Decentralized Lightweight Methods for Coordination and Cooperation in Peer-to-Peer Networks.[*]

Stefano Arteconi
Universitá di Bologna
Dip. Scienze dell'Informazione
arteconi@cs.unibo.it

February 24, 2006

Abstract

Peer-to-Peer networks definition implicitly includes features like decentralization and dynamicity that can be exploited to achieve interesting emerging properties such as self-organization and self-configuration.

On the other hand cooperation between nodes is a necessary condition for peer-to-peer system to work well, even though the lack of central control often gives incentives for nodes to behave selfishly. Our work explores novel methods that encourage nodes to act in a cooperative coordinated fashion even when there are incentives to behave selfishly.

With respect to peer-to-peer characteristic, such method should be based on local nodes' interactions and knowledge and face possible large scale network dynamicity, which makes cooperation between nodes a hard and stimulating challenge.

Our attempt to deal with such challenges draws inspiration from different disciplines as sociology, biology and economics. So far interesting result regarding obtained cooperation level, topology, and system performance against cheating nodes have been obtained, but there are still many issues to face.

[*]Ph.D. Thesis Proposal.

1 Introduction

Peer-to-Peer (P2P) network paradigm is a relatively recent approach to distributed systems and network computing. Its main innovation with respect to the classic client-server one is given by the fact that no distinction is made between client and server machines, in fact from the P2P point of view all the nodes in the network are ideally both client and server at the same time. This means that a P2P node does not only generate requests or provide certain services, but it should be able both to ask for services and to provide them at the same time. It is quite easy to see that the key point of such paradigm is *decentralization*. There is no powerful trusted server to control the system and to provide services to small clients, on the contrary, resources, as well as services, are scattered throughout the network.

Another important feature of P2P systems is that they are usually defined as *overlay networks*, meaning that no physical network infrastructure is taken into account, on the contrary transport and network level ensuring connection between nodes are assumed to exist (i.e. the Internet and TCP/IP protocol). As a consequence to this the nodes of a P2P overlay are defined just as processes running in a given machine, while links between nodes are just logical labels (i.e. IP, port pair). In such setting rewiring of nodes, hence network topology modification and management, are very easy to perform since no physical channel modification is needed. A third important feature of P2P systems is dynamicity, that is, many nodes continuously join and leave the network.

Being P2P systems implicitly highly decentralized, dynamic, and based on simple local nodes' interactions, they can be related to the studying of *complex systems*. This definition includes many systems, such as ant colonies, immune system and friendship formation, drawn from very different disciplines. What links this very different settings together is the lack of centralized control and, on the contrary, the ability to set up large systems just from local interactions and knowledge.

The passage from local to global is the key point in P2P systems (and more generally in complex systems), when designing a system of this kind, it is only possible to define local rules and behaviors to be followed by each node, while the formation of a working system on a global scale is cannot be directly designed or implemented, but should appear in the system as an *emergent behavior* obtained only from simple local interactions between simple agents having just a limited local knowledge of the system.

The passage from client-server to P2P paradigm has a lot of different (not always positive) effects: on one hand the system implicitly become more robust (no single point of failure) and scalable (no bottlenecks), moreover performances could be easily improved (higher use of *leaf nodes* resources), but on the other hand decentralization also leads to a lack of system control, hence to the possibility to *free-ride*, and the difficult to define and set up global behaviors.

It turns out, finally, that when designing P2P systems not only system performances optimization should be taken into account, but a way to ensure coordination between nodes and to avoid malicious behavior should be designed as well, so to improve global system performance starting from a local basis. Taking into account that everything has to be done locally, has to be as simple as possible, and no global knowledge, coordination, or control should ever be assumed, dealing with these tasks turns out to be very hard.

2 Problem Definition

Given the peculiar P2P systems features described above, the main problem that is going to be addressed consists in defining a general mechanism to produce *coordination* and *cooperation* between nodes in a P2P network in a totally decentralized lightweight fashion. Even though these problems are quite easy to deal with in presence of centralized controlling entities, they become hard tasks to solve under P2P systems implicit constraints.

Since the goal to achieve has been defined as *cooperation* (or in a broader sense *coordination*) between nodes, a definition of such terms will be given now.

2.1 Need for Cooperation in P2P Systems

The extreme decentralization typical of P2P networks, hence the total lack of centralized and controlling entities, makes P2P systems *open*, in the sense that there is no direct control neither on who joins the system, nor on nodes behavior.

In practice nodes can freely join such systems and *free-ride*, that is they can assume a behavior that could lead to system performance degradation or could directly act against system protocol specification.

Basically what happens is that decentralization, and the subsequent ability to free-ride, lead to a struggle between selfishness and cooperation. In fact what usually happens is that nodes benefit from having their requests fulfilled, but on the other hand they pay a fee to provide services to other nodes. In this basic abstract scenario each node is brought to behave selfishly, trying to maximize its benefit, while minimizing costs, hence denying services to other nodes. It is quite easy to notice that if every node acted selfishly no service would ever be provided, therefore nodes cooperative behavior is a necessary condition from a global system performance point of view. The contradiction between selfishness and cooperation typical of open systems where basic entities are not supervised by any kind of central control is known as *the tragedy of the commons*[14].

Turning to free-riding, an easy to understand example of free-riding in P2P is given by leechers (users downloading files but not sharing anything) in file-sharing systems, obviously a high number of leechers would prevent the system from working. In such example coordination could consist in having all the nodes sharing resources so that anybody could easily find and retrieve the files he is interested in. The bast way to coordinate nodes behavior is not trivial anyway: should any node give as much files (or bytes) as it get? Should upload/download ratio be distributed according to nodes capability (i.e. bandwidth)? What is the optimal topology?

It could be very hard, if not impossible, to define an optimal (or at least acceptable) solution to satisfy these requirements "a priori" on a global scale.

2.2 Prisoners' Dilemma

The simple file-sharing example above shows how hard the problem of coordination could be and how many different aspects it involves. A classic example to describe the contradiction, typical of such kind of systems, between selfish and cooperative behavior in a more general and formal way is the well-know *Prisoners' Dilemma* game (PD)[4]. In PD two players play one against the other and could choose between two possible strategies: Cooperate (C) or Defect (D), payoffs are then distributed to the players according to the pair of played strategies as illustrated in table 1.

$P1$ | $P2$	C	D
C	R, R	S, T
D	T, S	P, P

Table 1: Prisoners' Dilemma payoff table.

For the dilemma to originate the two constraints expressed by equations 1 and 2 must hold:

$$T > R > P > S \tag{1}$$

$$2R > T + S \tag{2}$$

Under such constraints D (defective) strategy always ensures a payoff equal or higher than the opponent's one, moreover D is a dominant strategy for both players, hence (D, D) is a Nash equilibria [24] for PD, that is, if PD constraints hold, nodes are implicitly pushed to defection. In spite of this, the highest possible total payoff is obtained when both the players choose a cooperative strategy. To sum up, PD pushes single players to defection but ensures better global performances to pairs of cooperative players, this is exactly what happened in the previous file-sharing system example, and what happens in general in many open systems.

How to deal with such problem is the main aim of the present work. Some of the existing mechanisms to do this, and a novel one in development stage will be presented in the following.

3 State of the Art

P2P system can be used with very different application domains, anyway free-riding is a problem common to any kind of P2P system, and trying to find a way to deal with this problem is a major issue within the P2P community. Looking at the problem in an abstract way, with no distinction between different applications, open systems are bound to suffer from the contrast between the cooperation needed for the system to work and the selfish behavior of users who wants to maximize their income from the system without hardly paying anything (being it money, bandwidth, computational resources, etc.) back.

In P2P such phenomenon is usually called *free riding*, many analysis have been performed on existing P2P system to study its size and impact [1][27]. For such systems to work under high free-riding rates, balancing nodes providing many services (and paying high costs) are needed, hence the effort needed to keep the system working is unevenly distributed between peers and the actual P2P nature of the system is lost.

In addition to the problem of free-riding, P2P system should also be able to deal with cheating nodes. In open systems usually identities are very cheap, hence the most common (and feasible) attacks are based on false identities. Two example are given by the so called *sybil attack* [8], where basically a node uses many different identities at the same time, and *whitewashing* [10], where a node continuously changes its identity.

Researchers have been proposing many different attempts to deal with these problems and to prevent users from free-riding or misbehave and to identify and isolate such users by avoiding interactions with them.

In [19] Eigentrust, a mechanism to evaluate and assign to each node a trust value is defined, the basic idea is to aggregate a global trust values for each node based on the result of past interactions, and to store them in a distributed hash table (DHT) [25][28][26] so to make them easy to be retrieved and updated by any node.

A different approach is proposed in Samsara [6], an extension to a P2P distributed storage system, called Pastiche [7], that ensures that nodes consume no more resources than they contribute. Samsara is not based on trust values, the mechanism used to obtain this is based instead on reciprocity between nodes (that is, if node A stores data on account of B, then B must ensure to eventually store data in account of A), and probabilistic punishment inflicted to nodes that do not respect such reciprocity.

Another proposed solution, very different from the two cited above is given in [2]. Here methods drawn from "classic" distributed systems such as replicated state machines [5] are used to tolerate both *byzantine* (including broken, misconfigured, or malicious nodes) and *rational* (selfish free-rider nodes) behaviors. To obtain this, the strong requirement of a close system is needed, that is there has to be a central authority deciding which nodes may enter the system and which may not.

An alternative approach able to deal with free-riding nodes as well as cheating and colluding ones is described in [9]. The main idea in this work is given by the fact that in addition to a shared history maintained aggregating all the nodes interaction in a DHT (somewhat similar to the Eigentrust approach), to avoid colluders each node also evaluates a subjective trust value for other nodes through a maxflow algorithm where the generic (i, j) edge capacity $c(i, j)$ represents the amount of services node i received from node j. This approach seem to resolve all the problems in a clever simple way, its main limit anyway consists in the fact that since all new nodes joining the system are considered as a unique entity, whitewashers can potentially cause the system to become too hard to join, in the sense that newcomers (even non cheating and non selfish ones) will be initially punished as if they were cheating nodes, hence they might be discouraged from joining the network.

Not all the existing techniques are based on reputation or detection techniques explicitly hard-coded into the protocol. An alternative approach consists in using social networks formed by users as a network topology to perform collaborative task and to choose trusted "friend" nodes.

In [20] a collaborative spam-filtering technique based on email network is described. Here the network topology is indirectly defined by users' email contacts. Basically one's immediate neighbors in such network are the people contained in his email contact list. Each user then defines some level of trust on its neighbors and marks the received spam mails. Finally spam-filtering is performed upon new mail arrival searching the network to check if received messages have already been marked by some trusted node.

A similar approach, even if applied on a quite different scenario, is described in [21]. Here a social network defined by instant messengers (such as ICQ[1], MSN Messenger[2] and Skype[3]) contact lists is used to define alternative path routes for a DHT system. To send or forward a packet a node can choose to use a "friend" node defined by the social network rather than directly following DHT specification and use an unknown (and possibly untrustworthy) node. This simple mechanism ensures higher reliability than a simple DHT at the expenses of a less uniform traffic load distribution due to the very different connectivity between nodes in the social network.

Even if social networks usually have very interesting properties as illustrated in section 4.1, using pre-existing ones might not always be a good idea. The point is that social networks formation is usually led by some specific goal, hence the resulting topologies could be biased to work for some specific tasks, but not for others. In example user A could enjoy having a chat with user B, but B's computer can be misconfigured as well, so a good link in a social network obtained by email or an instant messenger contact list could even be a bad link to perform other kinds of task as spam-filtering or message routing.

4 Approach to the Problem

The presented problem do not seem to be a typical computer science one, in fact it involves many other different research fields: entities interacting and getting rewards/costs according to their behavior are studied in *Game theory*, while different aspects of the way such entities should (or are bound to) behave, as well as the way acquaintances between them are formed, are studied in *economics* and *sociology*. Moreover P2P networks are often inspired by various disciplines as *biology* and *physics*, whose studied systems share many common features and techniques notwithstanding the very different settings.

4.1 Sociological Inspiration

What we want to obtain here is the ability to define a *network process formation* through which both network topology and nodes' behavior evolve

[1]http://www.icq.com
[2]http://messenger.msn.com
[3]http://www.skype.com

according to the system performance improvement in spite of nodes' selfishness, in this way it is possible to bring coordination between nodes both from an individual behavior and from a network topology point of view. Regarding the present approach, we found many resemblance between the need for coordination and joint cooperative problem solving in P2P networks, and the way human friendship relations are formed.

A good source of inspiration to define such kind of network evolution could be drawn from *social networks* formation, that is, the way human social relationships are formed. Network of people acquaintances form a *small-world* topology [31], in which highly connected clusters of mutual friends are linked to other clusters by individuals that form social bridges or hubs. The upshot of this, as has been dramatically demonstrated [29], is that it is often possible to find a short chain of friends of friends for any pair of people around the world. Such kind of friendship networks have several desirable properties: they tend to be cooperative (we trust our friends), and they support a number of social functions that we need to achieve in everyday life (i.e. we can ask friends for favors or cooperatively solve a difficult problem with work colleagues, we can turn to friends for good advice or discuss with them confidential issues). Furthermore, they are constructed and maintained in a completely distributed way: there is no central authority deciding who should be your friend or who should not.

4.2 Tag Systems

To import social networks features into P2P systems the proposed approach uses techniques drawn from *computational sociology*. It is a recently developed and rapidly spreading branch of sociology that uses computer simulation to analyze social phenomena. It involves the understanding of social agents, the interaction among these agents, and the effect of these interactions on the social aggregate.

In particular the model we are going to use is called *tag system* [15]. *Tags* are defined as markings or social cues that are attached to individuals and are observable by others. Tags evolve like any other gene in the genetic pool. The key point is that the tags have no direct behavioral implication for the agents carrying them. Through indirect effects (such as biasing of interaction), however, they can evolve from initially random values into complex ever changing patterns that serve to structure interactions between individuals. There exist in fact a lot of different models in which through tag systems cooperative behavior between agents emerges even though the environment pushes them to selfishness. The main aim here is to translate the concept of tags from multi-agent systems into P2P computer networks, allowing nodes' behavior and network topology to *evolve*.

The main proposed idea consists in embedding tag systems into P2P networks (opportunely modified, if necessary, to fulfill P2P system constraints and requirements) so to make the nodes themselves able to evolve local behaviors and topology in order to achieve (*emerge*, from a complex system point of view) coordination needed to obtain high global system performance level.

4.2.1 How does Tags Work?

Before going on to show the work that has been done from a P2P point of view, let us give a brief general explanation of how tag systems work. Each agent state in a tag system is characterized by 3 values:

- **Strategy**: The definition of the behavior of the agent in the system (i.e. PD strategy to play).
- **Tag**: Tags can be defined in different ways (bitstrings, real values, etc.), the important things about tags are:
  - Tag should not affect agent's behavior.
  - Tag should evolve.
- **Utility**: A measure of the agent's performance (i.e. PD payoff obtained).

While nodes' interactions are described through strategies, tags can be used to guide the way opponent agents are chosen. In example agents could be forced to interact only with others having their same tag or a tag similar to their one. Finally the result of such interaction is described by utility value.

Through tags evolution a mechanism of clustering formation between agents is implicitly defined. To give a more detailed description, the actions performed by a single agent (say i) in a tag system are:

- Periodically play with a node having same or similar tag.
- Update utilities accordingly.
- Periodically compare utility with a random agent (say j).
- The node with lower utility (say $U_i < U_j$).
  - i copies j's strategy.
  - i copies j's tag.

- i resets its utility.
- with low probability i mutates its tag and strategy (assigning them random values).

Through this simple mechanism tags define groups of interacting agents, and through strategies and tags evolutionary mechanism, an evolution in agent groups and behaviors is defined as well. Basically what happens through this mechanism is that poor performing agents copies the behavior and start interacting with the same set of agents of the better performing ones.

Turning back to the sociological metaphor tag systems simply describe the way people try to improve their lifestyle (utility) by joining better performing groups of people (identified by tags) and imitating their behavior (strategy).

Turning back further to P2P systems, tag systems seem to preserve P2P main constraints, in fact system evolution is performed in a totally decentralized and local fashion and agents interactions are very simple, in fact behavior could even have a complex definition, but the basic step through which it evolves (utility comparison, and copy of strategy and tag) is performed without any need for agents' rationality or specific computational capabilities.

Now that tag systems have been briefly explained and similarities with P2P networks have been pointed out we can finally turn to the result obtained so far from tag system embedding in P2P.

5 Obtained Results

The effort to use tag model in P2P system led to the definition of an algorithm, called SLAC[13], able to deal with the problem of *free-riding* and to bring the network to high level of cooperation. Then starting from SLAC some modification and analysis have been performed that lead to further algorithm improvement [12] and to interesting results regarding the presence of cheating nodes[3].

The phases of this work will now be examined step by step. So far SLAC and SLACER have only been tested through simulations. All the results presented in the following have been obtained using Peersim[4] simulator[17].

[4]SLAC and SLACER peersim implementation can be found at: http://peersim.sf.net

5.1 SLAC Algorithm

In the passage from tags to P2P the relation between agents and nodes is straightforward, while the passage from agent's state to node's state is not immediate. In P2P networks nodes strategies and utilities should be defined at application level, in example in a file-sharing scenario strategy could be the download/upload limits ratio, while an utility measurement could consist in the quantity of downloaded data or the actual download speed. About tags, as said before they should be used to limit the interactions between nodes, hence the way we translated the concept of tags into P2P is looking at nodes' view (the list of immediate neighbors) as a tag itself, and allowing direct interactions only between immediate neighbors, hence copying other nodes tag simply means rewiring one's links to move from a zone of the network to a different one.

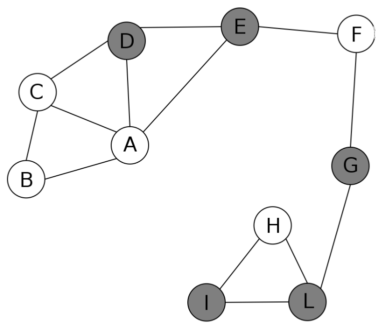
After this conceptual shift from tag systems to P2P we can finally describe the SLAC algorithm. The best way to do this is looking at the pseudo-code in figure 1 and illustrating the main performed actions.

```
Each node periodically do
  Compare utility with a randomly chosen node
  if the other node has higher utility
    Drop all the current links
    Link to that node and copy its strategy and links
    Reset utility
    Mutate with low probability links, strategy
  fi
od
```

Figure 1: SLAC algorithm pseudo-code.

The main evolutionary step performed are *reproduction* and *mutation*. Reproduction (illustrated in figure 2) consists in comparing utility with a random node (in our experiments random peer sampling is provided at lower level by NEWSCAST algorithm [16]), and in the copying of higher utility node by lower utility one. Even though what really happens is that a node moves and changes strategy, this step can be seen as the lower utility node dying out and the higher utility one reproducing.

Mutation is performed after reproduction by the losing nodes with low probability. Tag mutation (illustrated in figure 3) consists in dropping all the current links and linking to a randomly chosen node, while strategy mutation should be defined at application level and in general it consists in randomly



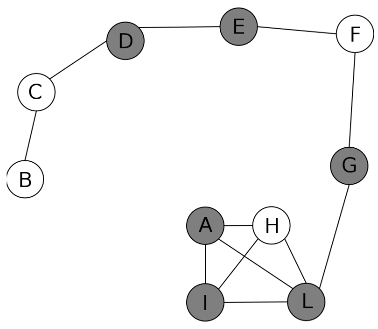
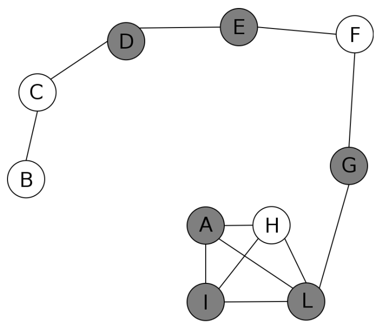


Figure 2: SLAC reproduction step. Nodes' color representing strategies. In the given example node A compares utility with node I (on the left). Supposing $U_A < U_I$ node A copies node's I tag (view) and strategy.

choosing a strategy out of all the possible ones.



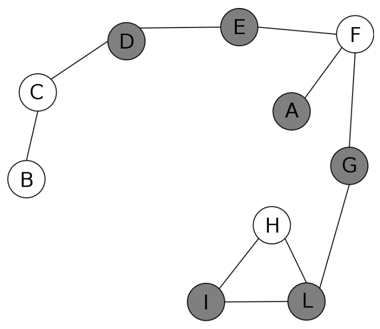


Figure 3: SLAC tag mutation. In this example node A mutates its tag (neighborhood). On the right mutation has been performed linking to a randomly chosen node. Notice that tag mutation do not affect strategy.

It has been observed that these simple basic steps bring to the formation of a highly cooperating network even when starting from full defection. To show how this happens a simple PD game application testbed has been used. PD has been chosen because as pointed out in section 2.2 it is very simple and represents in an abstract way the contradiction between selfishness and cooperation.

Before moving on to analyze result with PD let us sum up system architecture. 3 different layers have been defined: random peer sampling (used for reproduction), SLAC itself, and an application layer (providing utility). Such architecture and the interactions between different layers are shown in figure 4.

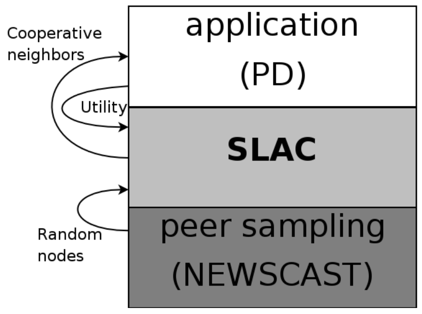


Figure 4: SLAC 3-layers architecture.

In our PD implementation each node periodically chooses a random neighbor and plays a single round with it. Nodes can only use pure strategies (always C or always D) and the utility measurement applied is just the average payoff obtained from repeated game interactions.

Figure 5 illustrates the trend of cooperation level and clustering coefficient of the network in a single simulation run.

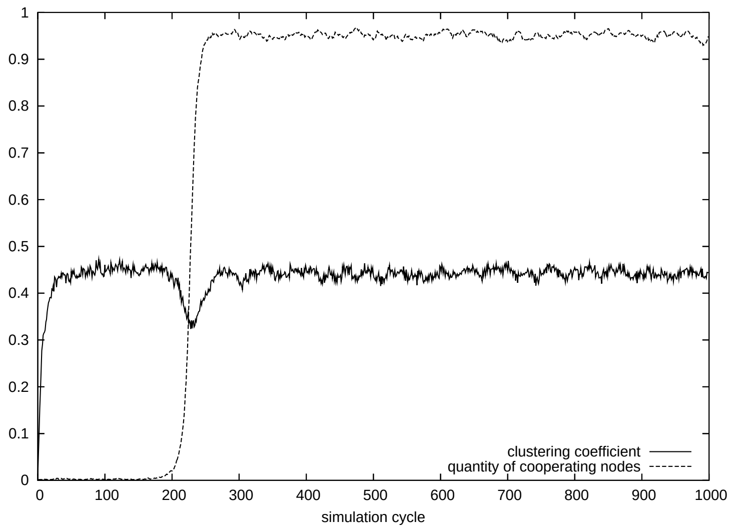
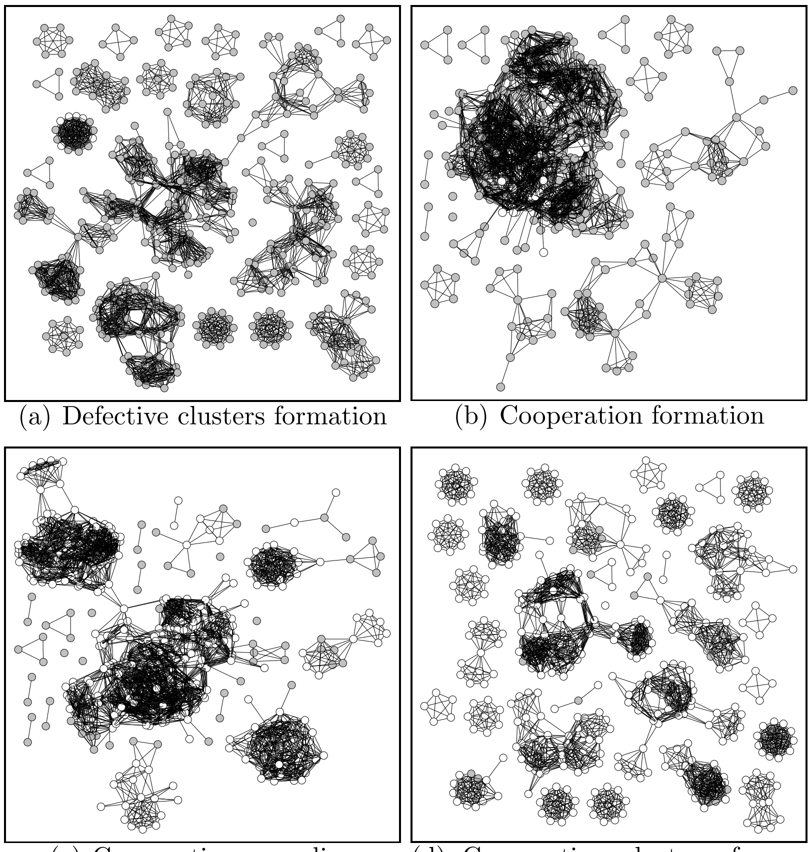


Figure 5: SLAC single run cooperation and Clustering coefficient trend. The behavior is basically the same in all performed simulations.

To have a stronger intuition of what is happening some network snapshot are given in figure 6.

Now we can easily guess what kind of process is going on:

- Through random reproduction selfish clusters are formed (figure 6(a)).



(a) Defective clusters formation (b) Cooperation formation

(c) Cooperation spreading (d) Cooperative clusters formation

Figure 6: SLAC 500-node network snapshots. White nodes are cooperative, gray ones are defective. Unfortunately the pictures are confused but can be helpful to understand network topology evolution.

- Eventually two neighbor nodes mutate to cooperative strategy and get higher utility than the rest of the network from their mutual interaction.
- Cooperation level rises fast through reproduction, bringing all the nodes to a big cooperative cluster (lower clustering coefficient) (figure 6(b)).
- When high cooperation is reached some node turn to defective strategies(figure 6(c)).
- Neighbors of defecting nodes have lower utility and move away.
- Cooperative clusters are formed and defective nodes are isolated (figure 6(d)).

- Both cooperation level and clustering coefficient remain stable.

In conclusion tags in P2P implemented by the SLAC algorithm seem to produce an emergent group-like selection between clusters of nodes, bringing nodes to cooperation in spite of an individual temptation to defect. Moreover once high cooperation is achieved it remains stable, in fact cooperating nodes being suckered by selfish ones are able to move away through reproduction, isolating defecting nodes and forcing them to eventually turn back to cooperation. Even if for the sake of simplicity all the example given here are based on PD application SLAC has been tested with more sophisticated application like a simple file-sharing model [13] and in different setting requiring not only cooperation but a higher level of coordination in the form of the grouping of nodes with different properties [11].

5.2 SLACER Algorithm

The passage from tags to P2P performed with SLAC led to cooperating networks, but on the other hand it produced a highly disconnected topology. Such kind of topology could not be acceptable in many application domains, so a step further has been done trying to keep the network connected. To achieve this a very simple modification to the original SLAC algorithm has been done: when reproducing or mutating, nodes do not drop all of their current links, but retain each of them with low probability.

This simple step produces a topology similar to the one in figure 6(d) with the addition of occasional edges linking different clusters (an example of topological differences between SLAC and SLACER is given in figure 7). Basically what we get is a Watt-Strogatz like network [31] where the analogous of the β parameter [31] is the probability of dropping links when reproducing. It is easy to see that drop probability equal to 1 is but SLAC, while the lower the drop probability the less small-worldish and more random-like the resulting network turns out to be. Since high clustering is the key point in cooperation formation and in similarities between the concepts of tag and view, our aim was to keep drop probability as high as possible while trying to keep the network connected.

From different experimental settings observation we found that drop probability value equal to 0.9 lead to network connection and to a small-world like topology as expected, this fact is indirectly proved by relatively high and stable clustering coefficient and low and logarithmically growing average path length as shown in figure 8.

To measure the relationships between network topology and cooperation

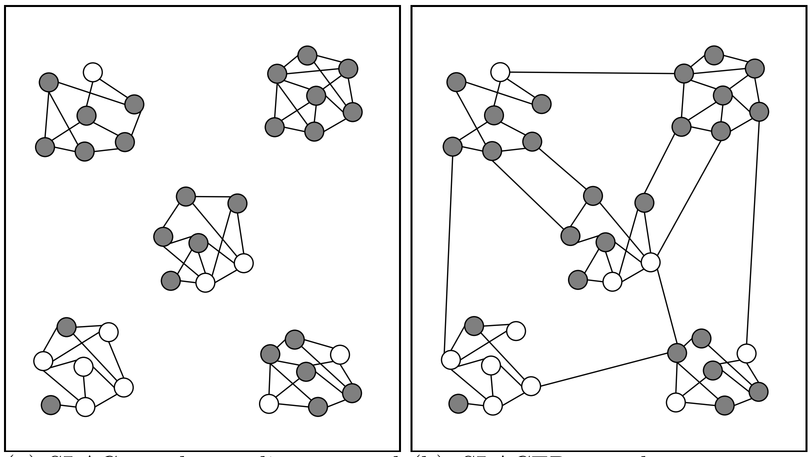

(a) SLAC topology - disconnected clusters (b) SLACER topology - watts-strogatz like

Figure 7: SLAC vs. SLACER: topological differences.

two new measurements have been introduced: Largest Cooperative Component (LCC) and Connected Cooperative Path (CCP).

LCC measures the proportion of the largest cluster composed only by cooperating nodes with respect to the network size. CCP measures the proportion of pair of cooperating nodes linked by cooperative paths (or either directly linked) with respect to all the possible pairs of nodes in the network. Value 1 means that each pair of nodes is linked either directly or by a path entirely composed of cooperative nodes. Lower values could be due to occasional network disconnection or to defective nodes occupying strategic position able to block cooperative paths.

Once high cooperation is reached CCP and LCC always have very high values independently from network size, as illustrated in figure 9.

To draw conclusions about SLACER, starting from SLAC and applying a simple modification we obtained a connected topology with interesting properties such as small-world likeliness and high and scalable LCC and CCP. In addition, through drop probability parameter it is possible to tune the network topology (disconnected clusters, small-world, random) and cooperation level according to the application task to be performed.

5.3 Greedy Cheating Liar Nodes

A step further in SLAC and SLACER development consisted in testing system performance in presence of cheating nodes. Cheating behavior should

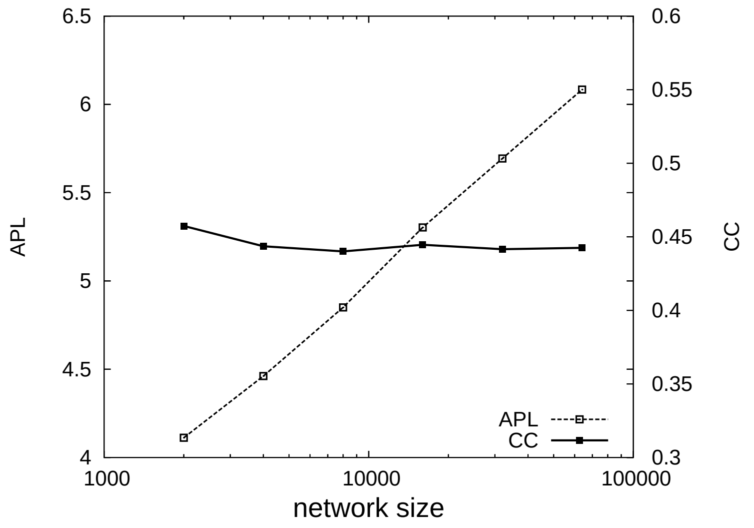


Figure 8: SLACER topology: Clustering coefficient and Average Path Length values for different network sizes. Constant and high clustering vs. logarithmically growing and low average path length give confidence on the small-world likeliness of the network.

not be confused with free-riding. What is meant with free-riding is that nodes selfishly try to maximize their own income not cooperating with other nodes, while for cheating we mean that protocol specification are deliberately violated.

One main assumption made so far is that nodes report honestly their state during reproduction. What would happen if, on the contrary, nodes were able to cheat, reporting false strategy, utility, or neighborhood? To answer this question two different families of cheating nodes have been defined:

- **Network exploiters: Greedy Cheating Liars (GCL)**: GCL nodes' aim is to get the maximum possible outcome from the network by selfishly exploiting their neighbors.

- **Network destroyers: Nihilists (NIH)**: NIH nodes' aim is to destroy the network opposing cooperation formation, spread and maintenance with no care about their own performance.

Taking into account GCL node we try to model the behavior of a rational user who wants to get the maximum possible outcome from the system. In general the best way to do this is to act selfishly while being surrounded by cooperating nodes. Using PD as the application layer it translates into

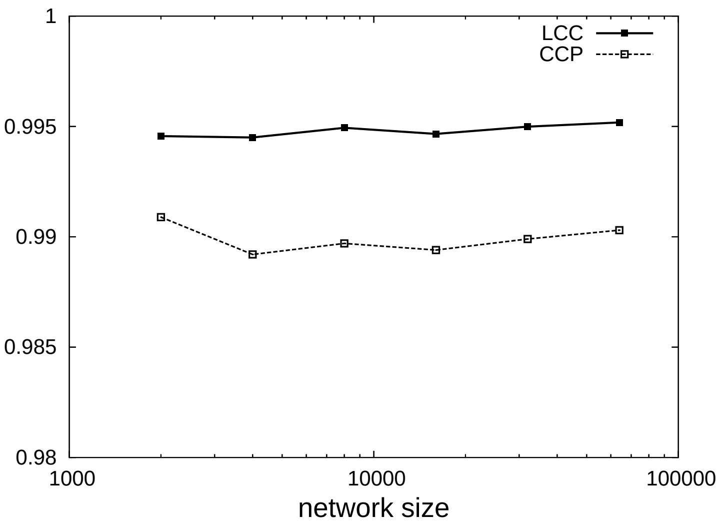


Figure 9: CCP and LCC in SLACER network. Neither of them is influenced by network size, suggesting good SLACER scalability.

always playing D while being surrounded by nodes who are playing C. This kind of circumstances can be easily produced through false state reports.

A GCL node first should always defect, then to force its neighbors to cooperate, it should always report high utility (so to bring other nodes into its neighborhood) and cooperative strategy (so to turn new neighbors to cooperation). Moreover it could also pass a single link to itself so to avoid links between its neighbor and create a local star topology.

Looking at NIH nodes the analysis is similar to GCL ones. To get nodes into its neighborhood a NIH node should always declare high utility (as in the GCL case), it should also declare defective strategy so to turn new neighbors to defection, and defect itself since its only goal is to spread defection. To try spreading defection faster a NIH node could also pass random links in spite of its real neighborhood.

To sum up the two proposed cheating behavior:

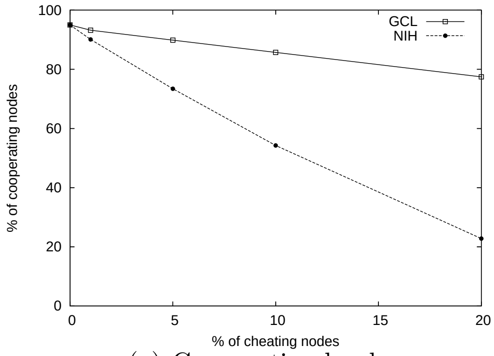
- **GCL**
  - **Utility**: Always declare high utility.
  - **Strategy**: Always declare C while always playing D.
  - **Links**: Pass a single link to itself.
- **NIH**

- **Utility**: Always declare high utility.
- **Strategy**: Always declare and play D.
- **Links**: Pass links to a set of random nodes.

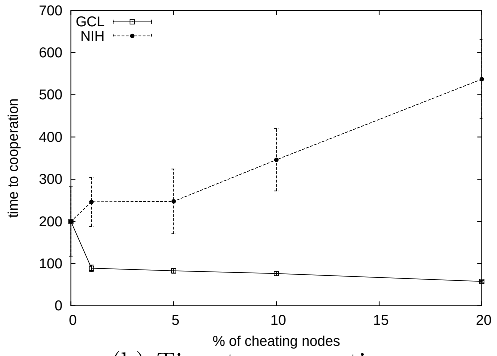
We experimented with different quantities of cheating nodes in a 4000node network and we get very different results from the two kind of cheating behavior, here no results related to false links report are given, they can be found in [3].

Analyzing results in presence of nodes lying about utility and strategy, NIH nodes seem to achieve their goal leading the network to low cooperation level, long time to get it, and very low average nodes' utility as illustrated in figures 10(b) and 11(b).

Turning to GCL nodes, on the contrary, we get very interesting and surprising results: the system seems to someway benefit from the presence of GCL nodes! What happens with GCL nodes is that even if they are able to exploit others, global performance degradation is very graceful, that is GCL nodes exploitation is evenly distributed throughout the network and the utility achieved by non cheating nodes is not much lower than the optimal case given by a fully cooperating network (figure 11(a)). Looking at cooperation level and at time needed for cooperation to spread starting from a fully defective network (figure 10(a)), GCL nodes have positive effects. Cooperation level reaches very high level (basically all non cheating nodes cooperate) while the time needed to reach such high levels of cooperation is about the half than in the case with no cheating nodes.



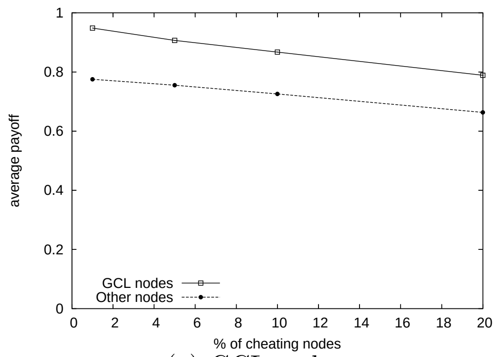
(a) Cooperation level



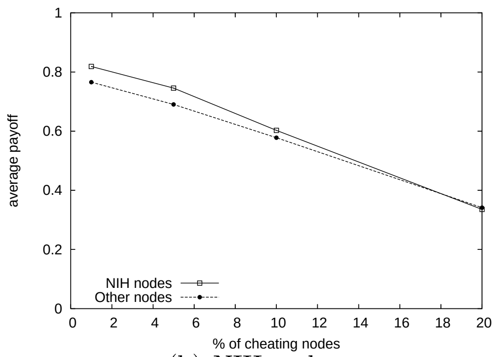
(b) Time to cooperation

Figure 10: Cooperation formation properties in presence of cheating nodes.

GCL nodes effects can be interpreted as the offering of a service (fast and high cooperation) requiring some sort of payment (high utility for cheating



(a) GCL nodes



(b) NIH nodes

Figure 11: Utilities comparisons between cheating and non cheating nodes.

nodes and graceful degradation of non cheating nodes performance). This results suggest a provoking idea: maybe, rather than trying to detect and stop cheating nodes, a way to deal with and to get some form of benefit from them should be found!

Even though we believe this result to be very interesting, we certainly know that it needs further investigation and analysis. In particular SLAC system and its way of dealing with cheating nodes is based on the implicit assumption that cheating behavior would not spread neither in the system the way free-riding could do, nor outside the system, i.e. through hacked clients. This assumption can be argued and involves many different issues not necessarily directly related to SLAC and SLACER algorithms specification. This problem will be discussed deeper in section 6.

6 Open Issues

Our goal is to produce coordination between nodes in a lightweight fashion, hence no reputation system has been introduced to avoid system overhead. To make things work with such premises some implicit assumptions has been made anyway. Such assumptions will now be made explicit and discussed.

6.1 Distinction between Free-Riding and Cheating

In the presented work a main distinction has been made between free-riding and cheating behavior. By free-riding we mean behaving in a selfish way to increase one's own income with no care about possible system level performance degradation. Free-riding then implies the following of protocol specification but the possibility to act selfishly.

For cheating, on the other hand, we mean any kind of behavior that do not respect protocol specification, as in the case of nodes reporting false states during reproduction.

One main distinction between this two different behaviors is in the way they spread throughout the system. Free-riding behavior can spread in our system through reproduction, like in the case when high utility defective nodes bring other nodes to defection. As for cheating behavior, we assume that it does not spread, like in the GCL case, where non cheating nodes just copied declared strategy, but not the ability to lie about their own state.

6.2 In-Protocol vs. Out-Protocol

The reason why we assumed that cheating behavior would not spread requires to abstract from the system itself and to look at possible users' behavior. We refer to everything regarding protocol specification and the way the system works as *In-Protocol*, and to everything regarding "the real world" and the way users behave and interact as *Out-Protocol*.

This classification reflects in some sense the free-riding vs. cheating one. Using a file sharing system example we can see at leeching as an example of free-riding behavior: not to sharing files do not need to subvert protocol specification, and result in a lowered cost (in term of outgoing traffic) for the user, but it could degrade system performance. In the same setting, an example of cheating could be represented by the use of an hacked client to improve performances, hence to exhibit a behavior different from protocol's specification.

We have shown that in SLAC and SLACER even if free-riding can potentially spread throughout the system this does not happen, on the contrary the combination of selfish (but non cheating) nodes' interaction bring the system to high cooperation level.

If cheating behavior were free to propagate as free-riding is, SLAC and SLACER would not work. In fact our main assumption requires that in every case the majority of nodes follows the protocol's rules. In SLAC and SLACER there is no mechanism to prevent cheating behavior from spread as it happens for free-riding, so if cheating nodes were able to spread their behavior, everyone in the network would have eventually ended up cheating!

Our assumption about non spreading cheating behaviors is drawn from the fact that for cheating to spread Off-Protocol capabilities (like hacking a client) and communication (like spread the client) are required, moreover we observe that spreading an hacked client, hence spreading cheating behavior in the system, would reduce cheaters' (and hacker's) benefit itself, hence a rational hacker who could be able to subvert the protocol is disincentivized from spreading its knowledge about cheating behavior.

An example of this fact can be done looking at figure 11(a): in this case few GCL nodes get high utility, but if more than 20% of the nodes cheat what happens is that even if cheaters continue to have better utility than other nodes, their utility is anyway lower than if there were no cheating nodes at all!

We are aware that such assumptions are quite strong and can be argued. In fact further development of the presented work will consist in trying not to depend on them. This can be obtained modifying the presented algorithms adding mechanisms to locally discover cheating nodes and avoid interactions with them, the way such mechanisms should be designed, and if to use novel specific approaches or already existent ones (like shared history) is the main scope of future investigation.

7 Expected Results

We are currently working to find a way to deal with the problem discussed above and to make the system able to detect anomalies. The method we plan to develop is based on network topology analysis and draws inspiration from the study of complex networks from a biological point of view.

7.1 Motif Analysis

Networks are often characterized using average global measures, such as average path length and clustering coefficient. Although valuable, such measures rarely give a picture of the detailed structure of the networks. This means that networks with different topologies can have identical global average measurements, therefore, in order to further understand and classify natural and artificial networks new methods have been proposed.

Recently more sophisticated topological techniques than the classical global measurements have been defined to study complex networks' (both natural and artificial) properties. Among these approaches the analysis of small local pattern (called *motif analysis* [23]) turned out to be a potentially powerful tool when applied to the algorithms presented above.

Motif analysis is performed by breaking the network down into all possible n node subgraph patterns and counting their occurrence in the whole network comparing those counts against randomly generated networks with the same characteristics (number of nodes, and in and out degree links). From such data it is possible to extract information about peculiar local patterns appearing much more often (or seldom) than in the random case. Then n node subgraph patterns significantly more prevalent than in random networks are considered *mofits* of the network. Equally important to define network structure, although less discussed in the literature, n node subgraph patterns that are underrepresented in the network have been termed *anti-motifs* [22].

Obviously, for large subgraph sizes the number of possible motifs becomes large but for smaller sizes (3 and 4 nodes) it is possible to efficiently search for all occurrences even in large networks.

7.1.1 Subgraph Ratio Profile (RSP)

The P2P networks produced by SLAC and SLACER are undirected in the sense that all links are bidirectional. So for the purposes of analysis we search for all undirected four node subgraphs (tetrads). Figure 12 shows the six possible undirected tetrads. In order to analyze the P2P networks we used a *subgraph ratio profile* (SRP) method [22]. This approach is particularly useful for analysis of the P2P networks since traditional motif analysis methods using z-scores are not network size invariant for non-directed tetrads and this makes comparison with networks of different sizes difficult.

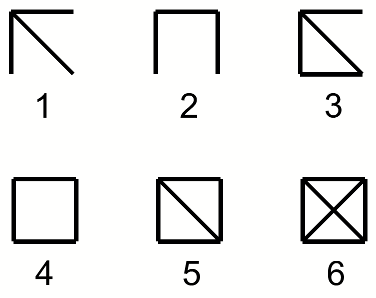


Figure 12: Undirected 4-node motifs.

For a given network N the SRP is a normalized vector of Δ_i values:

$$SRPi = \frac{\Delta_i}{\sqrt{\sum_i \Delta_i^2}} \tag{3}$$

The vector elements, one for each of the six tetrads, are calculated based on the abundance of each tetrad i relative to randomly generated networks, To avoid large values as an artefact of very small occurrences of tetrads in both the real and random networks a constant small value ϵ (=4 in the present measurements) is added to the the denominator:

$$\Delta_i = \frac{Nreal_i - < Nrand_i >}{Nreal_i + < Nrand_i > + \epsilon} \tag{4}$$

A given SRP can be graphed producing a curve which characterizes the tetrad motifs and anti-motifs visually as in figures 13 and 14.

7.2 SLAC and SLACER Networks Motifs

Motif analysis has been defined and is used in biology to study complex networks related to biological systems, such as protein structures and food-webs, were local structure can be interpreted to have very precise meanings and functions. Even though this is not the case in SLAC and SLACER, where it is difficult to argue specific reasons for the formation of specific local patterns, motif analysis, especially the aggregation on a global scale of motifs occurrences turns out to have potential interesting applications.

SLAC and SLACER motif analysis results [30] are illustrated in figures 13 and 14 respectively, these figures represent the SRP for both SLAC and SLACER networks at different stages of their evolution. In each case three time ordered network snapshots are shown.

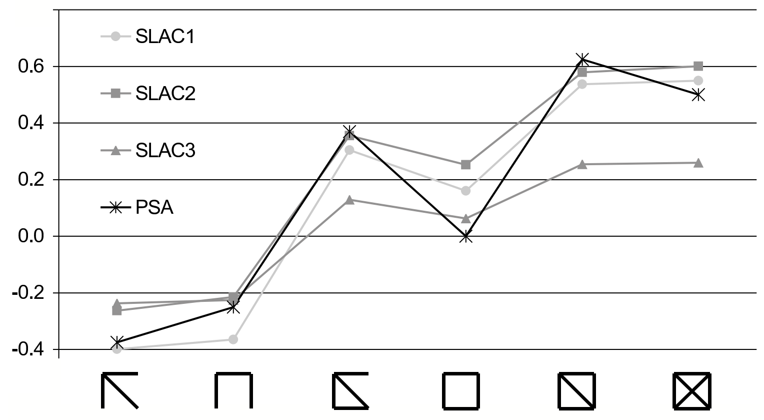


Figure 13: SLAC network RSP at different cooperation evolution stages.

It is interesting to notice that the curves in figures 13 and 14 follow a similar time evolution for both SLAC and SLACER: immediately before cooperation (snapshot 1) the curve already has a very similar shape to the final curve, during the outbreak of cooperation (snapshot 2) the curve tends to move upward slightly (less anti-motifs 1 and 2, but more motifs 3 to 6), then, after stable cooperation is attained (snapshot 3) the curve tends to

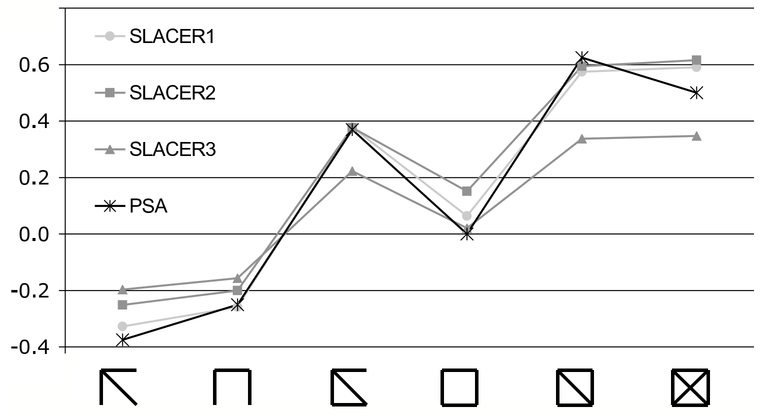


Figure 14: SLACER network RSP at different cooperation evolution stages.

flatten (with all points moving towards the x-axis). Motifs 1 and 2 are under represented (anti-motifs) and motifs 3 to 6 are over represented (motifs) but with a large dip for motif 4 - almost close to zero (identical to the random occurrence) when cooperation has stabilized (snapshot 3). Moreover the SRP curves shown in figures 13 and 14 can be directly compared to protein structure (PSA), taken from [23].

Another interesting result is that the different snapshots show significant differences in the SRP, such that for a given network's SRP it might be possible to predict if it was in a stable cooperative mode or not, just taking into account topological properties and ignoring nodes' state. This could possibly be very useful, since it indicates a way to detect if the network is working cooperatively or which stage in the evolution to cooperation it has reached, as well as detecting unexpected undesired behaviors, based purely on structural characteristics.

7.3 Distributing Motif Analysis

As argued before, motif analysis can be used as a tool for detecting global network state from local characteristics. The main problem with this approach is that so far motif analysis techniques and algorithm are not designed to be distributed: network analysis is usually performed on a single local machine requiring a graph representation of the whole network in input. This is obviously unsuitable in P2P environment, where everything should be performed locally in a decentralized fashion and with no global knowledge.

What we want to obtain is a decentralized technique for motif analysis that can be directly applied and used in P2P networks. The schema we want to follow to achieve this is to implement a mechanism through which nodes will be able to detect the local motifs they are part of (i.e. through views exchange). Once local motifs are evaluated they should be aggregated so to make the whole network aware of the current motifs quantities and distributions.

The fact that aggregation mechanisms for P2P networks have already been successfully defined [18] improves our confidence in the feasibility of a distributed motif analysis algorithm for P2P networks.

8 Conclusions

The lack of central control makes cooperation between nodes in a P2P networks a fundamental and hard to achieve task. Moreover P2P systems implicitly define many restrictions as the impossibility to manage the system from a global point of view and the necessity to manage everything on a local interactions basis.

An approach to produce cooperation and coordination between P2P nodes in order to achieve high global performances has been proposed. We draw inspiration mainly from computational sociology and the way social networks form and evolve. From this source of inspiration we inherited important novel features with respect to previously proposed approaches. In our model there is no need for global information (as global shared history) or for a structured network (as DHT), moreover our model is able to build an *artificial social network* whose topology can be successfully used to produce and maintain cooperation, with no need to import any other kind of social network into the system.

Even though the presented approach seems to be promising it is still in need for many further improvement. The main improvement to be done is about system reliability against cheating (other than free-riding) nodes. A first step in this direction can be done looking at motif analysis tool and trying to design a distributed motif analysis technique.

Further improvements could involve the definition of more realistic applications to test SLACER and the utilization of effective network environments once a satisfying simulative prototype will have been finally designed.

References

[1] Adar, E., and Huberman, B. Free riding on gnutella, 2000.

[2] Aiyer, A. S., Alvisi, L., Clement, A., Dahlin, M., Martin, J.-P., and Porth, C. Bar fault tolerance for cooperative services. In *Proceedings of the 20th ACM Symposium on Operating System Principles (SOSP)* (October 2005).

[3] Arteconi, S., and Hales, D. Greedy cheating liars and the fools who believe them. Tech. Rep. UBLCS-2005-21, University of Bologna, Dept. of Computer Science, Bologna, Italy, Dec. 2005. Also available at: `http://www.cs.unibo.it/pub/TR/UBLCS/2005/2005-21.pdf`.

[4] Axelrod, R. *The Evolution of Cooperation.* Basic Books, New York, 1984.

[5] Castro, M., and Liskov, B. Practical byzantine fault tolerance. In *OSDI: Symposium on Operating Systems Design and Implementation* (1999), USENIX Association, Co-sponsored by IEEE TCOS and ACM SIGOPS.

[6] Cox, L., and Noble, B. Samsara: Honor among thieves in peer-to-peer storage, 2003.

[7] Cox, L. P., Murray, C. D., and Noble, B. D. Pastiche: making backup cheap and easy. *SIGOPS Oper. Syst. Rev. 36*, SI (2002), 285–298.

[8] Douceur, J. The sybil attack, 2002.

[9] Feldman, M., Lai, K., Stoica, I., and Chuang, J. Robust incentive techniques for peer-to-peer networks. In *EC '04: Proceedings of the 5th ACM conference on Electronic commerce* (New York, NY, USA, 2004), ACM Press, pp. 102–111.

[10] Friedman, E., and Resnick, P. The social cost of cheap pseudonyms, 1998.

[11] Hales, D. Choose your tribe! - evolution at the next level in a peer-to-peer network. Tech. Rep. UBLCS-2005-13, University of Bologna, Dept. of Computer Science, May 2005. Also available at: `http://www.cs.unibo.it/pub/TR/UBLCS/2005/2005-13.pdf`.

[12] Hales, D., and Arteconi, S. Slacer: A self-organizing protocol for coordination in p2p networks. *IEEE Intelligent Systems* (To appear).

[13] Hales, D., and Edmonds, B. Applying a socially-inspired technique (tags) to improve cooperation in p2p networks. *IEEE Transactions in Systems, Man and Cybernetics - Part A: Systems and Humans 35(3)* (2005), 385–395.

[14] Hardin, G. The tragedy of the commons. *Science 162*, 3859 (December 1968), 1243–1248.

[15] Holland, J. The effect of lables (tags) on social interactions., 1993.

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

[17] Jelasity, M., Montresor, A., and Babaoglu, O. A modular paradigm for building self-organizing peer-to-peer applications. In *Engineering Self-Organising Systems* (2004), G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, and F. Zambonelli, Eds., no. 2977 in Lecture Notes in Artificial Intelligence, Springer, pp. 265–282.

[18] Jelasity, M., Montresor, A., and Babaoglu, O. Gossip-based aggregation in large dynamic networks. *ACM Trans. Comput. Syst. 23*, 1 (2005), 219–252.

[19] Kamvar, S. D., Schlosser, M. T., and Garcia-Molina, H. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *12th International World Wide Web Conference* (Budapest, Hungary, 20-24 May 2003).

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

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

[22] Milo, R., Itzkovitz, S., Kashtan, N., Levitt, R., Shen-Orr, S., Ayzenshtat, I., Sheffer, M., and Alon, U. Superfamilies of evolved and designed networks. *Science 303*, 5663 (March 2004), 1538–1542.

[23] Milo, R., Shen-Orr, S., Itzkovitz, S., Kashtan, N., Chklovskii, D., and Alon, U. Network motifs: simple building blocks of complex networks. *Science 298*, 5594 (October 2002), 824–827.

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

[25] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Schenker, S. A scalable content-addressable network. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications* (October 2001), vol. 31, ACM Press, pp. 161–172.

[26] Rowstron, A., and Druschel, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)* (Nov. 2001), pp. 329–350.

[27] Saroiu, S., Gummadi, P., and Gribble, S. A measurement study of peer-to-peer file sharing systems, 2002.

[28] Stoica, I., Morris, R., Karger, D., Kaashoek, F. F., and Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM Comput. Commun. Rev. 31*, 4 (October 2001), 149–160.

[29] Travers, J., and Milgram, S. An experimental study of the small world problem. *Sociometry 32*, 4 (1969), 425–443.

[30] Valverde, S., Sole, R. V., Hales, D., Babaoglu, O., Arteconi, S., and Canright, G. Application of motif analysis to artificial evolving networks. d 5.4.1, Dec 2005. DELIS project deliverable. Also available at: `http://cfpm.org/~david/delis/2005/d5.4.1/D5.4.1.pdf`. Delis project homepage: `http://delis.upb.de/`.

[31] Watts, D. J., and Strogatz, S. H. Collective dynamics of 'small-world' networks. *Nature 393*, 6684 (June 1998), 440–442.