table 1), players receive the respective payoffs. The payoffs are constrained by the relations: $T > R > P > S$ and $2R > T + S$. These constraints give rise to the dilemma: both the players are pushed towards selfish behavior (D) since a selfish player will never have a payoff lower than its opponents', can reach the highest payoff and it never gets the lowest one. In other words (D,D) is a *Nash equilibrium* [7] for the single round PD. On the other hand, note that if both players cooperate, then the collective payoff is the highest possible.

| | C | D |
|---|---|---|
| C | R,R | S,T |
| D | T,S | P,P |

**Table 1. Prisoner's Dilemma payoff table.**

The PD game is useful since it captures, in an abstract form, a situation in which there is a tension between collective and individual self interests. We used PD as a test application for SLACER. In our experiments, the application level consisted in periodically playing a single round PD with a randomly selected neighbor. What we obtain is that from an initial state where everyone is defecting, a *cooperation seed* consisting of two or more linked cooperating nodes is eventually produced through mutation. Suddenly, this seed grows and expands throughout the whole network. When a high cooperation level is achieved, it is very stable because mutation to defection will cause the mutated node's neighbors to have a lower utility and to move to higher utility zones isolating the defecting node.

## 3. Experimental Results

All of our experiments were performed using the P2P network specific simulator *peersim* [4]. All of the reported results are averages of 10 experiments, except where stated otherwise. We set the PD payoffs to $T = 1$, $R = 0.8$, $P = 0.1$, $S = 0$. We set the mutation rates to $M = 0.005$ and $MR = 0.01$. Nodes have a view size equal to 20, hence each node can link to maximum of 20 neighbors. As stated earlier, if a node has to add a new neighbor to its already full view, a randomly selected neighbor is removed.

### 3.1. Evaluation Parameters

We describe briefly the network properties that were evaluated in the experiments.

**Clustering Coefficient.** Let us define the network as a graph with a set of vertices $V = v_1, v_2...v_n$ and edges $E = V \times V$ with $e_{ij} \in E$ denoting a link between $v_i$ and

$v_j$. Node $i$'s view is defined by $N_i = \{v_j\} : e_{ij} \in E$ and its clustering coefficient is given by equation 1.

$$C_i = \frac{|\{e_{jk}\}|}{k_i(k_i - 1)} : v_j, v_k \in N_i, e_{jk} \in E \qquad (1)$$

Clustering Coefficient (CC) measures the proportion of links between the vertices within each node's neighborhood divided by the number of links that could possibly exist between them. In other words, it counts how many times two distinct neighbors of a node are each others neighbor. The entire network's clustering coefficient is given by the average of the clustering coefficients of the single nodes.

**Average Path Length.** Average Path Length measures the average number of hops needed to connect any pair of nodes in the network.

**Largest Cooperative Cluster.** Largest Cooperative Cluster (LCC) measures the ratio of cooperative nodes in the largest weakly connected cluster in a subnetwork consisting only of cooperating nodes. The highest LCC value, given by a fully cooperative, not partitioned network, is equal to 1.

**Connected Cooperative Path.** Cooperative Connected Path (CCP) measures the ratio between pairs of nodes connected through paths consisting only of cooperative nodes, and all possible pairs of nodes in the network. The maximum possible value is obviously 1. A network does not need to be fully cooperative to reach the maximum value since it could be possible to find alternative cooperative paths to avoid defective nodes. On the other hand, a network cannot reach the maximum value even if it is fully cooperative but it is also partitioned. This is because in this case, there are no paths between nodes in different network partitions.

### 3.2. Typical SLACER Behavior

A single SLACER run is illustrated in Figure 2 to show the trend of cooperation formation. To evaluate the tribe formation process, clustering coefficient is probably the most useful parameter. At the beginning, there are no cooperating nodes and the clustering coefficient is very low. As the nodes start reproducing, clustering coefficient grows, meaning that through rewiring a small-world like topology is created, even if cooperation remains very low. At around cycle 150 cooperation emerges via mutation and a curious thing happens to the clustering coefficient: it takes a low dip, suggesting that as the cooperation seed is formed, many nodes join to it lowering the clustering coefficient. This increases network randomness since each node's neighborhood size is bounded. When high cooperation is reached, a high clustering coefficient is restored, indicating that a small-world like topology created through SLACER rewiring similar to the one present before the cooperation was achieved is reestablished.

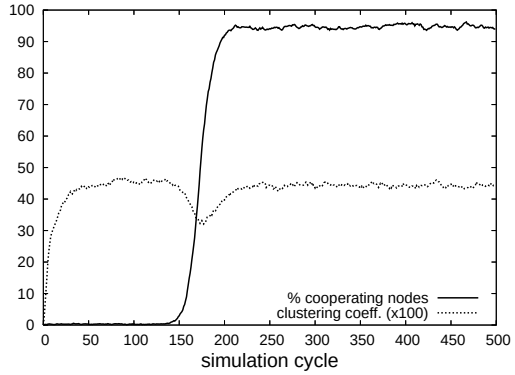



**Figure 2. Cooperation trend and clustering coefficient in a single SLACER run.**

### 3.3. Time to Cooperation

Time needed to achieve a given level of cooperation for different network sizes is illustrated in Figure 3. It is interesting to note the inverse relation between network size and time to cooperation: the larger the network, the smaller the time needed to achieve cooperation. This property has a probabilistic explanation. In fact cooperation emerges as two neighboring nodes mutate to cooperation at the same time and interact with each other at least once. Since mutation rates and maximum view size are independent of network size, it is obvious that in a larger network is easier to create a cooperation seed through mutation and not to have it destroyed immediately by defective neighbors.

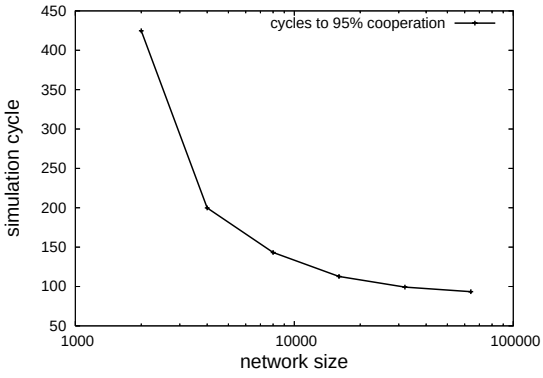



**Figure 3. Time to achieve 95% cooperating nodes with different network sizes.**

### 3.4. Tribalism

In Figure 4 two different kinds of tribalism are illustrated by setting the link drop probability ($W$) to 1 and 0.9. We measured the size of the largest cooperative cluster (LCC) and the cooperative connected path (CCP). Notice that for $W = 0.9$ the largest cluster includes almost all the nodes and CCP is very high, indicating that no defective node occupies a position that allows it to block a large number of unique cooperative paths between pairs of nodes. For $W = 1$ instead, there is a sort of *extreme tribalism* leading to a highly partitioned network, hence a very small CCP since even if cooperation is high there are no links connecting the different tribes. Also note that while $W = 0.9$ scales well, for $W = 1$ performance decreases when network size increases.

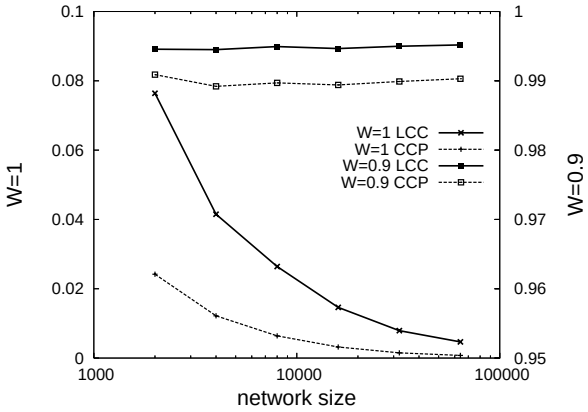



**Figure 4. CPP and LCC for W=1 and W=0.9. Note the very different scales for the two vertical axes.**

### 3.5. Topology

Figure 5 shows the different topologies created by SLACER for different network sizes. For different network sizes the clustering coefficient remains almost constant while average path length grows logarithmically, hence showing good scalability properties. One interpretation of these results is that the network remains small-world like, with a growing number cooperating clusters.

### 3.6. Robustness to Churn

To test SLACER robustness to nodes joining and leaving the network, different churn patterns have been evaluated on a 4000-node network: a continuous substitution of random nodes with ones that always defect (Figure 6(a)), a periodic substitution of 200 nodes every 50 simulation cycles

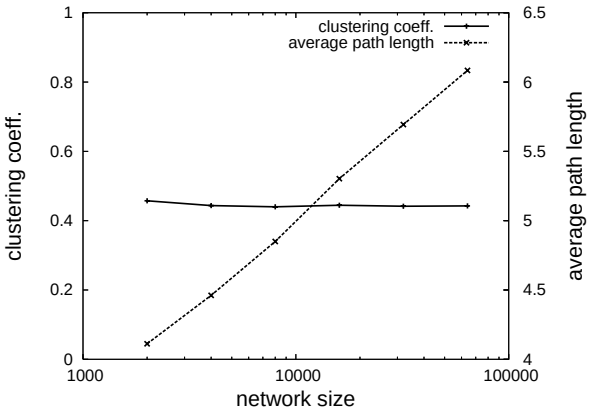



**Figure 5. Clustering coefficient and average path length for W=0.9 as a function of network size.**

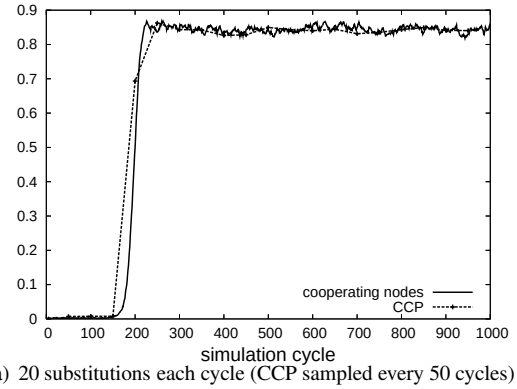
(Figure 6(b)), and a single catastrophic substitution of 2000 nodes (half of the network) every 200 cycles (Figure 6(c)). Inserted nodes always defect and are linked to 20 (maximum view size) random nodes each in all the experiments.

The results obtained are shown in Figure 6. SLACER recovers fast from periodic node failures while cooperation level is decreased when there is a continuous and relatively high dynamicity in the network. This is plausible since in this case the network is invaded by defecting nodes at a higher rate than it can effectively recover through reproduction. Nonetheless, cooperation remains stable at a relatively high level. Moreover, network changes (even catastrophic ones) don't affect cooperation formation. In fact, in the reported experiment they occur before, during and after cooperation formation without causing any irreversible performances degradation.

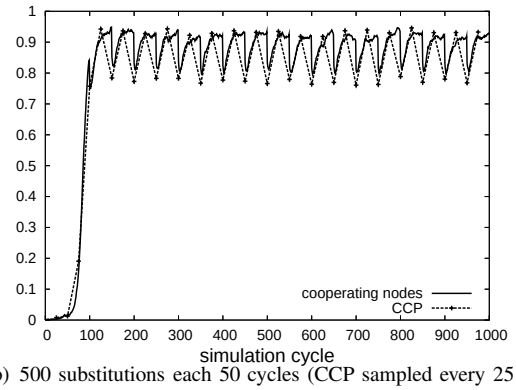
## 4. Conclusions and Future Work

The SLACER algorithm has been presented and tested with the PD game. Previously we had demonstrated how PD results can be translated into more realistic P2P scenarios (e.g. altruistic file-sharing) [2]. SLACER networks self-organize a cooperative small-world topology, optimizing application level performance by inducing nodes to produce cooperative behavior. The key operations (utility comparison and node copying) are performed with randomly selected nodes and are scalable with respect to network size. We believe that existing P2P algorithms, that require existing human social networks as input [5, 6], could be made completely autonomous by using SLACER to supply them suitable artificial social networks (ASN).

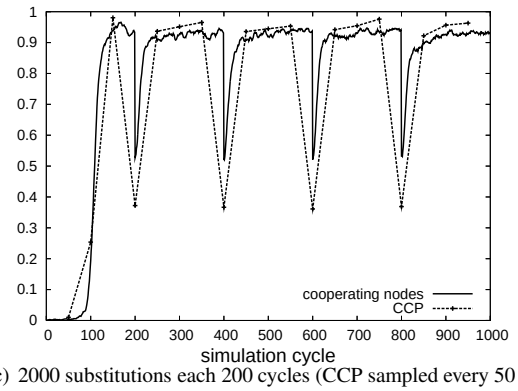
Even though the proposed PD application corresponds well to the problem "selfish interest versus social benefit"




(a) 20 substitutions each cycle (CCP sampled every 50 cycles)




(b) 500 substitutions each 50 cycles (CCP sampled every 25 cycles)




(c) 2000 substitutions each 200 cycles (CCP sampled every 50 cycles)

**Figure 6. Cooperation level and CCP for different churn scenarios. Results for a single run are reported for a network of 4000 nodes. The trend is the same for all other tested configurations.**

which is common to most P2P systems, it would be interesting to evaluate SLACER performance when dealing with more realistic applications. Some possible candidates are: file sharing systems to optimize upload/download ratio, content delivery networks that self organize a content distribution policy, DHT systems to improve routing and searching performances, and collaborative anti spam or anti spyware systems.

## References

[1] D. Hales. Cooperation without memory or space: Tags, groups and the prisoner's dilemma. In *MABS '00: Proceedings of the Second International Workshop on Multi-Agent-Based Simulation-Revised and Additional Papers*, pages 157–166, London, UK, 2001. Springer-Verlag.

[2] D. Hales and B. Edmonds. Applying a socially-inspired technique (tags) to improve cooperation in p2p networks. 35(3):385–395, 2005.

[3] M. Jelasity, W. Kowalczyk, and M. van Steen. Newscast computing. Technical Report IR-CS-006, Vrije Universiteit Amsterdam, Department of Computer Science, Amsterdam, The Netherlands, Nov. 2003.

[4] M. Jelasity, A. Montresor, and O. Babaoglu. A modular paradigm for building self-organizing peer-to-peer applications. In G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, editors, *Engineering Self-Organising Systems*, number 2977 in Lecture Notes in Artificial Intelligence, pages 265–282. Springer, 2004.

[5] J. S. Kong, O. P. Boykin, B. A. Rezaei, N. Sarshar, and V. P. Roychowdhury. Let your cyberalter ego share information and manage spam, May 2005.

[6] S. Marti, P. Ganesan, and H. Garcia-Molina. Dht routing using social links. In *IPTPS*, pages 100–111, 2004.

[7] J. Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences of the United States of America*, 36:48–49, 1950.

[8] R. Riolo, M. D. Cohen, and R. Axelrod. Cooperation without reciprocity. *Nature*, 414:441–443, 2001.