Applying a socially-inspired technique (tags) to improve cooperation in P2P Networks

David Hales and Bruce Edmonds

***Abstract*— Here we focus on the problem of maintaining significant levels of cooperation in peer-to-peer networks of selfish adaptive peers. We propose a simple algorithm that maintains high levels of cooperation in such a network whilst performing the collective task of file sharing. The algorithm is adapted from novel "tag" models of cooperation that do not rely on explicit reciprocity, reputation or trust mechanisms.**

A sequence of three simulation models is presented - starting with an abstract model of tag-based cooperation (TagWorld) and finishing with a peer-to-peer file-sharing model (FileWorld) that puts the technique to work.

From analysis of extensive computer simulations, we demonstrate the technique to be scalable, robust and decentralized – requiring no central servers or authorities. The algorithm is relatively simple - peers do not need to store additional trust information about other nodes or to perform significant additional processing.

***Index Terms*— Commons tragedy, Networks, Peer-to-Peer systems, Self-Organization, Tags.**

I. Introduction

Computational simulations of social phenomena are a fertile source of ideas for computational systems in general – especially where those systems are not under central control, but composed of many autonomous (or simply unknown) parts. One such phenomena is the "Tragedy of the Commons" [1] where, although everybody may greatly benefit from a common resource, it is destroyed through the selfish actions of each person. These sorts of situation are well studied in the social sciences since they occur in many situations in naturally occurring social systems (e.g. over-grazing on a common plot of land or polluting the environment). Starting from [2] there have now been many computational models to explore ways out of these dilemmas. In this paper we explore how one particular solution to such "tragedies" may be applied to solve a similar situation in peer-to-peer (P2P) systems. This involves an algorithm that employs the recognition of arbitrary externally detectable attributes, which are called "tags".

Recently a number of novel cooperation models based on this "tag" mechanism have been presented [3]-[6]. They were designed to explore theories of cooperation and coordination in the social sciences. We show how they can be applied to produce efficient P2P networks.

We migrate the "tag" mechanism from a social science oriented model to a practically inspired P2P file-sharing model. In order to achieve this migration we present a sequence of three models moving from the initially abstract model to the final applied simulation model.

The initial model, TagWorld (section III), demonstrates the emergence of cooperation in a population of evolving agents that interact randomly in pairs to play a cooperation game called the Prisoner's Dilemma (PD) game. The second model, NetWorld (section IV), adapts the TagWorld by situating interactions between neighbors on a dynamic graph topologically similar to a P2P network. In the final model, FileWorld (section V), we maintain the graph topology but change the task from the PD game to file sharing.

As we adapt the models, we check that we preserve the desirable scalability and robustness properties of the initial model while still emerging the required level of cooperation.

After presenting the models and simulation results, we compare related work (section VI) and conclude with a brief discussion (section VII).

Initially, in the next section, we introduce the motivating problem, how to control selfishness in P2P systems, and our general method – drawing on existing social simulation models.

II. Social Mechanisms for P2P systems

Social Scientists have always been interested in complex, self-organizing, emergent systems because their target is naturally occurring societies that display these properties.

Recently a wealth of, agent-based, computer models of social phenomena have been advanced [7]-[11]. Such models specify individual rules followed by multiple agents and the consequent emergent results when they are executed in a shared environment. Results often demonstrate properties that would appear desirable in engineered large-scale distributed systems.

Here then, it would seem, is a developing body of work that can be used by engineers of self-organizing systems. However, sociologically inspired models are realized at a high level of abstraction - very different from the application task domains in which engineers are interested. How can the results from such models be used for the solution of practical

Manuscript received June 1, 2004. This work partially supported by the EU within the 6th Framework Program under contract 001907 (DELIS).

David Hales is with the Computer Science Department, University of Bologna, 40127 Bologna, Italy (phone: +39 051 2094981; fax: +39 051 2094 510; e-mail: dave@davidhales.com).

Bruce Edmonds is with the Centre for Policy Modelling, Manchester Metropolitan University, Manchester, UK (e-mail: bruce@cfpm.org).

engineering problems?

The approach we use here is to move, via a succession of models, from an initial abstract model to the actual application by introducing piecemeal refinements, while preserving the, emergent, phenomena of interest [12].

In this article, we take a sociologically inspired model that produces high cooperation between selfish entities and develop two further models that move closer to an application domain, P2P file sharing, in which high cooperation is desirable. Our initial results are extremely encouraging indicating that novel and useful techniques *can* be imported from sociologically inspired models.

A. The 'tragedy of the commons' in P2P networks

P2P file sharing systems using unstructured overlay type networks have become very popular over recent years [13]-[15]. At a given time millions of users (peers) may be running, connected over the Internet. Each peer (or node) connects to some set of known peers – forming a graph structure. These applications are self-organizing in that peer software is provided freely for downloading with users deciding when to download and execute these peers on their hardware. There is no centralized administration or control and such systems are open because there is nothing stopping peers from modifying their software (and hence behavior). In addition, users can choose to share nothing or only poor quality files. Given these latter considerations and empirical evidence [13] it can be argued that a major problem in such applications is to develop mechanisms that discourage selfish behavior, where peers download files without uploading them, and encourage altruistic behavior (sharing high quality files). These kinds of situations are precisely analogous to "commons tragedies" [1], since all individuals benefit if all act altruistically but each has an incentive to act selfishly.

In P2P systems, this problem is evident in many applications, not just file sharing – e.g. the sharing of processing power or storage, the passing of messages and performing remote operations. Hence, techniques that can address the "commons tragedy" would appear to have wide applications within P2P systems.

Existing models from the social and biological sciences suggest many possible candidate mechanisms for addressing this problem, including, reciprocal altruism [2], [16] group reputation [17] and interaction on fixed topological structures [18]. In this article, we focus on a recent novel "tag" model that solves an abstract form of a commons tragedy without requiring complex algorithms, reputational knowledge or fixed interaction topologies.

We start with the abstract model, TagWorld (section III), demonstrating how the tag mechanism produces cooperation and then progressively modify it and produce two further models: NetWorld (section IV) and FileWorld (section V). NetWorld situates interaction on a P2P type network topology. FileWorld changes the task domain from an abstract game to a simulated P2P file-sharing task.

III. The TagWorld Model – How tags work

Here we introduce the TagWorld model [3] that demonstrates a mechanism for promoting cooperation between simple adaptive entities acting in a self-interested way. The simulations show that self-interested agents playing the game, normally known as, the Prisoner Dilemma (PD), that usually leads to non-cooperation, can instead be made to cooperate by the use of tags. The results from the TagWorld experiments serve as the foundation upon which further experiments are performed - firstly, to a networked, and then, to a P2P scenario, in which the tag technique is used as a mechanism to incentivize cooperation.

In the remainder of this section, we introduce the PD game, then the "tag" concept, then the model and results.

A. The Prisoner's Dilemma

The Prisoner's Dilemma (PD) game captures a situation in which there is a contradiction between collective and self-interest. It can be considered as a minimal and abstracted form of a "commons tragedy". Two players interact by selecting one of two choices: to "cooperate" (C) or "defect" (D). For the four possible outcomes of the game, players receive specified payoffs. Both players receive a reward payoff (R) and a punishment payoff (P) for cooperation and mutual defection respectively. However, when individuals select different moves, differential payoffs of temptation (T) and sucker (S) are awarded to the defector and the cooperator respectively. Assuming that neither player can know in advance which move the other will make and wishes to maximize her own payoff, the dilemma is evident in the ranking of payoffs: $T > R > P > S$ and the constraint that $2R > T + S$. Although both players would prefer T, only one can attain it. No player wants S. No matter what the other player does, by selecting a D move a player ensures she gets either a better or equal payoff to her partner. In this sense a D move can't be bettered since playing D ensures that the defector cannot be suckered. This is the so-called "Nash" equilibrium for the single round game – hence an ideally rational selfish player would always choose D. It is also an evolutionary stable strategy (ESS) for a population of randomly paired players, where reproduction fitness is proportional to payoff – hence evolution also selects D [19].

Therefore, the dilemma is that if both individuals selected a cooperative move they would both be better off but both evolutionary pressure and ideal "rationality" result in defection.

B. Tags

Tags are markings or social cues that are attached to individuals (agents) and are observable by others [20]. For example, people may be able to tell whether another is of the same social group as them by observing the style of clothes of the other – such subtle external indicators, where they have no other function, are their tags. These tags are often represented in computational models by a single number or a bit string; they evolve like any other trait in a given evolutionary model. The key point is that the tags have no direct behavioral implication for the agents that carry them. But through

indirect effects, such as the restriction of interaction to those with the same tag value, they can evolve from initially random values into complex ever changing patterns that serve to structure interactions. A number of tag models have been applied to social dilemma type scenarios [3], [5], [6] but only the TagWorld model appears to produce cooperation in the single-round PD game (i.e. interactions with strangers) and we envisage this would be something that would be beneficial in our target P2P application.

C. Description of TagWorld

In TagWorld agents play the PD in pairs. The model is composed of very simple agents. Each agent is represented by a small string of bits. On-going interaction involves pairs of randomly selected agents, *with matching tags*, playing a single round of PD. Agent bits are initialized uniformly at random. One bit is designated as the PD strategy bit: agents possessing a "1" bit play C but those possessing a "0" bit play D. The other (L) bits represent the agents' tag – a binary string. Tag bits do not affect the PD strategy played by the agent but they are observable by all other agents. Fig. 1 gives an outline of the simulation algorithm used.

```
LOOP some number of generations
    LOOP for each agent (a) in the population
        Select a game partner agent (b) with same tag (if possible)
        Agents a and b invoke their strategies and get payoffs
    END LOOP
    Reproduce agents in proportion to their average payoff
    Apply mutation to tag and strategy of each reproduced agent
      with low probability
END LOOP
```

Fig. 1. Pseudo-code algorithm for main loop of the model.

Each agent is selected in turn to play a single-round of PD. Agents do not selected an opponent randomly but selectively based on the tag string. The opponent is selected randomly from the subset of the population sharing the same tag string as the agent. If this subset is empty, because no other agents have an identical tag, the agent plays against some randomly chosen partner from the entire population – whatever their tag values.

After each pair of agents plays a game of PD the payoffs are accumulated against each agent. When all agents have been selected in turn, and played a game, agents are *reproduced* probabilistically in proportion to the average payoff they received (using a "roulette wheel" selection algorithm). With a small probability, each bit of each reproduced agent is mutated (i.e. flipped).

The TagWorld has no topological structure since agents are not situated in a space – such as a lattice or a ring – interaction is only structured using tag similarity and random selection.

D. Results from TagWorld

Extensive experimentation varying a number of parameters showed that for a large enough number of tag bits; high levels of cooperation quickly predominate in the population. High cooperation is obtain when there is at least $L \geq 32$ tag bits (per agent) for a population of 100 agents with a mutation rate of $m = 0.001$ and PD payoffs of $T = 1.9$, $R = 1$, $P = S = 0.0001$. P and S are set to the same small value for simplicity. If a small value is added to P (enforcing $T > R > P > S$) results are not significantly changed. If tags are removed from the model making game pairing completely random then the population quickly goes to complete defection - the Nash equilibrium for the single-round PD.

More interesting still, if all the agents are initially set to select action D (as opposed to randomly set) then the time required to achieve a system where C actions predominate is found to monotonically decrease as population size increases. This is an inverse scaling phenomena: the more agents, the better. Additionally, the fact that the system can recover from a state of total D actions to almost total C actions, under conditions of constant mutation, demonstrates robustness to noise.

The TagWorld produces an efficient, scalable and robust solution – based on very simple individual learning methods – here modeled as reproduction with mutation. How do tags produce this result? We discuss this in the next section.

E. How Tags Work

The key to understanding the tag process is to realize that agents with identical tags can be seen as forming an "interaction group" or "tribe". The population can be considered as partitioned into a set of such groups. If a group happens to be entirely composed of agents selecting action C (a cooperative group) then the agents within the group will outperform agents in a group composed entirely of agents selecting action D (a selfish group). This means that individuals in cooperative groups will tend to reproduce more than agents in selfish groups because they will obtain higher average payoffs. If an agent happens to select action D within a cooperative group then it will individually outperform any C acting agent in that group and, initially at least, any other C acting agent in the population – remember the T payoff is 1.9 but the best a C acting agent can do is $R = 1$.

However, due to its high payoff such a D acting agent will tend to reproduce many copies of itself and then the group to which it belongs becomes very quickly dominated by the newly reproduced D acting agents. The group then becomes a selfish group and the relative advantage of the lone D acting agent is lost – *the group snuffs itself out due to the interaction being kept within the group*. So by selecting the D action an agent destroys its group very quickly (remember groups are agents all sharing an identical tag). Fig. 2 visualizes this group process in a typical single run. Each line on the vertical axis represents a unique tag string (i.e. a possible group). Groups composed of all C action agents are shown in light gray, mixed groups of C and D agents are dark gray and groups composed of all D are black.

F. TagWorld Discussion

The results indicate that by partitioning the population into distinct "interaction groups" (or "tribes") tags facilitate a kind of "group selective" process in which defection is kept in check. However, from these results, it is unclear how

important the binary string form of the tags is – does the large string (remember $L \geq 32$ bits for high cooperation) impose some kind of necessary topological structure? Previous experiments with the model in which the tag was replaced by a single value (a real or integer number) produced low cooperation so it appeared that the bit string structure was important. However, after further experimentation, we discovered that the significant factor was differential mutation between tag and strategy. An artifact of a long tag string is that, given mutation is applied with equal probability to each bit; the mutation rate applied to the tag *as a whole* relative to the PD strategy is significantly higher. Mutation applied to the tag as a whole is $1-(1-m)^L \approx 0.0315$ (where $m = 0.001$ and $L = 32$). This means that the effective mutation rate on the tag is well over one order of magnitude higher than on the strategy.

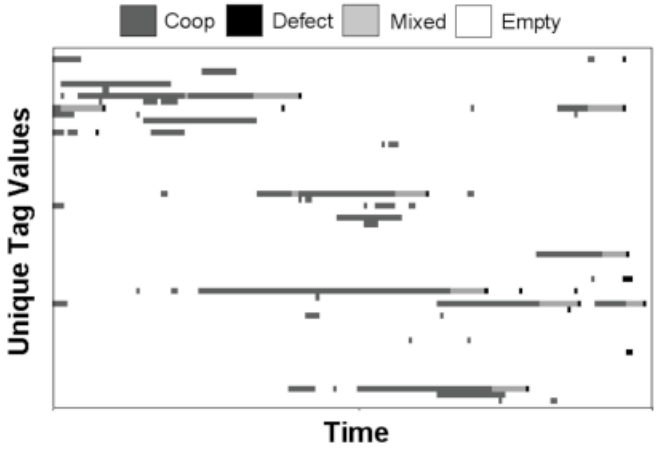

Fig. 2. Visualization of 200 cycles (generations) from a single simulation run showing cooperative groups coming into and going out of existence. See the text for an explanation.

In order to test if differential mutation alone produced the high cooperation we re-implemented the TagWorld model but replaced the binary tags with a single real number, allowing tags to take values between [0..1], and increased mutation on the tag by a factor (f) relative to the strategy bit. We experimented with different values for f and found that indeed a high f produced high cooperation whereas a low f gives low cooperation. Typical results are shown in Fig. 3. Notice that cooperation comes in three phases – a low phase with $f < 4$, a high phase with $f > 6$, and an uncertain phase between the two, where the populations can go into either high or low cooperative regimes depending on initial conditions and on-going stochasticities (i.e. different numbers from the pseudo random number generator in each run).

Minimally these results indicate that there is nothing special about binary strings of tags – rather it is differential mutation that is important. The tag needs to change faster than the strategy. Intuitively this makes some sense, the mechanism of cooperation is driven by cooperative groups forming more quickly than defectors can invade and destroy them. With high mutation on the tag, a cooperative group, as it grows, will tend to spawn more new cooperative groups. A group containing defectors does not grow and quickly deflates and dies (as discussed above).

We wish to move these desirable cooperative properties into a scenario that is closer to our target P2P file-sharing application. We begin this process by implementing a new model in which we transplant the PD interactions onto a graph or network topology as a first step towards our P2P application. Will cooperation still be produced if interactions occur on a network?

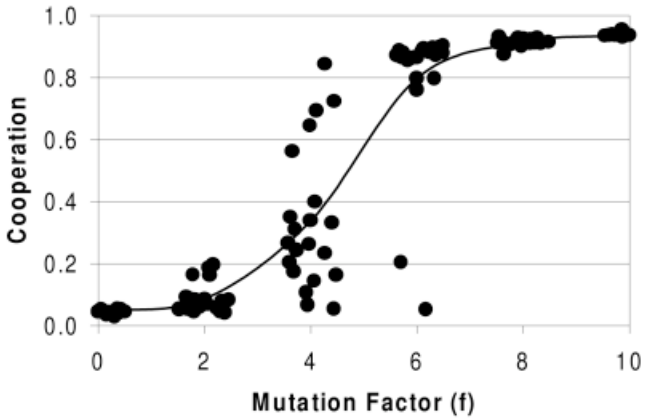

Fig. 3. Results from a number of simulation runs. For each run the mutation factor (f) is plotted against the level of cooperation. There are 20 runs per f value: 0, 2, 4, 6, 8, and 10. Slight noise has been added to the x-axis for readability. The solid line is a smooth curve that passes through the average cooperation value at each f value.

IV. The NetWorld Model - from tags to networks

We now consider how to translate the cooperation producing tag mechanism from TagWorld, where there is no topological structure, into a network topology. We represent the network as an undirected graph in which each vertex represents a node and each edge represents a link between nodes. We assume each node has a fixed capacity (a maximum degree) of links defining its neighbors (a neighbor list).

The underlying mechanism driving cooperation within the TagWorld is the formation and dissolution of sharply delineated groups of agents (identified by sharing the same tag). Each agent could locate group members from the entire population. Each member of the group had an equiprobable chance of interacting with any agent in the population sharing the same tag. In this sense each agent could determine which agents were in their group and always selected an in-group agent to play PD with if this was possible.

If we consider a sparse P2P network in which each node (or peer) knows of some small number of other nodes (neighbors) and those neighborhoods are highly interconnected (clustered) such that most neighbors share a large proportion of other neighbors then we have something similar to the tag-like groupings (or tribes) in TagWorld. Instead of a tag (an observable marker) we have an explicit list of neighbors. In a highly clustered network the same list will be shared by most of the neighborhood. In this sense, one can visualize the table of known peers (the neighbor list) stored in each node as something similar to a tag. It is shared by the group and is the key by which the group can directly interact with each other. To this extent it defines a group boundary. A nice feature of this way of defining group boundaries is that it

offers a watertight method of isolating nodes into their neighborhoods since nodes cannot directly interact with other nodes that they do not know of (i.e. that are not in their neighbor list).

Initially we will consider only direct interactions between neighbors in our network model. If cooperation can be established between the majority of neighborhoods in a network then it follows that any pair of nodes in the network that are indirectly connected will have a good chance of being able to find a path of cooperation through the network. In order to capture this kind of neighborhood interaction in the simplest possible way we have each node in the network play a single round of PD (see above) with a randomly chosen neighbor. No information is stored or communicated about past interactions and the topology is not fixed (see below).

A. The Evolutionary Rewiring Algorithm (ERA)

In the TagWorld model change was produced over time by mutation and differential reproduction based on average payoff. How can these be translated into the network? The interpretations placed on previous tag models have been biological or cultural [3], [20]. But in NetWorld we do not view nodes as "reproducing" in a biological or cultural sense. We translate this process into the network by allowing nodes to "move" or "rewire" within the network. It is consistent with our initial assumptions that nodes may relocate to a new neighborhood in which a node is performing better. That is, we assume that periodically nodes make a comparison of their performance against another node randomly chosen from the network. Suppose node *i* compares itself to *j*. If *j* has a higher average payoff then *i* erases its neighbor list, and strategy, and copies the strategy and neighbor list of *j* also adding *j* into its list. This process of copying can be visualized as movement of the node into a new neighborhood that appears more desirable.

Mutation in the TagWord model was applied after reproduction. Each bit of the tag and the strategy was mutated (flipped) with low probability. Since we are using the same one bit strategy we can apply mutation to the strategy in the same way. We therefore flip the strategy bit of a node with low probability (*m*) immediately after "reproduction" (the movement to a new neighborhood as described above). Since we treat the list of neighbors in each node as the "tag", a mutation operation implies changing the list in some way. However, we can't simply randomly change the list; we need to change the list in such a way as to produce an effect that closely follows the functional effect when mutation is applied in the TagWord model. In that model, tag mutation tended to give agents unique tags – i.e. tags not shared by other agents at that time. However, agents could interact with a randomly chosen agent with non-matching tags if none existed with identical tags. In this way tag mutation lead to the founding of new tag groups. In the network model we don't want to isolate the node completely from the network otherwise it will not be able to interact at all. Neither do we want to move into an existing neighborhood (as with reproduction) but rather to do something that may initiate the founding of a new neighborhood. So we pragmatically express tag mutation as the replacement of the existing neighbor list with a single neighbor drawn at random from the network.

We now have analogues of reproduction and mutation for the network model. Reproduction involves the nodes copying the neighbor lists and strategies of others obtaining higher average scores. Mutation involves flipping the strategy with low probability and replacing the neighbor list with a single randomly chosen node with a low probability – see Fig. 4.

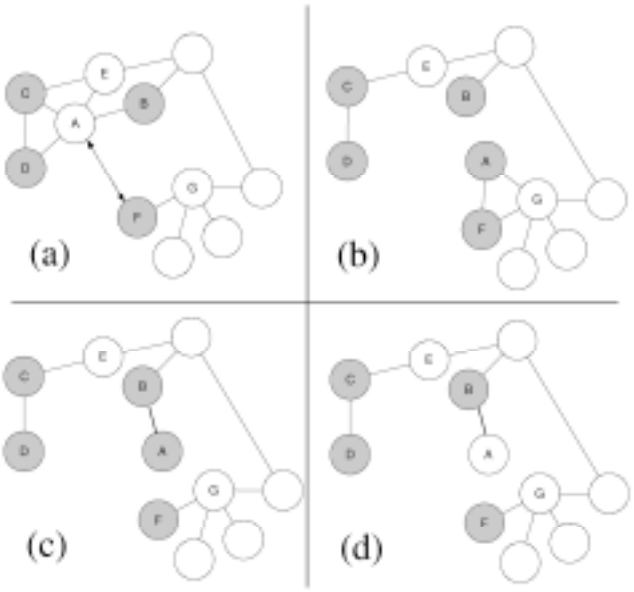

Fig. 4. An illustration of "replication" and "mutation" as applied in the Evolutionary Rewiring Algorithm (ERA). Shading of nodes represents strategy. In (a) the arrowed link represents a comparison of utility between A and F. Assuming F has higher utility then (b) shows the state of the network after A copies F's links and strategy and links to F. A possible result of applying mutation to A's links is shown in (c) and the strategy is mutated in (d).

Perhaps the biggest shift we have made here is in the group-level topology. In the TagWorld model groups had distinct boundaries (those sharing the same tag) and interactions was strictly within groups. Here, groups may overlap since neighboring peers will not necessarily share the same neighbors. In this sense the group boundaries are no longer absolute. This change could be significant and may destroy the cooperation process. For example, a previous tag model of "altruism" [5], [21] with overlapping and changeable group boundaries is not able to solve a social dilemma such as the PD [22].

In the next section, we describe the NetWorld model and then give results of simulation experiments. We find that although significant properties of the cooperation process are changed; scalable, robust and high cooperation is still produced.

B. Description of NetWord

The NetWorld model is composed of a set *N* of nodes (or peers). Each node stores a list of other nodes it knows about (we term this the neighbor list). The entries are symmetrical between neighbors (i.e. if node *i* has an entry for node *j* in its list then node *j* will have node *i* in its list). Hence the nodes

and links in NetWorld form an undirected graph.

In addition to the neighbor list each node stores a single strategy bit indicating if it is to cooperate or defect in a single round game of the PD. Neither the strategy bit nor the list is normally visible to other nodes. Initially, nodes are allocated a small number of neighbors randomly from the population. Periodically each node selects a neighbor at random from its list and plays a game of PD with it. Each node plays the strategy indicated by its strategy bit. After a game the relevant payoffs are distributed to each agent. Periodically pairs of randomly chosen nodes (*i, j*) compare average payoffs. If one node has a lower payoff then the strategy and neighbor list from the other node is copied (effectively moving the lower scoring node to a new neighborhood). Mutation is applied to the strategy with probability *m* and to the neighbor list with probability *mf*. In all cases given here $f = 10$. Therefore, the mutation on the neighbor table is one order of magnitude higher than on the strategy. We take this from our analysis of relative mutation rates in the previous TagWorld model. Mutation of the strategy involves flipping the bit. Mutation of the neighbor list involves clearing the list and replacing it with a single randomly chosen node from the population. Fig. 5 gives an outline pseudo-code algorithm of the NetWorld simulation main loop.

```
LOOP some number of generations
    LOOP for each node (i) in the population of size N
        Select a game partner node (j) randomly from neighbor list
        Node i and j invoke their strategies and get payoffs
    END LOOP
    Select N/2 random pairs of agents (i, j)
    Copy neighbor list and strategy of higher scoring node to lower
      scoring node
    Apply mutation to neighbor list and strategy of each copied
      node with probability m
END LOOP
```

Fig. 5. Outline pseudo-code algorithm of the NetWorld simulation main loop

The neighbor lists are limited in size to a small number of entries. If a link is made to a node that has a full neighbor list then it discards a randomly chosen neighbor link in order to make space for the new link. If a node is found to have no neighbors when attempting to play a game of PD (this can happen if all neighbors have moved away) then a randomly chosen node is linked to and made a neighbor.

C. Results from NetWord

Fig. 6 and Fig. 7 give some typical results up to $N = 2 \times 10^4$. Similar results were obtained up to $N = 10^6$. In these experiments the mutation rate was $m = 0.001$ and the PD payoffs were as previously described in the TagWorld model. Notice that although there appears to be no scaling cost, with convergence to high cooperation taking approximately the same number of cycles for different *N*, we have lost the reverse scaling cost property observed in the TagWorld model. Interestingly, when the mutation factor (*f*) was decreased to 1, making mutation rates equal for links and strategy, high cooperation still occurred but the time to reach it did not scale. We found, what appeared to be, an exponentially increasing non-linear upper bound scaling cost (in time). This meant that some runs were not converging after thousands of cycles for large *N* (>10,000). We still, therefore, need the high mutation rate on the links to produce scalable results but the relationship of the mutation factor (*f*) to the results has changed. If we could understand fully why these scaling costs have changed over the TagWorld model then a deeper understanding of the mechanisms related to scaling costs could be gained. Currently however, this requires further analysis and we leave this for future work.

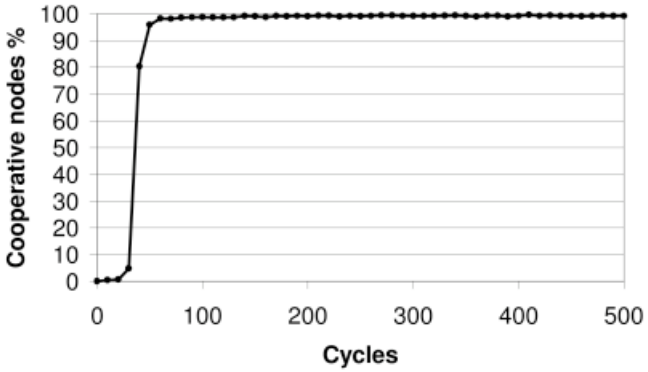

Fig. 6. Shows a typical NetWorld time series ($N = 10{,}000$ nodes) giving the % of cooperative nodes at each cycle.

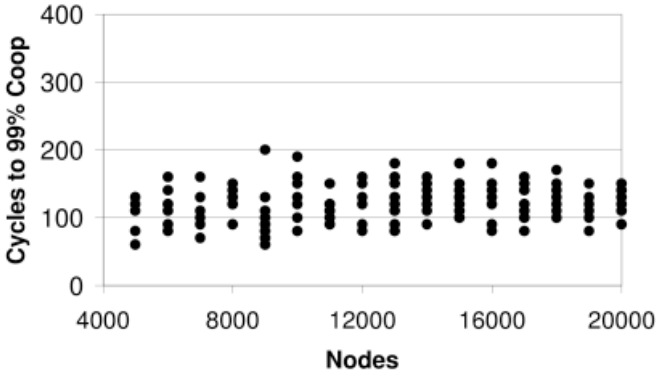

Fig. 7. Shows the number of cycles before 99% of nodes are cooperative for various values of N in NetWorld. Each dot is an individual run.

D. NetWorld discussion

The PD task, used in NetWorld, although capturing a minimal form of a commons tragedy, is rather abstract. The behaviors and coordination required is trivial - although the dilemma itself is *not* trivial. In order to test if the results obtained so far can be carried over into a less abstract task domain we apply the same basic NetWorld algorithm in a new model, FileWorld, which implements a P2P file-sharing scenario.

In the next section we present the model and results. We find that we can indeed use the basic rewiring technique to produce non-selfish file sharing at the network level even when nodes act selfishly at the individual level. We also note a number of important issues that moving from the PD game to something more realistic raises.

V. The FileWorld Model – From PD to File Sharing

In this section we apply the *ERA* algorithm from NetWorld and test it within a more realistic simulation of a P2P network. Thus we outline the results of applying the evolutionary node-rewiring algorithm used in NetWorld in a simulated P2P file-sharing scenario [24], [29]. It models a flood-fill query process where nodes periodically generate and forward queries to their neighbors. Neighbors either process the queries, by producing a "hit" or forwarding to all their neighbors, or ignore them - depending on their capacities and internal state variables.

A. The simulated P2P file-sharing scenario

The idea behind this scenario is that there is a P2P network where nodes are linked to a limited number of the other nodes (their neighbors). Nodes can generate queries (requests for a certain file) that are then sent to neighboring nodes (if the receiving node has spare capacity to do so), which then send it to their neighbors etc. up to limited number of 'hops'. If a node receiving the query has the file asked for, and has spare capacity, it sends the file to the node that generated the query.

This is simulated with a network of N nodes, each of which has a list of its links (all links are bidirectional). Each node, i, has an answering power Ai, and a questioning power Pi. These numbers represent the capability of a node to answer queries and generate queries. $Ai + Pi = Ci$, the capacity of the node. The measurement of satisfaction of the node is a utility: Ui. The idea is that nodes will adapt the balance between Ai and Pi according to the level of Ui - that is they will change Pi (and hence Ai) to increase their Ui.

B. The adapted evolutionary rewiring algorithm

After each time period (that is, after $N \bullet C$ node firings) the evolutionary rewiring algorithm (ERA), as used in NetWorld above, is applied. $N / 2$ pairs of nodes (i, j) are selected from the population at random with replacement. If $Ui > Uj$ then node j drops all existing links and copies node i's links and additionally links to node i itself. Also Pj is set to Pi (copying the query handling behavior of i). If $Ui < Uj$ then the mirror process is performed (i copying j). In the case $Ui = Uj$ then a randomly selected node (i or j) is designated "winner" and the process proceeds as if that node had a higher U value.

For the experiments presented here we used a utility (U) value equal to the total number of hits obtained by a node in the time period. Obviously this would tend to be higher if P was higher (generating more queries). We used the utility value of total hits (per node) since this gives an apparent incentive for freeloading – acting selfishly by generating queries but not answering them. If the average hits per query is used there is no commons tragedy – because nodes wont generally increase their utility by performing more queries.

After any node i copies another node j it applies mutation, with low probability, to the links and the Pi value. With probability m, Pi is changed to a random value selected uniformly from the range [0..1]. With probability $10m$ the links from i are removed and replaced with a single link to a randomly chosen node from the population.

C. Description of FileWorld

Each node i is defined by three state variables: an answering power Ai, a questioning power Pi and a capacity Ci. Both Ai and Pi are real values in the range [0..1]. Ci takes some cardinal value greater than zero. The values of each variable quantifies the behavior of a node over some unit of time t. Ci indicates the capacity of the node, given as total number of queries. When a node generates or answers a query this takes one unit of capacity. Pi gives the proportion of the capacity Ci that will be allocated to generating new queries. Conversely, $1 - Pi$ of the capacity will be allocated to answering queries from other nodes. The answering capacity, Ai, gives a probability that a node can directly match a query - producing a hit. It represents, indirectly, the amount and quality of files served by the node. For the experiments given here all nodes have fixed values of $Ai = 0.4$ and $Ci = 100$ but we allow Pi to be adapted by the node.

Over a single time period each node i may process a total of Ci queries. This capacity is divided between generating $Pi \bullet Ci$ new queries (passed to neighbor nodes) and reserving enough capacity to process $(1\text{-}Pi) \bullet Ci$ queries from neighbors. Pi therefore represents a measure of selfishness. If $Pi = 1$ then node i uses all its capacity to generate new queries - ignoring queries from neighbors. If $Pi = 0$ then i uses all its capacity processing queries from neighbors - generating none itself.

In a simulated time period, $C \bullet N$ nodes (N = node population, $C = 100$, the same as each Ci) are selected randomly from the population, with replacement, and "fired". If a fired node still has capacity to generate queries it generates one query and passes this to its neighbors otherwise the node takes no action. When a node i receives a query, if it has spare capacity, it processes the query. With probability Ai a "hit" is produced for the query. If no hit is produced the query is passed to the neighbor nodes of i. If a node has no capacity left to process a query it is ignored – no action is taken and the query is not passed on. Queries are not passed on indefinitely but have a preset "time-to-live" (TTL) after which they are ignored by all nodes. In all experiments presented here TTL = 3 - meaning that queries never get more than a maximum of 3 nodes depth from the originating node. The process of firing nodes in random order with replacement introduces noise in the form of some nodes firing more often than others and some nodes not being able to generate their full quota of queries. We view this as reasonable since it introduces realistic kinds of noise such as non-synchronized nodes with differential processing speeds etc.

For the purposes of simulation we represent the network as an undirected graph in which the degree of any node is fixed at a maximum value (20 in all cases). When any operation requires a further link from a node, the node simply deletes a randomly chosen existing link and continues with the new link operation. We experimented with various initial topologies for the graph, including randomly connected, lattice, "small world" and completely disconnected. All produced similar results to those presented here. We also experimented with different initial P values. Again we found we obtained similar results (even when all P values are

initially set to zero). Essentially then, we found the results obtained were insensitive to the initial conditions of the network. The results given, unless otherwise stated, start with initially random graphs and randomly selected *P* values

D. Results from FileWorld

In order to gain a baseline benchmark, that measures how the network behaves without the application of the evolutionary re-wiring algorithm, we ran, and averaged over, 10 trials (each an average over initial 10 cycles) on static networks with randomly initialized topologies and *P* values. We did this for a number of network sizes N = [200..51200]. All other values were kept as previously described. Since in the static case the network does not change, the averaging over 10 cycles is done to smooth out the stochasticities of the model. Averaging over 10 different trials (with different pseudo-random number seeds) gives us a sample of different initial random network topologies and *P* values.

We considered the following two measures for benchmarking: the average number of queries generated per node in a cycle (*nq*) and the average number of hits per node generated per cycle (*nh*). We found that the baseline level for these measures was, with low variance, nq = 49.45 and nh = 20.13 in all cases. Calculating *nh* / *nq* gives an average hit rate per query generated = 0.41. We might expect nq = 0.5 since the *P* values are selected uniformly randomly but this slightly lower value is a result of the (random selection with replacement) method of firing nodes as described earlier.

Given these baseline values for *nq* and *nh* we can investigate the effect of applying the evolutionary rewiring algorithm (ERA). If results give a consistently higher number of hits (*nh*) by keeping the number of queries generated (*nq*) low then ERA is suppressing the self-interest of the nodes and thus benefiting the network as a whole. Fig. 8 shows a time series for a typical run for a network of size $N = 10^4$ with ERA enabled. As can be seen, over time, *nq* decreases and *nh* increases. Notice also that initially these values move in the opposite direction, indicating an initial favoring of selfish behavior, but this is soon corrected. This shows that the evolutionary process (forming cooperative groups within the network) takes a few cycles to get started from the initially randomly initialized network.

Fig. 9 shows *nq* and *nh* measures averaged over cycles 40-50 for different network sizes with 10 independent runs for each network size. Notice that as the size of the network increases the variance of the individual runs decreases – indicating that larger networks are less sensitive to on-going stochasticities and initial conditions, since we are using average values this appears intuitive. Fig. 10 shows the numbers of cycles before high hit values are attained (when *nh* > 30). Again 10 independent runs are shown for various network sizes. As before the variance of results decreases as network size increases and there is no significant increase in the number of cycles required for larger networks - suggesting that ERA scales well in this scenario also (i.e. there is no cost for larger networks).

E. FileWorld discussion

Our initial experiments with the FileWorld model suggest that ERA does indeed control the self-interest of the nodes by keeping down the number of queries generated and hence increasing total hits. We appear to have preserved the zero scaling cost properties from the NetWorld model. This implies that the ERA has, at least some, general applicability – it performed well without changing the basic algorithm for the FileWorld task domain.

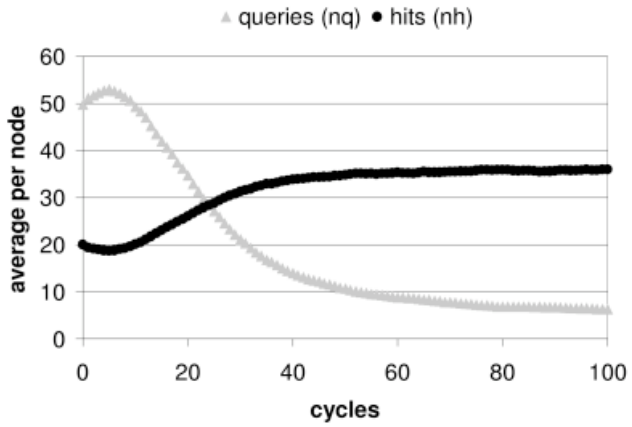

Fig. 8. A typical run for a 10^4 node network. Notice that *nq* and *nh* values initially get worse than the baseline values ($nq \approx 50$, $nh \approx 20$) but then quickly improve.

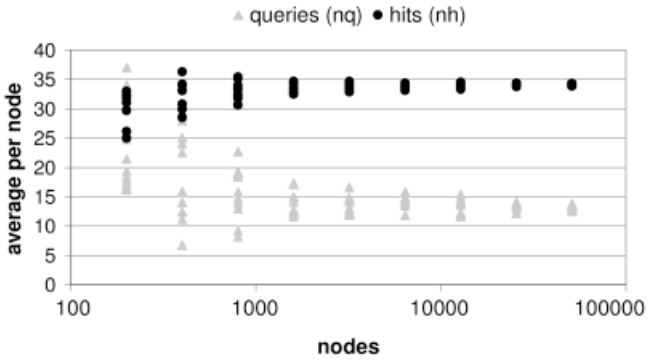

Fig. 9. Shows the values of *nq* and *nh* (averaged over cycles 40..50) for different network sizes. Ten independent runs for each for each network size.

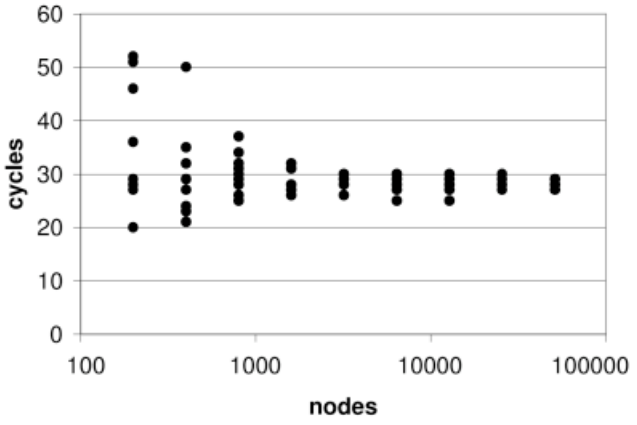

Fig. 10. Shows cycles required before high hit values (nh > 30) are attained. Ten independent runs for each for each network size are shown.

Although the ERA performs well in the FileWorld domain, this is currently a rather generous task domain. By fixing the answering power of each node to a relatively high level (40% of queries can be answered) we simulate a "file rich" environment [23]. It would be of interest to perform experiments with an initially reduced answering power or add a cost to nodes having high answering powers (diminishing utility in some way) and allow them to adapt in the same way as their links and questioning power. Under those conditions would the system still produce high hit rates?

In addition, we have not explored, in depth, the kinds of topologies produced in the NetWorld or FileWorld domains. In both these scenarios, a network disconnected into a number of components will still produce respectable functionality (since long range routing is not a requirement of these task domains). In future work we may refine the task domain to include long range routing between distant nodes.

Our initial investigations of NetWorld and FileWorld topologies shows that networks tend to form into disconnected components that constantly grow and reform (rather like the tag groups in TagWorld). Interestingly, since the network is in constant flux it would appear that forms of "temporal routing" might be applicable for long-range task domains [21].

VI. Related work

The mechanism we apply is distributed; each node only has to concern itself with its own interactions. In this way, it contrasts to centralized systems of trust that could provide similar kinds of functionality [26].

ERA bears similarities to the more complex "SLIC" algorithm [23]. In SLIC (from which we adapted our FileWorld task scenario) "incentive structures" are explicitly programmed into the nodes. Each node monitors the service it receives from its neighbors and updates weights which moderate the future service it offers to others. There is therefore explicit retaliation programmed into the model – similar to that applied in actual deployed peer applications [27]. If a neighbor is acting selfishly then the node will detect this and cease to share with that neighbor. Thus, each node has to maintain a list of information about all of its neighbors.

In the FileWorld model the "incentives" effectively emerge from the dynamic behavior of the nodes (moving in the network) rather than being explicitly programmed in. Nodes therefore do not need to monitor or store the performance of others – reducing overheads. In addition, in SLIC [23] simulations are only applied to scenarios in which single "probe nodes" behave selfishly – nodes do not adapt their behavior to increase their utilities network wide. Consequently, it is not clear how that model would react when all nodes are acting selfishly rather than just a small number (however, this is mentioned as future work).

The APT protocol [28] compares closely with SLIC and ERA. In APT the nodes explicitly store trust values based on past interaction and move within the network to maximize these trust values rather than a measure of selfish utility (such as the number of high quality files personally downloaded by the node). In this latter sense, the nodes are programmed to act in a non-selfish way. However, APT is shown to be effective against a number of attach scenarios in which selfish or malicious nodes enter the network and to produce efficient topologies based around content level semantic clustering.

Previous models inspired by game theoretical approaches offer some similar insights [29]. This work relies on there having being a sequence of interactions in the past so that it is possible to remember what occurred in past interactions between the same entities, i.e. the tit-for-tat strategy (in which you cooperate if they did in the past) in the iterated PD [2], [16]). However, neither the P2P network topology nor the strategies are modeled there. Such iterated strategies require on-going interactions with recognizable individuals – our model does not rely on this since it is based on mechanisms that work well in the single-round game – allowing interactions with strangers.

Our model bears some comparison to the social simulation of leadership dynamics presented in [30]. In these dynamics clusters and chains of high cooperation form around so-called "leader agents" in a simulated social network. Since such agents can break links with non-cooperating neighbors they can be viewed as "leaders" moving large sub-networks of cooperative "followers" to better locations in the social network. However, this model is deterministic and relies on agents knowing the strategies of others when moving. In addition, like the NetWorld model, presented previously, interaction is only between immediate neighbors – there is no indirect interaction through intermediate nodes.

VII. Conclusions and Open Issues

The desirable properties from novel tag models [3], [4], [6] have been carried over into a P2P file-sharing scenario. The ERA algorithm potentially offers a generally applicable mechanism for controlling selfish behavior in many possible P2P task domains without the need to program and test explicit incentive mechanisms for each domain. In the work presented here, the ERA algorithm effectively emerges an incentive mechanism from the selfish moving behavior of the nodes. This happens because, although it may do well for a while, a very selfish node will tend to lose neighbors as they find other nodes that are members of more cooperative groupings and hence have higher utilities. Additionally, selfish nodes doing well (exploiting neighbors) are a signal for copycats to latch onto them and their neighborhood and exploit it - speeding up the dissolution of the cluster.

We have moved closer to our target, P2P, application by a progressive succession of models. The results appear encouraging and we are confident that further iterations of the process would lead to a sequence of models finishing in a deployable implementation.

There are several open issues about adapting a model that has started life in the social or biological sphere, towards engineering applicability that need further exploration. These include: how to capture conceptions of utility and utility

comparison; how to interpret and implement reproduction; thirdly, how to deal with non-adaptive agents; and finally the various methods by which on could implement random selection in the population.

Utility is a fundamental concept in much evolutionary modeling. It is assumed that some value or measure can be determined for each agent that defines its utility or fitness. In the models described in this article the relative simplicity of the scenarios means that selecting a utility measure is easy – this may not be the case for more complex domains. This is especially true when rewards are delayed. It would be desirable therefore to eliminate a simplistic conception of utility and move towards a more general "performance targets" arrangement. We have assumed in our model that agents can freely compare utility but this would be problematical in many application domains. Two possible workarounds are possible; we could abandon maximization of utility and apply a "satisficing" metric. When agents "satisfice" they do not change behavior after they reach some "aspiration" level of performance. This has some intuitive appeal and would appear easy to implement but it raises new questions: how does one determine the aspiration level? How and when, if at all, should it change? Interestingly, because humans are considered to often "satisfice" - rather than optimize, there are bodies of work, including simulations, exploring this technique going back to the ideas of Simon [31]. The other potential solution is for nodes to monitor the performance they receive from at least two subsets of their neighbors and then drop and replace the links to the poorer performing neighbor subset.

For simulation models of biological or cultural phenomena the idea of reproduction makes sense because successful behaviors are copied and increase. But within an application domain, such as a P2P system, nodes don't reproduce directly. It can be claimed [29] that we can model nodes as reproducing because this captures the notion of human users of systems installing and using clients that appear to offer desirable results - fast file downloading say. It is important to realize the dramatic implications of this interpretation however; it is that we are no longer modeling potential deployable mechanisms alone but rather the behavior of a system with *humans within the loop*. Currently there appears insufficient data (although there is some [13]) to know how humans behave in these contexts.

This latter problem (that of non-adaptive agents) leads onto our next major issue – how do you ensure agents (or nodes in our case) do in fact follow an adaptive process. In our models we have assumed all the agents follow the same adaptive process, but, in a real system, why would they? A robust system cannot assume this. Our conception of robustness has been that the system is resistant to mutation on selected components under the assumption of utility maximization but what if a subset of the population simply stopped adapting and acted in a non-greedy way: ignoring the fact that others were doing better? How robust would our model be then? This kind of "whitewashing" behavior can be explored, in each domain, by running various attack scenarios. We intend to test the robustness of the ERA against such attacks in a number of task domains in future work.

The ERA algorithm presented allows nodes to move around a network based on self-interest. Nodes are not required to store any additional information about neighbors or to calculate trust scores. The ERA makes no distinction between stranger nodes and those for which there has been on-going interactions and therefore is suited to a highly dynamic environment. We find it promising that such a simple algorithm has performed well in several simulated scenarios with little or no "tweaking" of parameters. We aim to develop the algorithm and apply it to a number of real P2P application task domains based on newly emerging tools and infrastructures [32].

ACKNOWLEDGMENT

We thank Mark Jelasity, Alberto Montresor and Ozalp Babaoglu from the University of Bologna, Dept of Computer Science, for perceptive discussions, observations and pointers concerning P2P systems. We thank the reviewers for their suggestions on how to improve the article – their comments were of great help.

REFERENCES

[1] G. Hardin, "The Tragedy of the Commons," *Science*, 162, 1968, pp. 1243-1248.
[2] R. Axelrod, *The Evolution of Cooperation*. New York: Basic Books, 1984.
[3] D. Hales, "Cooperation without Space or Memory: Tags, Groups and the Prisoner's Dilemma," in *Proc. 2nd Int. Workshop on Multi-Agent-Based Simulation - LNAI 1979*, Berlin: Springer, 2000, pp.157-166.
[4] D. Hales, "Tag Based Cooperation in Artificial Societies," Ph.D. dissertation, Dept. Comp. Sci., Essex Univ., Colchester, UK, 2000.
[5] R. L. Riolo, M. D. Cohen and R. Axelrod, "Evolution of cooperation without reciprocity,". *Nature* 414, 2001, pp. 441-443.
[6] D. Hales and B. Edmonds, "Evolving Social Rationality for MAS using Tags," in *Proc. 2nd Int. Conf. on Autonomous Agents and Multi-agent Systems*, Melbourne, 2003, ACM Press, pp. 497-503.
[7] J. Epstein and R. Axtell, *Growing Artificial Societies: Social Science from the Bottom Up*. Cambridge, MA: The MIT Press, 1996.
[8] N. Gilbert and R. Conte, *Artificial Societies: the Computer Simulation of Social Life.* London: UCL Press, 1995.
[9] N. Gilbert and J. Doran, *Simulating Societies: the Computer Simulation of Social Phenomena*. London: UCL Press, 1994.
[10] D. Hales, B. Edmonds, E. Norling and J. Rouchier (2003) *Multi-Agent-Based Simulation III, LNAI 2927*. Berlin: Springer, 2003.
[11] JASSS, The Journal of Artificial Societies and Social Simulation [Online]. Available: http://jasss.soc.surrey.ac.uk/JASSS.html
[12] B. Edmonds and J. Bryson, "The Insufficiency of Formal Design Methods - the necessity of an experimental approach for the understanding and control of complex MAS," in *Proc. 3rd Int. Conf. Autonomous Agents and Multi-agent Systems,* New York, 2004, ACM Press.
[13] E. Adar and B. A. Huberman. (2000, October). Free Riding on Gnutella. *First Monday* [Online]. 5(10). Available: http://www.firstmonday.dk/issues/issue5_10/adar/
[14] Gnutella. http://www.gnutella.com
[15] Kazaa. http://www.kazaa.com
[16] R. Trivers, "The evolution of reciprocal altruism," *Q. Rev. Biol.* 46, 1971, pp. 35-57.
[17] D. Hales, (2002, October). Group Reputation Supports Beneficent Norms. J. of Artificial Societies and Social Simulation [Online]. 5(4). Available: http://www.soc.surrey.ac.uk/JASSS/5/4/4.html
[18] M. Nowak and R. May, "Evolutionary Games and Spatial Chaos," Nature, 359, 1992, pp. 826-929.
[19] J. M. Smith, *Evolution and the Theory of Games*. Cambridge: Cambridge University Press, 1982.
[20] J. Holland, "The Effect of Labels (Tags) on Social Interactions," *Santa Fe Institute Working Paper 93-10-064*. Santa Fe, NM, 1993.

[21] K. Sigmund. and M. A. Nowak, "Tides of Tolerance," *Nature* 414, 2001, pp. 403-405.
[22] B. Edmonds and D. Hales. (2003, October). Replication, Replication and Replication - Some Hard Lessons from Model Alignment. *J. of Artificial Societies and Social Simulation* [Online]. 6(4). Available: http://jasss.soc.surrey.ac.uk/6/4/11
[23] Q. Sun and H. Garcia-Molina, "SLIC: A Selfish Link-based Incentive Mechanism for Unstructured Peer-to-Peer Networks," in *Proc. of 24th IEEE Int. Conf. on Distributed Systems,* IEEE Computer Society, 2004.
[24] D. Hales, "From Selfish Nodes to Cooperative Networks – Emergent Link-based Incentives in Peer-to-Peer Networks," in *Proc. of the 4th Int. Conf. on Peer-to-Peer Computing*, IEEE Computer Society, 2004.
[25] D. Kempe, J. Kleinberg and A. Kumar, "Connectivity and inference problems for temporal networks," in *Proc. 32nd ACM Symposium on Theory of Computing*, 2000.
[26] S. Kamvar, M. Scholosser and H. Garcia-Molina, "The EigenTrust Algorithm for Reputation Management in P2P networks," in *Proc.12th Int. World Wide Web Conf.,* 2003. Available: http://www2003.org/cdrom/
[27] B. Cohen, "Incentives build robustness in BitTorrent," presented at the Workshop on Economics in Peer-to-Peer Systems, Berkeley, CA, June 5-6, 2003. Available: http://www.sims.berkeley.edu/research/conferences/p2pecon/
[28] T. Condie, S. Kamvar, H. Garcia-Molina, (2004) "Adaptive Peer-To-Peer Topologies," in *Proc. of the 4th Int. Conf. on Peer-to-Peer Computing*, IEEE Computer Society, 2004.
[29] K. Lai, M. Feldman, I. Stoica, and J. Chuang, "Incentives for cooperation in peer-to-peer networks," presented at the Workshop on Economics in Peer-to-Peer Systems, Berkeley, CA, June 5-6, 2003. Available: http://www.sims.berkeley.edu/research/conferences/p2pecon/
[30] M.G. Zimmermann, V. M. Egufluz and M. San Miguel, "Cooperation, adaptation and the emergence of leadership," in *Economics with Heterogeneous Interacting Agents*. A. Kirman and J. B. Zimmermann Eds. Berlin: Springer, 2001, pp. 73-86.
[31] H. A. Simon, Administrative *Behavior*. 3rd edition. New York: The Free Press, 1976.
[32] M. Jelasity, A. Montresor and O. Babaoglu, (2004), "A Modular Paradigm for Building Self-Organizing Peer-to-Peer Applications," in *Engineering Self-Organising Systems - Nature-Inspired Approaches to Software Engineering, LNAI 297*, G. Di Marzo Serugendo, A. Karageorgos, O. F. Rana, F. Zambonelli Eds. Berlin: Springer, 2004, pp. 265-282.

David Hales. BSc (Hons) Computer Science (Aston University), 1991; MSc Artificial Intelligence (Essex University), 1995; PhD (Essex University) 2001 on "Tag Based Cooperation in Artificial Societies".

He is a Postdoctoral Researcher at the Dept. of Computer Science, University of Bologna, Italy. A full list of his publications is available from his website at http://www.davidhales.com. His research interests include: agent-based social simulation, self-organizing software, the application of social theories to computational systems and P2P systems.

Dr. Hales is an elected officer and member of the European Social Simulation Association (ESSA).

Bruce Edmonds. BA mathematics (Oxford University), 1983; PhD (Manchester University) 1999 on "Syntactic Measures of Complexity."

He is Senior Research Fellow in Logic and Formal Methods at the Centre for Policy Modelling, Manchester Metropolitan University, UK. A full list of his publications is available from his website at http://bruce.edmonds.name. His research interests include: the methodology and practice of computational social simulation; the application of social theories to computational systems; and the use of context in cognition and AI.

Dr. Edmonds is an officer and member of the European Social Simulation Association as well as the European Networks of Excellence in complexity (EXYSTENCE) and software agents (AgentLink).