Designing Incentives for Peer-to-Peer Routing (Technical Report)

Alberto Blanc
Electrical and Computer Engineering Dept.
University of California, San Diego

Yi-Kai Liu, Amin Vahdat
Computer Science and Engineering Dept.
University of California, San Diego

***Abstract*— In a peer-to-peer network, each node is typically required to route packets on behalf of other nodes. We study the problem of designing incentives to ensure that nodes carry out this responsibility. We model the interactions between nodes as a "random matching game," and describe a simple reputation system that can provide incentives for good behavior. Using simulations, we investigate the robustness of this scheme in the presence of noise and malicious nodes, and we attempt to quantify some of the design trade-offs.**

I. Introduction

Peer-to-peer networks suffer from the problem of free-loaders, users who consume resources on the network without contributing anything in return. Originally it was hoped that users would be altruistic, "from each according to his abilities, to each according to his needs." In practice, however, altruism breaks down as networks grow larger and include more diverse users. This situation can lead to a "tragedy of the commons," where the individual players' self-interest causes the system to collapse.

This paper focuses on a special version of the free-loader problem which arises in peer-to-peer routing. Each node in the network relies on other nodes to forward its requests, and it in turn is expected to forward the requests sent by other nodes. However, a self-interested user might choose to free-load by refusing to forward requests, conserving local bandwidth.

To reduce free-loading, the system as a whole must provide incentives for behavior that maximizes aggregate utility while delivering acceptable payoffs to individual users. This paper investigates one such scheme, using tools from game theory. First, we model a peer-to-peer network using a random-matching game. This game was previously studied by Kandori, who showed that, if there is a simple reputation system, then cooperation can be sustained as a robust and subgame-perfect equilibrium [1]. We extend Kandori's result by showing that this equilibrium can tolerate malicious nodes and noise in the system.

We then consider a more complicated version of the random matching game that models peer-to-peer routing. Under certain assumptions, we again get a subgame-perfect equilibrium. We use simulations to show that cooperation in this game can be sustained in the presence of malicious nodes and noise. We also attempt to quantify some of the design trade-offs; in particular we demonstrate that the reputation system can still be effective even if it only monitors a small fraction of routing events. A key contribution of this paper is quantifying the necessary effectiveness of a reputation scheme to enforce the incentives required to deliver high levels of global aggregate utility. Our initial results are encouraging, with implications for the design of such a practical system.

There is a substantial body of related work on using incentives in computer science; for a recent survey, see [2]. For peer-to-peer networks, a variety of reputation schemes have been proposed; most of them are based on notions of currency (money), trust relationships, or some combination of the two [3]. These schemes can be quite sophisticated, and are not easy to analyze. Also, incentive problems have been extensively studied in economics. Useful techniques can be found in game theory and mechanism design [4]. In particular, one can view peer-to-peer networks as an example of a public goods problem [5].

Section II in this paper describes the basic random-matching game and the equilibrium strategy, and gives some theoretical analysis; section III presents some simulation results. Section IV describes the peer-to-peer routing game, and section V presents simulation results for that game. Section VI discusses some open issues and related work. Section VII concludes the paper.

II. The Random Matching Game

A. The Prisoner's Dilemma

The Prisoner's Dilemma is a well-known game, with the following payoff matrix:

	C	D
C	R, R	S, T
D	T, S	P, P

Here C means "cooperate," D means "defect," and the payoffs consist of R (reward), P (punishment), T (temptation) and S (sucker). The payoffs satisfy the inequalities $T > R > P > S$ so that, for each player, defection is a dominant strategy. The dilemma comes from the fact that, if both players had

E-mail: alberto@cwc.ucsd.edu
E-mail: y9liu@cs.ucsd.edu. Supported by the National Security Agency (NSA) and Advanced Research and Development Activity (ARDA) under Army Research Office (ARO) Grant No. DAAD19-01-1-0520.
E-mail: vahdat@cs.ucsd.edu. Supported by NSF grant ITR-0082912, NSF CAREER award CCR-9984-328, Hewlett-Packard, IBM, Intel and Microsoft.

cooperated, they could have achieved a much more desirable outcome.

In the context of networks, the Prisoner's Dilemma models two nodes who want to trade resources. Suppose that if node 1 agrees to service node 2's request, node 1 pays some cost $c > 0$, and node 2 receives some benefit $b > 0$; and let $b > c$, so it is a positive-sum game. Also, assume the game is symmetric, i.e. the costs and benefits are the same if node 2 is servicing node 1's request. Then we have a Prisoner's Dilemma: a node cooperates if it services a request, a node defects if it ignores a request, and the payoffs are

$$T = b, \quad R = b - c, \quad P = 0, \quad S = -c$$

As noted earlier, in a single-round Prisoner's Dilemma, rational players will always defect. In a repeated game, however, the situation is different because cooperation can be sustained by the threat of punishment in the next round. One effective strategy is "tit-for-tat," where each player cooperates in the first round, and in each subsequent round, does whatever his opponent did in the previous round. There are a wide variety of possible strategies; see [6] for a survey of work in this area.

B. Random Matching

The main difficulty with peer-to-peer networks is that users do not form long-lived relationships with other users, so strategies like "tit-for-tat" do not work. The common case is to interact with a stranger, with no prior history and no expectation of meeting again in the future. We model this using Kandori's random matching game [1]: In each round, nodes are randomly matched, and then each pair plays a (single-round) Prisoner's Dilemma. For simplicity, we will do matchings between the left and right vertices in a complete bipartite graph. We do allow non-uniform random matching.

It would seem that cooperation cannot be sustained between complete strangers unless they have some way to determine each other's past behavior before making their moves. Indeed, the strategy described below does rely on a primitive reputation scheme. But Kandori also found another equilibrium strategy that does not require any information; instead, it relies on a global threat that any deviation from equilibrium will eventually cause the system to collapse. This is completely impractical, since in a real system there are always errors and mistakes, which should be tolerated. This example illustrates some of the issues in applying game theory to a realistic situation.

C. A Simple Equilibrium Strategy

We now describe a simple strategy for the random-matching game and prove that it is a subgame-perfect equilibrium. This result is due to Kandori [1]. In this paper we refer to it as the "social norm" strategy.

Each node has a reputation consisting of a number in the range $\{0, 1, ..., \tau\}$; 0 means innocent, nonzero means guilty. The reputations are maintained by a trusted authority, who observes the players' actions and updates their reputations accordingly. Essentially, the reputations ensure that a node who defects will be punished in the next round, even though it plays a different opponent in each round. The strategy is as follows:

- If the two players are innocent, they both cooperate.
- If the two players are guilty, they both defect.
- If one is innocent and one is guilty, then the guilty player cooperates, and the innocent player defects.

Any deviation from the above strategy triggers a punishment that lasts for τ rounds. That is, the offending node is marked "guilty," causing it to be punished by its opponents. After τ rounds, the node becomes innocent again, provided it has followed its assigned strategy. If a node deviates during the τ-round punishment phase, the punishment is re-started from the beginning. To implement this protocol, the trusted authority uses the following rule to update the reputations:

- If a node is innocent and it follows its assigned strategy, then leave its reputation unchanged.
- If a node is guilty and it follows its assigned strategy, then decrement its reputation by 1.
- If a node deviates, then set its reputation to τ.

Observe that a node's action only depends on the reputation of its opponent, and a node's reputation only depends on its own action.

Kandori showed that the social norm strategy is a subgame-perfect equilibrium, provided that we set the punishment length τ and the discount factor δ correctly. The discount factor can be interpreted as the probability that a player will participate in the next round of the game. Setting $\delta = 1$ means that players are infinitely patient, while setting δ to a smaller value, say 1/2, means that players favor more short-term gains. The proof that this strategy is an equilibrium can be found in [1]; we reproduce it here for convenience.

The proof consists of checking the incentives of all the players:

For a guilty player who follows the assigned strategy, the payoff is

$$\geq (x(1) + \delta x(2) + \cdots + \delta^{\tau-1} x(\tau)) + (\delta^\tau + \delta^{\tau+1} + \cdots) R$$

where $x(t)$ is either S (if the opponent is innocent) or P (if the opponent is guilty). For a guilty player who deviates in this round, the payoff is

$$\leq P + (\delta x(2) + \cdots + \delta^{\tau-1} x(\tau) + \delta^\tau S) + (\delta^{\tau+1} + \cdots) R$$

Note that after τ rounds, all the other players will be innocent. The difference between these two expressions is

$$\begin{aligned} &\geq x(1) - P + \delta^\tau (R - S) \\ &\geq S - P + \delta^\tau (R - S) \end{aligned}$$

This must be ≥ 0; so we require the following condition:

$$\boxed{\delta^\tau \geq \frac{P - S}{R - S}}$$

For an innocent player who follows the assigned strategy, the payoff is

$$\geq (1-\delta^\tau)R + \delta^\tau R = R$$

For an innocent player who deviates in this round, the payoff is

$$\leq (1-\delta)T + (\delta - \delta^{\tau+1})P + \delta^{\tau+1}R$$

To ensure that the first expression is greater than the second, we require that:

$$\boxed{R \geq (1-\delta)T + \delta((1-\delta^\tau)P + \delta^\tau R)}$$

Now we will show how to set τ and δ so that the two inequalities are both satisfied. Fix $\delta^\tau = \gamma$, so we have $\tau = \log_\delta \gamma = \frac{\lg \gamma}{\lg \delta}$. The first inequality becomes:

$$\boxed{\gamma \geq \frac{P-S}{R-S}}$$

The second inequality becomes:

$$R \geq (1-\delta)T + \delta((1-\gamma)P + \gamma R)$$

$$\delta(T - (1-\gamma)P - \gamma R) \geq T - R$$

$$\boxed{\delta \geq \frac{T-R}{T-(1-\gamma)P-\gamma R}}$$

We can first pick γ to satisfy the first inequality, then pick δ to satisfy the second inequality, and calculate τ last. For example, if the payoffs are $T = 4$, $R = 2$, $P = 0$, $S = -2$, then we can set

$$\gamma = 1/2, \quad \delta = 2/3, \quad \tau = -\frac{1}{\lg(2/3)} \approx 1.71$$

We think that these parameter settings are reasonable for a practical system.

Finally, it is convenient to express the above inequalities in terms of the cost c and benefit b associated with servicing each request. Recall that $b > c > 0$, and the payoffs are given by:

$$T = b, \quad R = b - c, \quad P = 0, \quad S = -c$$

Then the inequalities are:

$$\boxed{\gamma \geq \frac{c}{b}} \qquad \boxed{\delta \geq \frac{c}{(1-\gamma)b + \gamma c}}$$

D. Tolerating Malicious Nodes

Here we describe some extensions to Kandori's original analysis, to look at the effect of malicious nodes, i.e., nodes that always defect. Let β be the fraction of malicious nodes, and suppose we use uniform random matching. Then one can derive new versions of the inequalities in the proof: (the actual calculation is straightforward but tedious)

$$\delta^\tau \geq \frac{1}{1-\beta}\frac{P-S}{R-S}$$

$$\begin{aligned}(1-\beta)R + \beta(1-\delta^{\tau+1})S \\ \geq (1-\beta)((1-\delta)T + (\delta - \delta^{\tau+1})P + \delta^{\tau+1}R) + \\ \beta(1-\delta^{\tau+1})P\end{aligned}$$

Observe that when β is small, these inequalities are simply the original inequalities with some minor perturbations. Therefore, the equilibrium can tolerate a small fraction of malicious nodes (the precise threshold depends on other parameters such as the payoffs in the game).

One can also analyze the global efficiency of the game. In particular, if uniform random matching is used, then a β fraction of malicious nodes causes only a gradual decrease in performance. To see this, recall that we have a complete bipartite graph with N nodes on each side (and a β fraction of the nodes on each side is malicious). We can generate a uniform random matching by randomly permuting the N vertices on the right-hand side, randomly permuting the N vertices on the left-hand side, and then pairing off the vertices on the left and right sides.

For each match i, let X_i equal 1 if both players in the match are good (non-malicious), and 0 otherwise. Then $X = \sum_{i=1}^N X_i$ is the number of good matchings. Using linearity of expectation, we get that $E(X) = \sum_{i=1}^N E(X_i) = N(1-\beta)^2$. So, on average, a $(1-\beta)^2$ fraction of the matchings will be between two good (non-malicious) nodes. This can be used to calculate the overall efficiency, given the payoffs for the game.

We also tried to analyze the effect of noise, i.e., random errors that cause nodes to defect when they intended to cooperate. This appears to be somewhat harder, since noise can trigger punishments that last for more than one round. Intuitively, we expect that the amount of noise that can be tolerated will depend on the punishment length τ; if the expected time between errors is less than the punishment length, the system will break down.

III. Simulations of the Random Matching Game

We ran simulations to measure the effects of malicious nodes and noise in the random matching game.

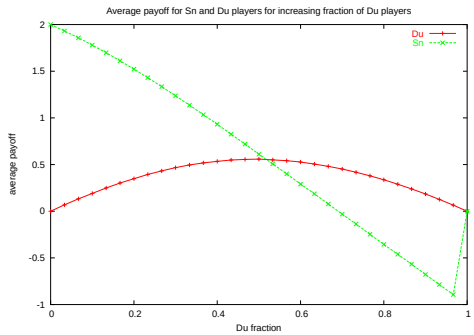
A. The Simulator

We simulated a random matching game with 1024 nodes and 1000 rounds. We considered three strategies: the social norm, denoted Sn; "defect unconditionally," denoted Du, which always defects; and "cooperate unconditionally," denoted Cu, which always cooperates. The Du and Cu strategies are intended to model malicious and altruistic nodes.

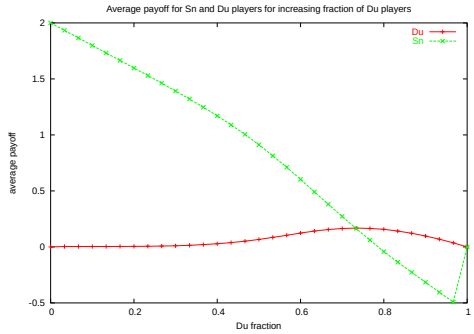
We also simulated an evolutionary game, where the number of players playing each strategy changes over time, depending on how well each strategy performs. After some number of rounds, the fraction of the population f_i that plays strategy i is updated according to:

$$f_i \leftarrow f_i \frac{V_i}{V}$$

where V_i is the average payoff of strategy i and V is the average payoff over the whole population. Hence, if a strategy



(a) Punishment $\tau = 1$



(b) Punishment $\tau = 5$

Fig. 1. Payoff per round of Sn and Du players as the number of Du players increases

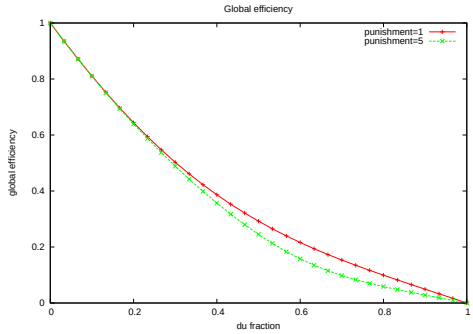


Fig. 2. Global efficiency as the number of Du players increases

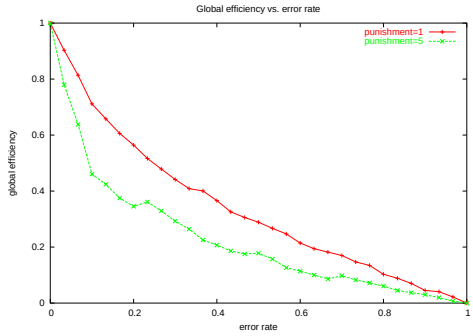


Fig. 3. Global efficiency as the error probability increases

does better than the average, a larger fraction of the population will start playing that strategy; and if a strategy does worse than the average, its fraction of the population will decrease. This process mimics biological evolution, and has been used before to study repeated games [6].

B. Simulation results

1) Malicious Nodes: Figure 1 shows the payoff per round of the Sn and Du players as the number of Du players increases. Not surprisingly, the Du players do better with punishment period $\tau = 1$ than with $\tau = 5$.

Observe that the payoffs of the Sn nodes drop significantly as the number of Du nodes increases. The payoffs of the Du nodes remain small, because they are marked guilty by the reputation system. However, the Du payoffs do rise slightly; this is because whenever a Du node defects on another Du node, it is actually behaving in agreement with the social norm, and thus its reputation will be restored. This phenomenon is less noticeable with the longer punishment period. Finally, when the fraction of Du nodes becomes too large, the Du payoffs fall back to zero since there are too few Sn nodes to exploit.

Note that there is a kink in the Sn curve at the right side of the graph; this is because when there are no Sn players, the average payoff for the Sn strategy is 0 by default.

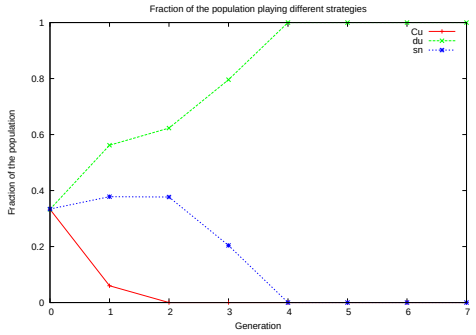
Figure 2 shows the global efficiency as the number of Du players increases. The global efficiency is the total payoff of all the players, divided by the total payoff of all the players assuming 100% cooperation. The results are similar with punishment period $\tau = 1$ and $\tau = 5$.

2) Noise: Next, we consider the effect of noise or errors in the system. We model this by setting p_{noise} to be the probability that a node who intends to cooperate will end up defecting instead; for instance, when a node attempts to forward a request, the request might get dropped due to a network failure. In these simulations, we use a network of all Sn nodes.

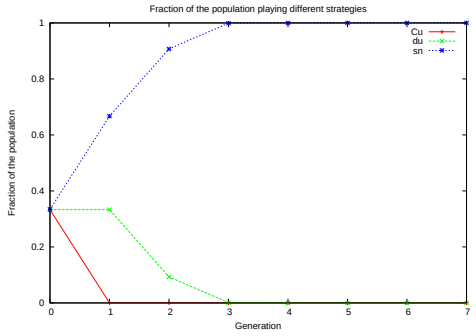
Observe in figure 3 that, as the noise level rises, the overall efficiency remains higher when we use the shorter punishment period. This illustrates a trade-off in choosing the punishment: longer punishments let us tolerate a higher fraction of malicious nodes, but shorter punishments let us tolerate a higher error rate.

3) Evolutionary Game: With these simulations we wanted to compare the relative merits of a variety of strategies. The results depend both on the initial population and on the punishment length. For example, for the starting population: Sn=1/3, Du=1/3 and Cu=1/3, with $\tau = 1$, Du overtakes the other strategies, while for $\tau = 2$ the social norm prevails. This is shown in figure 4. Different starting populations and different punishment lengths yield different results.

In general a longer punishment period helps the social norm, while a higher initial fraction of non cooperating (Du) players makes it harder for the social norm to prevail.



(a) Punishment $\tau = 1$



(b) Punishment $\tau = 2$

Fig. 4. Evolutionary game, with initial population $1/3$ Sn, $1/3$ Du, $1/3$ Cu

IV. The Peer-to-Peer Routing Game

To study the problem of peer-to-peer routing, we constructed a new kind of random matching game, and we defined an analogous "social norm" strategy for this game. We then ran simulations to measure the performance of the social norm strategy under varying conditions.

A. Peer-to-Peer Routing

We consider networks where each node has a routing table containing the addresses of a small number of nodes, and requests are forwarded through multiple hops until they reach their destinations. We use Chord [7] as an example, although this basic structure is found in many peer-to-peer networks.

We define a simplified model of routing as follows: We have a network of N nodes, arranged in a ring. Each node has a routing table of size $\lg(N)$, called its finger table, which contains the addresses of nodes ("fingers") that are located ahead of it on the ring at distances 2^i, $i = 0, 1, 2, \ldots, \lg(N) - 1$. That is, the fingers are at distances $1, 2, 4, \ldots, N/2$. To send a request, a node contacts the node in its finger table that is the closest predecessor to the destination node; this node then does the same using its own finger table, and so on until the destination node is reached. Sending a request thus takes at most $\lg(N)$ hops. To allow the sender to determine the identity of the node that dropped its request, we adopt iterative rather than recursive routing for our game.

Also note that a node will use some of its fingers more often than others: to send a request to a randomly-chosen destination, it will use its j'th longest finger with probability $1/2^j$. (That is, it uses its longest finger with probability $1/2$, its second longest finger with probability $1/4$, and its shortest finger with probability $1/N$; with probability $1/N$ it uses none of its fingers, i.e., the node chose itself as the destination.)

B. The Game

We define the peer-to-peer routing game as follows: We have N nodes, with routing tables as described above; the routing tables are filled in with randomly chosen neighbors before the start of the game. The game runs in continuous time, rather than discrete rounds: at any time, a node can send a request to be routed by the network. (The routing process is described below.) We assume that nodes do not control the generation of requests, but can only choose whether to route requests sent by other nodes. (Later, we will revisit this issue of how requests are generated.)

When a node sends a request, it is matched with a sequence of opponents, in a way that simulates the routing of a request to a destination chosen uniformly at random. For the first hop, the sender s is randomly matched with one of its fingers, choosing the j'th longest finger with probability $1/2^j$. In the case where none of the fingers is chosen (which happens with probability $1/N$), we match node s with its shortest finger.

Say that node s ends up matched with node t. The two nodes then play an asymmetric game: s does nothing, while t can either cooperate (forward the request) or defect (drop the request). At this stage, s does not receive any payoff, while t gets a payoff of -2 if it cooperates and 0 if it defects.

If t defects, then s is finished and gets payoff 0, since its request has been dropped. But if t cooperates, then s goes on to play another game—its request has been forwarded one hop, and it is now ready to make another hop. s can be matched with any of t's fingers that are *shorter* than t is as a finger of s. (In other words, the next hop must be shorter than the last hop.) We choose the j'th longest such finger with probability $1/2^j$.

Thus the game repeats, until either one of s's opponents defects, or s is matched with a finger of length 1 (which means there are no shorter fingers). Node s now plays the asymmetric game with this final opponent. If the opponent cooperates, s receives a large payoff of $+40$, because its request has reached its destination.

This completes the description of the game. We point out the following facts: First, this game uses non-uniform random matching. For the first hop, the matching is highly non-uniform, since there are only $\lg(N)$ possible choices (and one of them has probability $1/2$); but for later hops, the matching becomes more uniform.

Second, if we ignore the actual choices of the intermediate nodes, and simply look at the lengths of the hops, we observe that, for each $\ell = 1, 2, \ldots, \lg(N) - 1$, the probability of at some point taking a hop of length 2^ℓ is 1/2; for $\ell = 0$, the probability of taking a hop of length $2^\ell = 1$ is 1, but this is

really a quirk of the game. So the expected number of hops per request is $(\lg(N) - 1)/2 + 1 = \lg(N)/2 + 1/2$.

Finally, we think it is realistic that the sender receives a large payoff when its request reaches its destination, and nothing when its request gets dropped. A successfully delivered request presumably has a fairly high value to the sender, much higher than the cost of forwarding someone else's request; whereas, when a request gets dropped, the sender may learn some routing information, but it only amounts to a partial (and unreliable) route. Thus routing is a positive-sum game, but it is brittle, since a node that drops a request completely wipes out the sender's payoff. (Also note that as the network grows, the number of hops per request slowly increases. In order for the incentives to work, the final payoff must also increase, to balance out the cost of routing.)

C. The "Social Norm" Strategy

We would like to find an analogue of Kandori's "social norm" strategy, that will work in the peer-to-peer routing game. The routing game differs from Kandori's game in that it is asymmetric: in each round, we have node A requesting a service from node B and node B requesting a service from node C, where A and C are different.

$$A \longrightarrow B \longrightarrow C$$

However, it turns out that the social norm still makes sense in this situation. Using the social norm, what B should give to A depends only on A's reputation, and what B should receive from C depends only on B's reputation. So B only has to know about its own reputation and about A's reputation; it does not care if A and C are not the same entity.

So we can simply state the social norm strategy for the asymmetric game. Let A make a request to B. Then:

- If A is innocent, B cooperates; if A is guilty, B defects.

As before, we assume that there is a reputation scheme which marks nodes as innocent or guilty. We still assume that the reputation scheme is secure against tampering. However, we allow the reputation scheme to be unreliable in the following sense: If a node behaves properly, its reputation will always be updated correctly; but if a node misbehaves, the incident will only be detected with probability p_{rel}. This would describe a system that computes reputations based on random sampling. One of the goals of our simulations was to determine the amount of sampling, and the severity of punishment, that are needed to provide incentives that will deter cheating.

Finally, we need to specify what kinds of punishments will be enforced by the reputation scheme. We use a "time-based" punishment: when a node deviates from the social norm, it is punished for a period of time τ; if the node deviates again while it is being punished, the punishment period is re-started. During the punishment period, all requests sent by this node will be dropped; but this node will still be required to forward the requests of other innocent nodes. This is a natural way to do punishment in the continuous-time game, and it would not be hard to implement in a real system.

In certain cases, we can show that the social norm is a subgame-perfect equilibrium for the routing game. Specifically, if each node's requests are generated by a Poisson process with the same rate, then Kandori's original proof goes through with minor modifications. The intuition is that requests are generated at a smooth rate, so over the course of one punishment period, a node will ask other nodes to route its requests, and it will route requests for other nodes, roughly the same number of times. This situation is similar to the original (symmetric) random-matching game. The proof of this result is given in the following subsection.

Unfortunately, if requests are bursty, or if nodes can manipulate the timing of their requests, then the social norm may not be an equilibrium. If a node receives a very large burst of requests, it might be cheaper to drop the requests and undergo punishment. Also, a node can cheat by defecting while it accumulates a large number of requests, then cooperating just long enough to rebuild its reputation and send off all of the requests in one burst.

Finally, even when an equilibrium can be achieved, the routing game is not as robust in the presence of malicious nodes. This is due to the non-uniform matching. The burden of the malicious nodes falls disproportionally on a small group of honest nodes—namely, those nodes who have a malicious node as one of their frequently-used "long" fingers. For instance, a node whose longest finger is a malicious node will lose half of its requests. For these unlucky nodes, the incentives break down very quickly.

D. Proof of Equilibrium

First, some preliminary remarks: Suppose that some event occurs at random times specified by a Poisson process with rate λ. Let Y_i be the time of the i'th event, and T_i be the time between the $i-1$'st event and the i'th event. If we earn p points each time the event occurs, and δ is the discount factor, then our total payoff is

$$P = \delta^{Y_1} p + \delta^{Y_2} p + \cdots$$

and our expected payoff is

$$\mathrm{E}[P] = \mathrm{E}[\delta^{Y_1}]p + \mathrm{E}[\delta^{Y_2}]p + \cdots$$

Since $Y_i = T_1 + \cdots + T_i$, and the random variables $T_1, \ldots, T_i$ are independent and identically distributed, we have

$$\mathrm{E}[\delta^{Y_i}] = \mathrm{E}[\delta^{T_1} \cdots \delta^{T_i}] = \mathrm{E}[\delta^{T_1}] \cdots \mathrm{E}[\delta^{T_i}] = \mathrm{E}[\delta^{T_1}]^i$$

We define the "effective discount factor" to be $\delta_{\text{eff}} = \mathrm{E}[\delta^{T_1}]$. This lets us write the expected payoff as

$$\mathrm{E}[P] = \delta_{\text{eff}} p + \delta_{\text{eff}}^2 p + \cdots = p \frac{\delta_{\text{eff}}}{1 - \delta_{\text{eff}}}$$

The probability density function for T_1 is $p(t) = \lambda e^{-\lambda t}$, and so we have

$$\delta_{\text{eff}} = \mathrm{E}[\delta^{T_1}] = \int_0^\infty \delta^t \lambda e^{-\lambda t} dt = \frac{\lambda}{\lambda - \log \delta}$$

Also, the expected payoff during some time interval $[0, \tau]$ is

$$\mathrm{E}[P_{[0,\tau]}] = (1-\delta^\tau)\mathrm{E}[P] = (1-\delta^\tau)p\frac{\delta_{\text{eff}}}{1-\delta_{\text{eff}}}$$

The proof involves checking the incentives of each node. Each request has cost c for the node that services it, and benefit b for the node that sent it. In the case of routing, a single request may have to be serviced (forwarded) by several nodes. Each node sends requests at rate λ_s, and receives requests at rate λ_r. For routing, λ_r depends on how many other nodes are using this node as a finger; for the incentives to work, we need the gains to be large enough to offset the losses, even if this node has to forward more than its fair share of the requests, due to an unlucky configuration of the routing tables.

Punishment is measured in terms of time; for a guilty node, dropping a request delays its recovery. This delay time makes all the difference between following the social norm, and deviating from it. The non-burstiness of the requests is crucial. If multiple requests were to arrive simultaneously, a node could drop all of them without any additional penalty. But instead we ensure that a node has some time to recover its reputation after dropping a request; so, when a second request arrives, the node will have "something to lose" if it drops the request.

Note that punishing a node by dropping a certain *number* of requests does not create the right incentives. After dropping a request, a node will continue to drop requests, until the time when it *sends* its first request (which gets dropped). Time-based punishment is preferable for this reason: it provides an incentive for a node that has dropped a request to immediately resume forwarding requests.

So the situation for a guilty node is as follows: At time 0, we dropped a request and became guilty. Now, at time T_1, another request arrives. If we forward it, we will be forgiven at time τ; if we drop it, we will be forgiven no sooner than time $\tau + T_1$. This will affect any requests that we send during the interval $[\tau, \tau + T_1]$. Note that all the other guilty nodes are following the social norm, so they will be forgiven at time τ.

The expected payoff difference between following the social norm and deviating from it is

$$\begin{aligned}
&\geq -c + \mathrm{E}[\#\text{ of requests sent in } [\tau, \tau+T_1]]\delta^\tau b \\
&= -c + \lambda_s \mathrm{E}[T_1]\delta^\tau b = -c + \frac{\lambda_s}{\lambda_r}\delta^\tau b
\end{aligned}$$

We need this to be ≥ 0, that is,

$$\boxed{\delta^\tau \geq \frac{\lambda_r c}{\lambda_s b}}$$

Now consider the situation for an innocent node: The decision of whether to forward a request at time 0 will affect our payoffs during the interval $[0, \tau]$. Note that there may be other guilty nodes during this time. If we forward the request, we will stay innocent, gain b points each time we send a request, and lose at most c points each time we receive a request (this happens when the request is from an innocent node). If we drop the request, we will become guilty, gain 0 points each time we send a request, and lose 0 or more points each time we receive a request (we lose 0 points when the request is from a guilty node).

Let δ_{effs} and δ_{effr} be the effective discount factors for sending and receiving requests. The expected payoff difference between following the social norm and deviating from it is

$$\begin{aligned}
&\geq -c + (1-\delta^\tau)b\frac{\delta_{\text{effs}}}{1-\delta_{\text{effs}}} - (1-\delta^\tau)c\frac{\delta_{\text{effr}}}{1-\delta_{\text{effr}}} \\
&= -c + (1-\delta^\tau)b\frac{\lambda_s}{-\log\delta} - (1-\delta^\tau)c\frac{\lambda_r}{-\log\delta} \\
&= -c + (1-\delta^\tau)\frac{\lambda_s b - \lambda_r c}{-\log\delta}
\end{aligned}$$

We need this to be ≥ 0, that is,

$$\boxed{\frac{1}{-\log\delta} \geq \frac{c}{(1-\delta^\tau)(\lambda_s b - \lambda_r c)}}$$

To set the parameters, we first fix $\gamma = \delta^\tau$, then fix δ, and finally compute $\tau = \lg\gamma / \lg\delta$.

If the sending and receiving rates are equal, $\lambda_s = \lambda_r = 1$, and the benefit and cost are $b = 4$ and $c = 2$, then we get something similar to Kandori's random matching game, but running in continuous time and with asymmetric requests. We can set

$$\begin{gathered}
\gamma = 1/2 \\
\frac{1}{-\log\delta} \geq 2 \Longrightarrow \delta \geq e^{-1/2} \Longrightarrow \delta = 2/3 \\
\tau = -1/\lg\delta \approx 1.71
\end{gathered}$$

In the case of the routing game, with 1024 nodes, we set $\lambda_s = 1$ and $\lambda_r = 10$. (Each request takes 5.5 hops on average, but we add some margin to allow for irregularities caused by the particular configuration of the routing tables.) The benefit and cost are $b = 40$ and $c = 2$. Then we can set

$$\begin{gathered}
\gamma = 1/2 \\
\frac{1}{-\log\delta} \geq 1/5 \Longrightarrow \delta \geq e^{-5} \Longrightarrow \delta = 2/3 \\
\tau = -1/\lg\delta \approx 1.71
\end{gathered}$$

Note that, while these incentives are quite strong, they are sensitive to noise and malicious nodes.

V. Simulations

We ran simulations to measure the performance of the peer-to-peer routing game. For simplicity, we simulated a game with discrete rounds, where each node sends one request per round, and the nodes are randomly shuffled before each round (so the order of moves is random). This approximates a continuous-time game where each node's requests are generated by a Poisson process with the same rate.

We ran simulations with 1024 nodes and 1000 rounds. Each request took 5.5 hops on average, and the expected payoff per request (assuming 100% cooperation) was $40 - 5.5 \cdot 2 = 29$. (Recall that a node earns 40 points when its request reaches its

destination, and pays 2 points each time it forwards a request for another node.)

Punishment was measured in terms of rounds, with the reputations of the guilty nodes being decremented at the end of each round. We used punishment periods $\tau = 1$, $\tau = 2$ and $\tau = 5$. Note that with $\tau = 1$, every guilty node will be automatically forgiven at the end of the round; whereas with $\tau = 2$, a guilty node must cooperate for at least one full round before it is forgiven.

Error bars on the graphs show the 99% confidence intervals.

A. Simulation Results

1) Malicious Nodes: We show how the presence of varying fractions of malicious nodes affects the performance of the system. We model malicious nodes as nodes that always defect ("defect unconditionally" or "Du" nodes). Nodes that follow the social norm strategy are denoted "Sn" nodes. Figure 5 shows the payoff per request for the Sn and Du nodes, as the number of Du nodes increases. Note that punishment $\tau = 1$ is indeed much weaker than $\tau = 2$ or $\tau = 5$.

Observe that the payoffs of the Sn nodes drop significantly as the number of Du nodes increases, while the payoffs of the Du nodes remain small, because they are marked guilty by the reputation system. The cross-over occurs when the population is roughly 40% Du nodes. Note that there is a kink in the Sn curve at the right side of the graph; this is because when there are no Sn players, the average payoff for the Sn strategy is 0 by default.

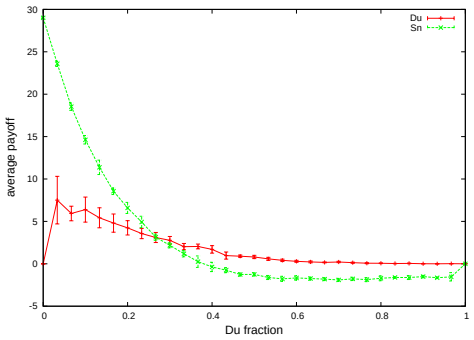
Figure 6 shows the global efficiency as the number of Du nodes increases. The results are similar with punishment periods $\tau = 1$, $\tau = 2$ and $\tau = 5$.

2) Noise: We next consider the effect of noise or errors in the system. We set p_{noise} to be the probability that a node who tries to cooperate will end up defecting instead (if, for instance, the request gets dropped due to a network failure). In these simulations, we use a network of all Sn nodes.

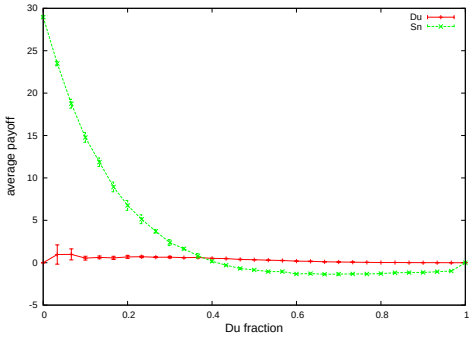
Observe in figure 7 that as the noise level rises, the overall efficiency remains higher when we use the shorter punishment period. This illustrates a trade-off in choosing the punishment: longer punishments let us tolerate a higher fraction of malicious nodes, but shorter punishments let us tolerate a higher error rate.

3) Reliability of the Reputation Scheme: Finally, we look at how the reliability of the reputation scheme together with the length of the punishment period affects the behavior of the system. Here we simulate a network with a 20% fraction of Du nodes. We plot the payoff per request for the Sn and Du nodes (figure 8).

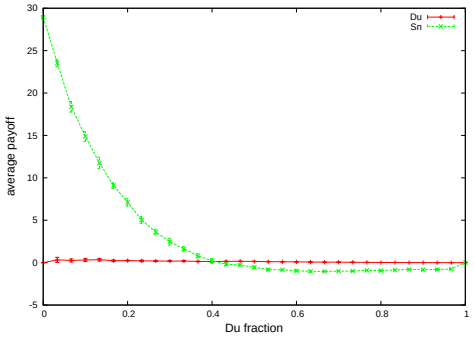
Observe that the reputation scheme is effective even when the reliability is quite low; this is partly because the Du nodes are easy to catch. In this particular scenario, to ensure that the Du nodes earn less than the Sn nodes, the reputation system must be at least 50% reliable when using a 1-round punishment, and at least 10% reliable when using a 5-round punishment.



(a) Punishment $\tau = 1$



(b) Punishment $\tau = 2$



(c) Punishment $\tau = 5$

Fig. 5. Payoff per request for Sn and Du nodes, as the number of Du nodes increases

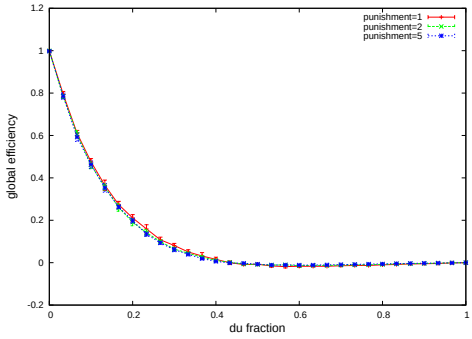


Fig. 6. Global efficiency as the number of Du nodes increases

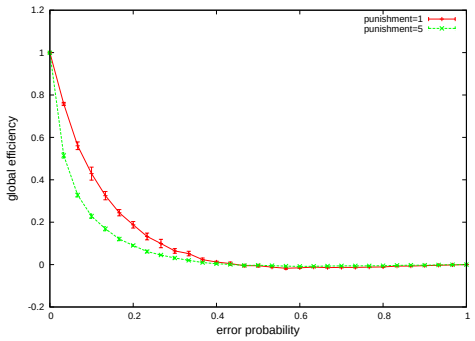
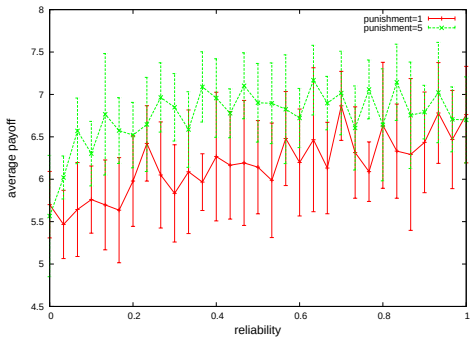
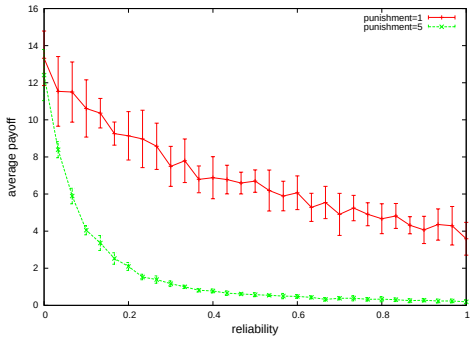


Fig. 7. Global efficiency with varying levels of noise



(a) Sn nodes

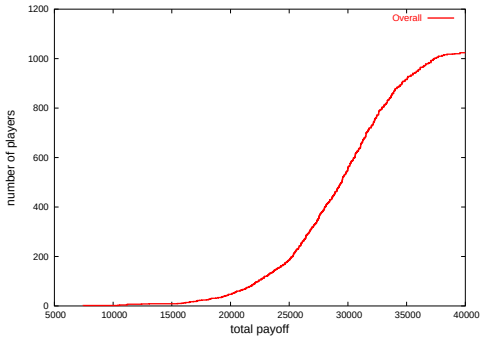


(b) Du nodes

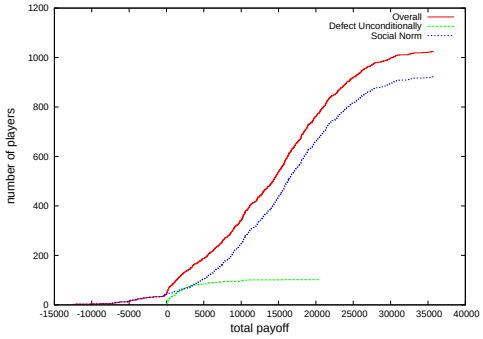
Fig. 8. Payoff per request for Sn and Du nodes, varying the reliability of the reputation scheme. Population is 20% Du nodes.

4) Distribution of Payoffs among the Nodes: To gain further insight into the effectiveness of the reputation system, we look at the distribution of the payoffs among the nodes. We assume a reputation system with fairly low reliability ($p_{rel} = 20\%$), but a fairly severe punishment ($\tau = 5$). We then vary the number of Du nodes, and plot the cumulative distribution function (CDF) of the total payoffs of the Sn and Du nodes (figure 9).

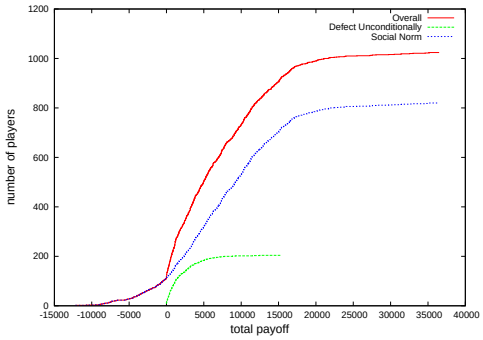
There is a fairly large variance in the payoffs of the Sn nodes. This is due to the random choices of the routing tables: if a node appears in many other nodes' finger tables, it will have to route many requests, reducing its own payoff. The presence of Du nodes also contributes to the variance: an Sn node whose longest finger happens to be a Du node will do very poorly, as half of its requests will be dropped. The payoffs of the Du nodes, on the other hand, are concentrated close to zero. (Note that, unlike Sn nodes, Du nodes never have negative payoffs.) This shows that the reputation system is effective.



(a) 0% Du nodes

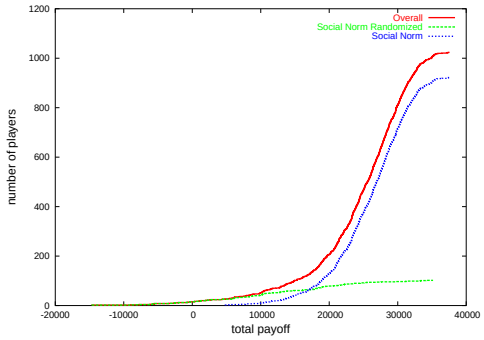


(b) 10% Du nodes

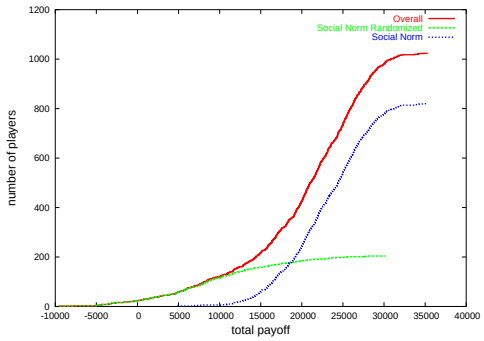


(c) 20% Du nodes

Fig. 9. CDF of the total payoffs of the Sn and Du nodes, with reliability $p_{rel} = 20\%$ and punishment $\tau = 5$



(a) 10% Snr nodes



(b) 20% Snr nodes

Fig. 10. CDF of the total payoffs of the Sn and Snr nodes (with $p_{def} = 20\%$), with reliability $p_{rel} = 20\%$ and punishment $\tau = 5$

B. Improved Strategies

1) Social Norm with Random Defections: Another possible strategy is the social norm with random defections, denoted Snr:

- If our opponent is innocent, we cooperate with probability $1-p_{def}$ and defect with probability p_{def}; if our opponent is guilty, we defect.

Note that Snr is actually a family of strategies, with parameter p_{def}, which includes both Du (when $p_{def} = 1$) and Sn (when $p_{def} = 0$). Typically we would set p_{def} to an intermediate value, say $p_{def} = 0.2$. The intuition here is that an unreliable reputation system will have trouble detecting intermittent or occasional defections. So, although Snr is less aggressive than Du, it may do better because it can avoid punishment.

To evaluate the performance of Snr (with $p_{def} = 20\%$), we again assume a reputation system with reliability $p_{rel} = 20\%$ and punishment $\tau = 5$. We vary the number of Snr nodes, and plot the cumulative distribution function (CDF) of the total payoffs of the Sn and Snr nodes. (See figure 10.)

The payoffs of the Snr nodes vary substantially, which indicates that they are not being consistently punished by the reputation system. However, the average payoff remains small; evidently the punishment is sufficiently severe that it wipes out any gains from the occasional defections.

VI. Discussion

In this section we discuss some open issues in our understanding of the routing game. We finish by describing some of the related work.

A. Open Issues in the Routing Game

1) Timing of Requests: As mentioned earlier, a node can cheat by sending its requests in batches, so that it only needs to maintain a good reputation for the time needed to send a single batch. This kind of "timing" attack would not be practical in some applications, since it delays the servicing of requests. One area of future work is to quantify this trade-off: How much delay must be incurred, in order to get significant savings with this strategy? What kinds of monitoring and punishments would be needed to prevent this sort of cheating?

One idea is to punish a guilty node by dropping a certain number of its requests, instead of dropping its requests for a period of time. This turns out not to work; a guilty node can "fake" its punishment by sending a string of worthless requests that it knows will be dropped. Time-based punishment still seems to be the most effective.

Another idea is to make the reputations "sticky," so that if a node is consistently good or consistently bad, then it takes a sustained change in behavior to cause a change in its reputation. This is a little bit like a monetary scheme, except that it doesn't do exact bookkeeping. The advantage of this scheme is that it punishes bad nodes that defect most of the time and cooperate only when they have a bunch of requests to make; but it tolerates good nodes that suffer from occasional bursts of noise. However, this scheme would be harder to implement in a practical system.

On a related note, our simple cost function (counting the number of requests) may not be realistic if the traffic is bursty. Five requests spread out over time may not incur the same cost as five requests in a single burst.

2) Tampering with Reputations: In a real network, reputations cannot be implemented entirely by a trusted third party. At the very least, one would probably have to rely on the nodes to report when their requests are dropped. This creates some difficult incentive problems: for instance, a self-interested node might falsely accuse other nodes, especially those who are using it as one of their fingers, so that it can drop their requests.

This is an example of a more general concern, that nodes may manipulate an untrusted reputation system as part of their strategy. Preventing these attacks may require a combination of incentives (to encourage nodes to report truthfully), and a reputation system that resists tampering by a single node or a small group of nodes.

3) Other Limitations: The social norm strategy works best when the network is more-or-less homogenous: all nodes send the same number of requests per round, and all nodes choose destinations uniformly at random. But there is nothing in our present scheme that limits the number of requests that a node can send; so long as it continues to forward other nodes'

requests, a node is free to send as many requests as it likes, thus maximizing its payoff. Our scheme works when all the nodes need to make roughly the same number of requests. In a highly heterogenous network, one may have to use some kind of monetary scheme instead, to obtain the right incentives.

Also, in the game, nodes are only allowed to choose between cooperating and defecting; whereas in real life, a node can do other things, such as forwarding a request incorrectly. And, in a real system, the utility function of each node may be more complicated than in our simple model.

Finally, in the routing game we do not consider collusion between nodes. But there is nothing that prevents a subset of the nodes from attempting to build their own overlay network on top of the peer-to-peer system. This could be a significant issue.

4) Retrying Dropped Requests: One could modify the routing game to allow a node to retry a request that has been dropped, or to give nodes greater freedom in choosing their fingers. This would make routing much more robust, which would benefit the innocent nodes. Guilty nodes would not be able to take advantage of this, because the reputation system causes them to lose their requests regardless of what route they choose.

We have not implemented these changes, because of their complexity. However, the fact that such improvements are possible suggests that our current simulation results are fairly conservative.

B. Related Work

There have been many studies of incentive and reputation systems in peer-to-peer networks; here we mention two papers that are similar in spirit to our work:

Lai et al [8] use the evolutionary Prisoner's Dilemma as a model for file sharing. They study strategies based on private and shared history, as well as strategies that "adapt" to the behavior of strangers.

Ranganathan et al [9] use the multiple-player Prisoner's Dilemma as a model for file sharing. They investigate two different reputation-based schemes and one monetary scheme.

Also, a number of systems have been built that incorporate incentives. Karma [10] is a peer-to-peer file sharing system that uses a monetary scheme. Each node has a bank account that is implemented by a set of the other nodes, called its "bank set." The system also has mechanisms to deal with pricing and inflation. This approach is flexible, but has a high performance overhead.

Samsara [11] is a distributed storage system, that requires each node to contribute as much disk space to the system as it is using. Claims to storage space can be traded, allowing the nodes to form asymmetric or transitive relationships.

Finally, Castro et al [12] studied a variety of attacks on peer-to-peer routing, and proposed some solutions using techniques in cryptography and security. Their approach is complementary to ours: we try to provide incentives for good behavior, while they seek to detect and prevent bad behavior.

VII. Conclusions

In this paper we used a random-matching game to model routing in peer-to-peer networks. We defined an analogue of Kandori's "social norm" strategy, which uses a simple reputation system to provide incentives for cooperation. Our simulation results show that this scheme is robust in the presence of malicious nodes and noise. Furthermore, we showed that an unreliable reputation system which monitors only a fraction of the routing events can still be effective, provided that the punishments are sufficiently severe. Although our model does not capture all aspects of a real network, we feel that it is a useful starting point for understanding the incentive problems that arise in peer-to-peer systems.

One area of future work is to develop more realistic games which model different aspects of peer-to-peer systems. Some of these issues, such as the timing of requests, have been discussed in this paper.

Another area of work is to implement reputation systems in real networks. A real reputation system cannot use a trusted third party, but must be distributed over the nodes themselves. It is a serious engineering challenge to build a reputation system that is secure against tampering, provides the proper incentives, and has good performance and scalability.

References

[1] M. Kandori. "Social Norms and Community Enforcement." *Review of Economic Studies*, Vol. 59, No. 1 (Jan. 1992), pp.63-80.

[2] J. Feigenbaum and S. Shenker. "Distributed Algorithmic Mechanism Design: Recent Results and Future Directions." *Proc. 6th Intl. Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, New York, 2002, pp.1-13.

[3] *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, 2003.

[4] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[5] Vincent Crawford, personal communication.

[6] Kuhn, Steven, "Prisoner's Dilemma", The Stanford Encyclopedia of Philosophy (Fall 2003 Edition), Edward N. Zalta (ed.), URL = http://plato.stanford.edu/archives/fall2003/entries/prisoner-dilemma/.

[7] I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan. "Chord: A Scalable Peer-To-Peer Lookup Service for Internet Applications." *Proc. ACM Sigcomm*, August 2001.

[8] K. Lai, M. Feldman, I. Stoica, J. Chuang, "Incentives for Cooperation in Peer-to-Peer Networks." *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, 2003.

[9] K. Ranganathan, M. Ripeanu, A. Sarin, I. Foster, "To Share or Not to Share: An Analysis of Incentives to Contribute in Collaborative File Sharing Environments." *Workshop on Economics of Peer-to-Peer Systems*, Berkeley, 2003.

[10] V. Vishnumurthy, S. Chandrakumar and E. Gun Sirer, "KARMA: A Secure Economic Framework for P2P Resource Sharing." *Workshop on the Economics of Peer-to-Peer Systems*, Berkeley, California, June 2003.

[11] L. Cox and B. Noble, "Samsara: Honor Among Thieves in Peer-to-Peer Storage", *Proc. SOSP 2003*, Lake George, NY, October, 2002.

[12] M. Castro, P. Druschel, A. Ganesh, A. Rowstron and D.S. Wallach. "Secure routing for structured peer-to-peer overlay networks." *Proc. OSDI 2002*, Boston, MA, December 2002.