Designing Incentives for Peer-to-Peer Routing

Alberto Blanc[1], Yi-Kai Liu[2], Amin Vahdat[3]
[1]Electrical and Computer Engineering Dept.
[2,3]Computer Science and Engineering Dept.
University of California, San Diego

***Abstract*— In a peer-to-peer network, nodes are typically required to route packets for each other. This leads to a problem of "free-loaders," nodes that use the network but refuse to route other nodes' packets. In this paper we study ways of designing incentives to discourage free-loading. We model the interactions between nodes as a "random matching game," and describe a simple reputation system that provides incentives for good behavior. Under certain assumptions, we obtain a stable subgame-perfect equilibrium. We use simulations to investigate the robustness of this scheme in the presence of noise and malicious nodes, and we examine some of the design trade-offs. We also evaluate some possible adversarial strategies, and discuss how our results might apply to real peer-to-peer systems.**

Keywords: Economics, system design.

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 good behavior. This paper investigates one such scheme, using tools from game theory. First, we model a generic peer-to-peer network as a *random-matching game*. This game was previously studied by Kandori, who showed that, using a simple reputation system, cooperation can be sustained as a stable subgame-perfect equilibrium [1]. We use simulations to measure the robustness of this scheme to malicious nodes and noise.

We then consider a modified version of the random matching game that models peer-to-peer routing. Using a simple reputation system, under certain assumptions, we again get a stable subgame-perfect equilibrium. We use simulations to measure the robustness of this equilibrium in the presence of malicious nodes and noise; we observe a trade-off between having strong incentives and tolerating noise. We also demonstrate that the reputation system can still be effective even if it only monitors a small fraction of the routing events. These results are encouraging, with implications for the design of practical systems.

Additionally, we use simulations to study some possible adversarial strategies, and some scenarios involving heterogenous populations of nodes. Finally, we discuss several attacks that could occur in a real peer-to-peer system, and we sketch out some ideas about how our reputation system might be implemented.

There has been a substantial amount of work on using incentives in computer science; for a recent survey, see [2]. Incentives for peer-to-peer networks have been studied in [3]. Also, 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 briefly describes the basic random-matching game and the equilibrium strategy; for more details, see the companion technical report [6]. Section III describes the peer-to-peer routing game, and section IV presents simulation results for that game. Section V discusses some open issues and related work. Section VI concludes the paper.

II. The Random Matching Game

A. The Prisoner's Dilemma

First, we review some basic notions from game theory. Consider a game with n players, labelled $1, 2, \ldots, n$. A *dominant strategy* for player i is a strategy which is optimal for player i, regardless of what the other players do. This is a desirable property, but it is difficult to achieve; in some games there are no dominant strategies. A *(Nash) equilibrium* is a set of strategies $(s_1, s_2, \ldots, s_n)$ such that, for each player i, if all the other players j follow their assigned strategies s_j, then strategy s_i is optimal for player i. This is weaker than the notion of dominant strategy, but it is easier to achieve. A special case is *subgame-perfect* equilibrium, where, at any time during the game, and regardless of what the players did prior to this moment, it is still an equilibrium for all the players to follow their assigned strategies from this time onward.

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

[1]E-mail: `alberto@cwc.ucsd.edu`.

[2]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.

[3]E-mail: `vahdat@cs.ucsd.edu`. Supported by NSF grant ITR-0082912, NSF CAREER award CCR-9984-328, Hewlett-Packard, IBM, Intel and Microsoft.

	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 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 providing a resource has some cost $c > 0$, and receiving a resource has some benefit $b > 0$, where $b > c$, so it is a positive-sum game. 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 [7] for a survey of work in this area.

B. The Random Matching Game

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. Indeed, the strategy described below does rely on a primitive reputation scheme. But Kandori also found another equilibrium strategy that does not use any kind of reputation information; instead, it relies on a global threat that any deviation from equilibrium will eventually cause the system to collapse. This equilibrium is unstable, since cooperation will break down after a single error or mistake by one of the players. So it is not suitable for a real system. This example illustrates some of the issues in applying game theory to a real-world 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. We assume the existence of 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.

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 continue playing the game for another round. It measures the "patience" of the players: 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 equilibrium is also stable, in the sense that, starting from any initial state, it will re-establish cooperation after τ rounds. The proof of the equilibrium can be found in [1].

D. Tolerating Malicious Nodes

We were able to extend Kandori's analysis to look at the effect of malicious nodes, i.e., nodes that always defect. Provided that we use uniform random matching, we find that the incentives and the global efficiency decrease gradually, as the fraction of malicious nodes increases. This work is described in the technical report [6].

E. Simulations of the Random Matching Game

We ran simulations to measure the effects of malicious nodes and noise in the random matching game. We found that the social norm equilibrium is robust to small fractions of malicious nodes and low levels of noise. In particular, there is a trade-off between using harsher punishments so as to tolerate malicious nodes, and using milder punishments to tolerate noise. We also tried to test the stability of the social norm in an evolutionary game, but got mixed results. These results are described in the technical report [6].

III. 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 [8] 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.

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 requests are generated according to some external process. The point is that nodes do not control the sending of requests, but can only decide whether to forward requests for 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 pays some cost c 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 t. If t cooperates, s receives a large payoff b, because its request has reached its destination; if t defects, s receives nothing. (If t cooperates, it pays the same cost c as in the previous cases.)

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.

Note that as the network grows, the number of hops per request slowly increases. In order for the incentives to work, the final payoff per request must also increase, to balance out the cost of routing. Since each request takes about $\lg(N)/2$ hops on average, we need at least $b > (\lg(N)/2)c$, assuming that the traffic is evenly distributed among the nodes. (That is, the benefit of sending a request must be at least $\lg(N)/2$ times the cost of routing a request.) In a real network, b might have to be somewhat higher, to account for uneven load balancing.

The size of the payoff b will depend on what a "request" actually means in a particular application. However, we think it is reasonable to assume that the cost c is small relative to b. Consider the following scenario: Each node is connected through a cable modem with uplink bandwidth of 256 Kb/sec, and it allocates 10% of this bandwidth to forward requests. For simplicity, we ignore the bandwidth used by the Chord stabilization protocol. We assume each request is 110 bytes long (this is the size of a single packet, including all headers, in one Chord implementation that we looked at). Then each node can forward about 29 requests/sec. If the network has 1024 nodes, then each request takes $\lg(1024)/2 = 5$ hops on average. So each node can send up to $29/5 \approx 6$ requests/sec, assuming the traffic is evenly distributed among the nodes; or, up to (say) 3 requests/sec, if we want to tolerate some uneven load balancing. This is adequate for many applications, and it suggests that each node can route a large number of requests for a fairly low cost.

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, 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 trusted authority which observes the nodes' actions and marks each node as innocent or guilty.

Finally, we need to specify what kinds of punishments will be enforced by the trusted authority. 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. As before, the equilibrium is stable in the sense that, starting from any initial state, cooperation will be re-established after time τ. 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, let $P_{[0,\tau]}$ be the payoff during some time interval $[0,\tau]$. Then

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

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.

For a guilty node, dropping a request delays its recovery (recall that punishment is measured in terms of time). This delay time makes all the difference between following the social norm, and deviating from it. The non-burstiness of the requests is crucial; this is why we assumed that requests were generated by a Poisson process. 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 at time T_1 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:

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

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.) Say the benefit and cost are $b = 40$ and $c = 2$. Then we can set:

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

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

IV. Simulations of the Routing Game

We ran simulations to measure the performance of the social norm strategy in the routing game. In particular, we observed: (1) how the presence of malicious nodes affects the incentive to cooperate; (2) how noise and errors hurt the global efficiency of the system; and (3) how an "unreliable" reputation system can still provide strong incentives. We also evaluated some possible "adversarial" strategies, and looked at some scenarios involving mixed populations of high- and low-rate nodes.

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 chose $c = 2$ as the cost of forwarding a request, and $b = 40$ as the payoff when a request reaches its destination. Since we did not have a specific application in mind, these values are somewhat arbitrary, but we think they are at least plausible. 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 $40 - 5.5 \cdot 2 = 29$.

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.

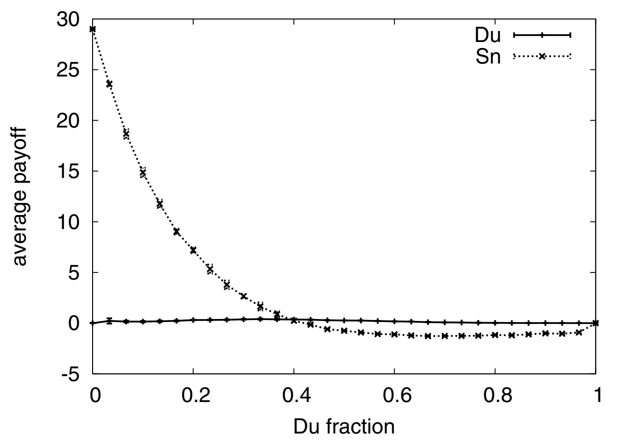


Fig. 1. Payoff per request for Sn and Du nodes, as the number of Du nodes increases (punishment $\tau = 2$)

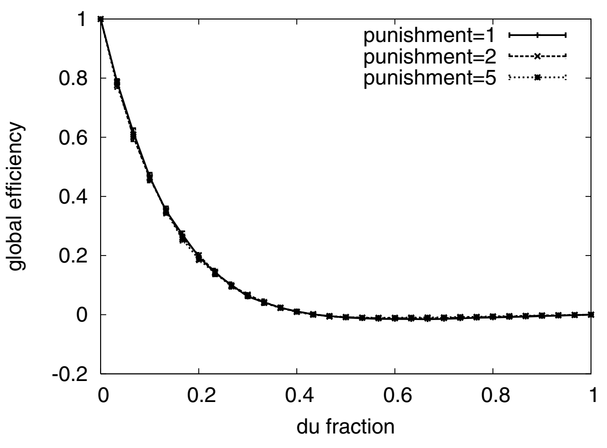


Fig. 2. Global efficiency as the number of Du nodes increases

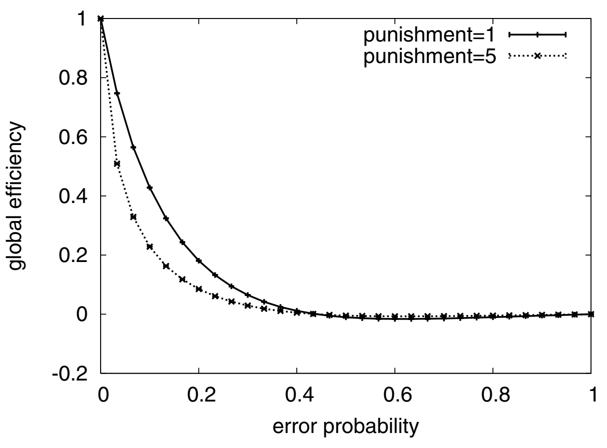


Fig. 3. Global efficiency with varying levels of noise

A. Simulation Results

1) Malicious Nodes: Because the social norm is only an equilibrium, not a dominant strategy, it does not guarantee that a given node will behave properly when some of the other nodes are faulty or malicious. Indeed, malicious nodes can cause the honest nodes to receive lower payoffs, thus weakening the incentive to cooperate. Here we try to measure this effect. We model malicious nodes as nodes that always defect, called "defect unconditionally" or "Du" nodes. Nodes that follow the social norm strategy are called "Sn" nodes.

Figure 1 shows the average payoff per request for the Sn and Du nodes, as the number of Du nodes increases. 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 2 shows the global efficiency as the number of Du nodes increases. (We define the global efficiency to be the total payoff of all the nodes, divided by the total payoff in the ideal case of 100% cooperation, no noise, etc.) The results are similar with punishment periods $\tau = 1$, $\tau = 2$ and $\tau = 5$.

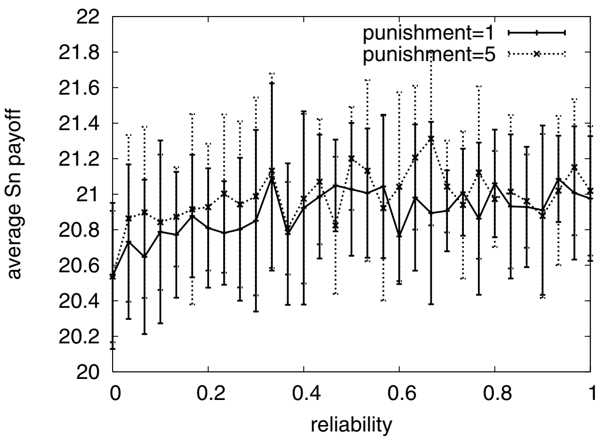
2) Noise: We next consider the effect of noise or errors in the system. Noise is harmful both because it causes requests to get dropped, and because it triggers additional punishment. We let p_{noise} be the probability that a node who tries to cooperate will end up defecting instead (if, for instance, the request gets dropped due to a network failure). In these simulations, we use a network of all Sn nodes.

Figure 3 shows the global efficiency as the noise level rises. Observe that the efficiency remains higher when we use the shorter punishment period. This illustrates a trade-off in choosing the punishment: longer punishments create stronger incentives, but they reduce the system's ability to tolerate noise.

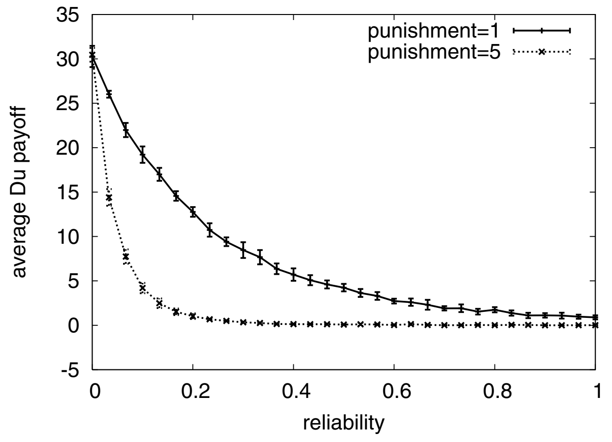
3) An "Unreliable" Reputation System: Consider a reputation system that is unreliable in the following sense: If a node behaves properly, its reputation will always be updated correctly; but if a node misbehaves, the incident will only be detected with probability p_{rel}. This would describe a reputation system that only observes a random sample of the activity in the network.

We want to see whether an unreliable reputation system, combined with a sufficiently severe punishment, can provide adequate incentives. Here we simulate a network with a 10% fraction of Du nodes. In figure 4 we plot the average payoff per request for the Sn and Du nodes, varying the reliability and using different punishment lengths.

Observe that the payoffs of the Sn nodes stay roughly constant, while the payoffs of the Du nodes drop rapidly as the reliability increases from 0. The reputation system is effective even when the reliability is quite low, partly because our punishments are strong, and 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 10% reliable when using a 1-round punishment,

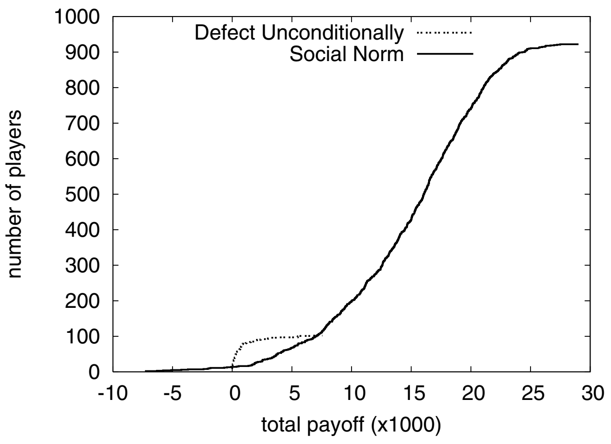


(a) Sn nodes

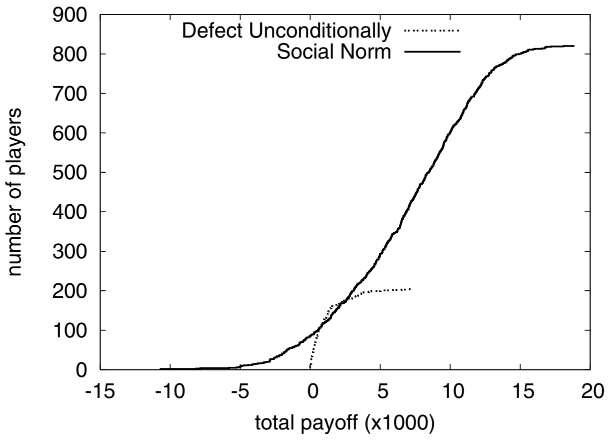


(b) Du nodes

Fig. 4. Payoff per request for Sn and Du nodes, varying the reliability of the reputation scheme. Population is 10% Du nodes.



(a) 10% Du nodes



(b) 20% Du nodes

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

and at least 5% reliable when using a 5-round punishment.

4) Distribution of Payoffs among the Nodes: To gain further insight into the effectiveness of an unreliable reputation system, we look at the distribution of the payoffs among the nodes. We assume a reputation system with fairly low reliability ($p_{\text{rel}} = 20\%$), 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 5).

The payoffs of the Sn nodes have a fairly large variance. 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. Also, an Sn node whose longest finger happens to be a Du node will do very poorly, as half of its requests will be dropped.

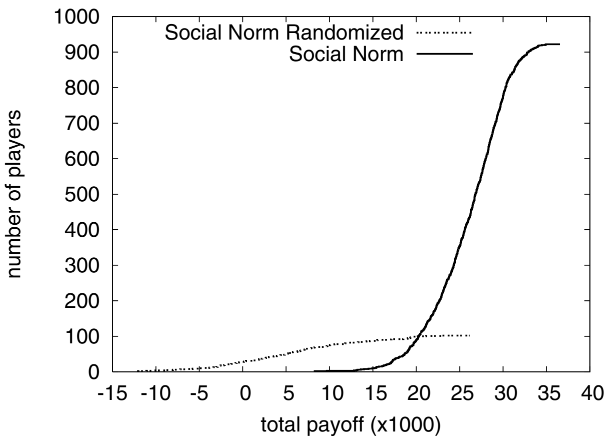
The payoffs of the Du nodes, on the other hand, are concentrated close to zero. (Note that, unlike Sn nodes, Du nodes never have negative payoffs.) This shows that the reputation system is effective.

B. Other Strategies

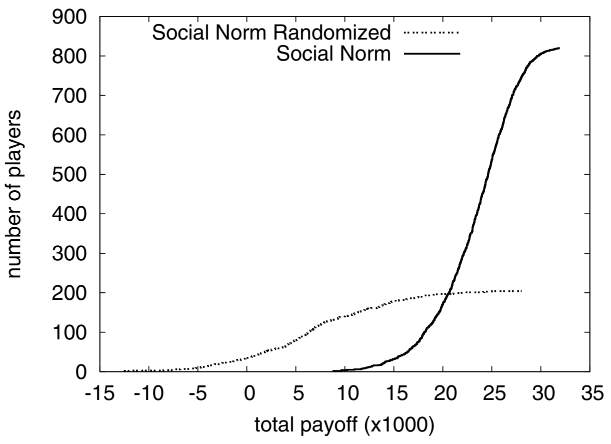
1) Social Norm with Random Defections: Another possible strategy is the social norm with random defections, denoted Snr:

- If the opponent is innocent, then cooperate with probability $1 - p_{\text{def}}$ and defect with probability p_{def}; if the opponent is guilty, then defect.

Note that setting $p_{\text{def}} = 0$ gives the Sn strategy, while setting $p_{\text{def}} = 1$ gives the Du strategy. Typically we would choose an intermediate value, say $p_{\text{def}} = 0.2$. The intuition here is that an unreliable reputation system will have trouble detecting occasional defections. So, although Snr is less aggressive than Du, it may do better because it can avoid punishment.

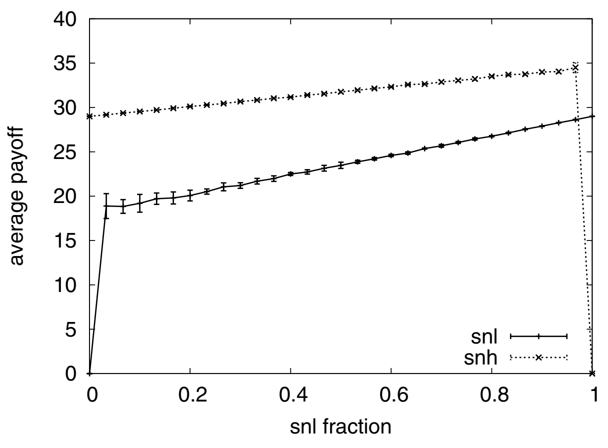


(a) 10% Snr nodes

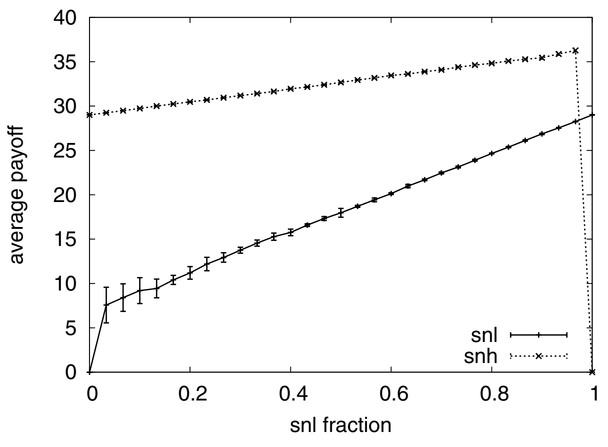


(b) 20% Snr nodes

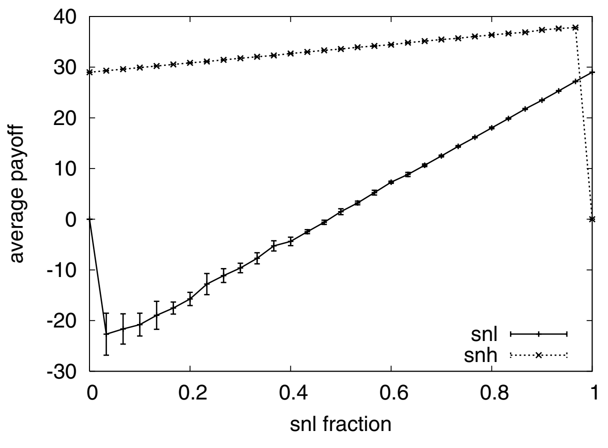
Fig. 6. CDF of the total payoffs of the Sn and Snr nodes (with $p_{\text{def}} = 20\%$), with reliability $p_{\text{rel}} = 20\%$ and punishment $\tau = 5$



(a) $p_{\text{high}} = 0.9$, $p_{\text{low}} = 0.45$



(b) $p_{\text{high}} = 0.9$, $p_{\text{low}} = 0.3$



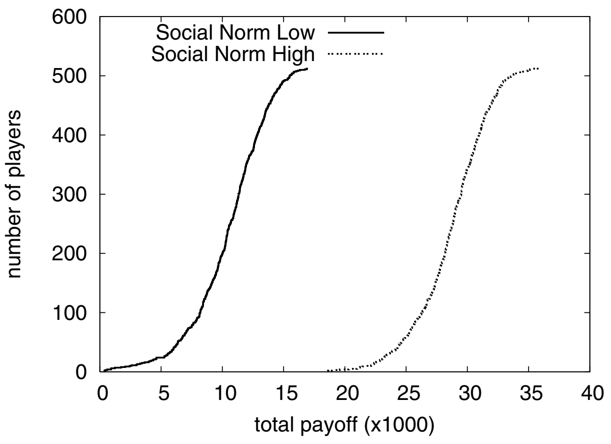
(c) $p_{\text{high}} = 0.9$, $p_{\text{low}} = 0.15$

Fig. 7. Payoff per request for the Sn-High and Sn-Low nodes, varying the numbers of Sn-High and Sn-Low nodes

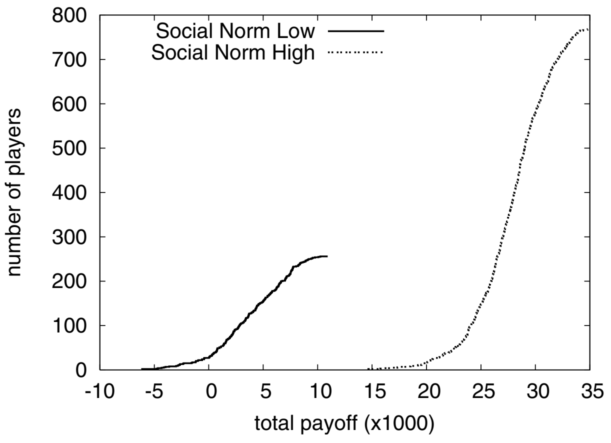
To evaluate the performance of Snr (with $p_{\text{def}} = 20\%$), we again assume a reputation system with reliability $p_{\text{rel}} = 20\%$ and punishment $\tau = 5$. We vary the number of Snr nodes, and plot the cumulative distribution function (CDF) of the total payoffs of the Sn and Snr nodes. (See figure 6.)

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.

2) Mixed Population of High- and Low-Rate Nodes: Thus far we have assumed that all nodes send requests at the same rate. But a real network may be heterogenous, with some nodes sending more requests than others. These nodes will receive higher payoffs, while imposing additional costs on their neighbors. In extreme cases, this can cause other nodes

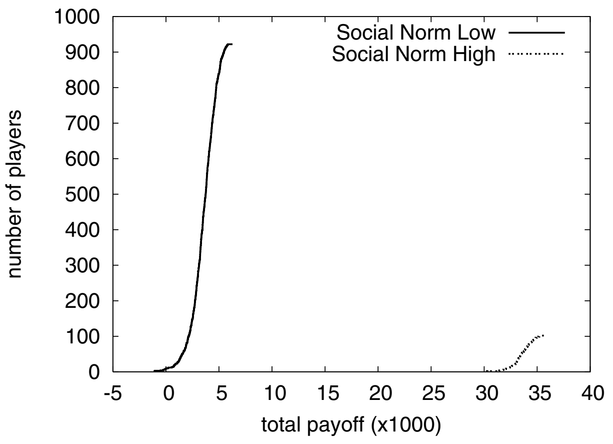


(a) 50% Sn-High nodes, 50% Sn-Low nodes, $p_{\text{high}} = 0.9$, $p_{\text{low}} = 0.45$

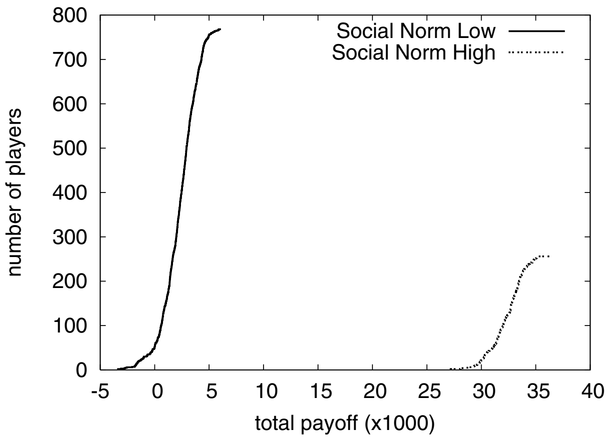


(b) 75% Sn-High nodes, 25% Sn-Low nodes, $p_{\text{high}} = 0.9$, $p_{\text{low}} = 0.3$

Fig. 8. CDF of the total payoffs of the Sn-High and Sn-Low nodes, for some specific cases



(a) 10% Sn-High nodes, 90% Sn-Low nodes, $p_{\text{high}} = 0.9$, $p_{\text{low}} = 0.15$



(b) 25% Sn-High nodes, 75% Sn-Low nodes, $p_{\text{high}} = 0.9$, $p_{\text{low}} = 0.15$

Fig. 9. CDF of the total payoffs of the Sn-High and Sn-Low nodes, for small numbers of "adversarial" Sn-High nodes

to get negative payoffs, destroying their incentive to cooperate. This limits the amount of heterogeneity that can be tolerated. We ran some simulations to measure this effect.

We considered a simple scenario with two classes of nodes: high-rate nodes (denoted Sn-High) and low-rate nodes (Sn-Low). Both of these classes practice the social norm, but in each round an Sn-High node makes a request with probability p_{high}, while an Sn-Low node makes a request with probability p_{low}.

Figure 7 shows the average payoff per request for the Sn-High and Sn-Low nodes, using various values of p_{high} and p_{low}. The population consists of a fraction f of Sn-Low nodes and a fraction $1 - f$ of Sn-High nodes, and we vary f along the horizontal axis. Note that the reputation system does not play any role, since all the nodes are following the social norm.

As we increase the number of Sn-Low nodes, the payoff per request increases for everyone; this is because each node now has to forward fewer requests for the other nodes. As for the incentives of the Sn-Low nodes, problems arise when there are very few Sn-Low nodes, and they send requests at a very low rate. In extreme cases, the payoffs of the Sn-Low nodes become negative.

Figure 8 shows the distribution of the total payoffs among the nodes, for some specific scenarios. The two classes are clearly visible. Note that some Sn-Low nodes in the lower tail of the distribution have negative payoffs; these are the "unlucky" nodes that were chosen to be the fingers of the Sn-High nodes. Because of the finger tables, the cost of an Sn-High node is not spread equally over the network, but falls disproportionately on a small number of neighboring nodes.

Issue:	Possible strategy:
Reputation system does not track resource usage	Batch requests, Sn-High
Sn is not a dominant strategy	Adaptive strategies
Untrusted reputations	Reputation tampering
Implementation details	Various hacks?
Outside communication, collusion	Sybil attack, clubs, coordinated behavior?

Fig. 10. Some open issues and possible adversarial strategies

3) Sn-High as an Adversarial Strategy: Our reputation system does not limit the number of requests that a node can send; as 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. Thus, Sn-High, sending requests at very high rates, can be viewed as an adversarial strategy. This may not be completely realistic, since in many applications a node only needs to make a certain number of requests, and cannot benefit more from sending additional requests; we will discuss this later.

We ran some simulations to measure the effect of a small number of "adversarial" Sn-High nodes on the network (figure 9). Here we set $p_{\text{high}} = 0.9$ to represent the high-rate nodes, and $p_{\text{low}} = 0.15$ to represent the "ordinary" nodes. Although the Sn-High nodes are sending requests at a very high rate, there are relatively few of them, and we find that they do not dominate the behavior of the network. However, they do have a major impact on the small set of Sn-Low nodes that serve as their fingers. This can be seen in the lower tail of the distribution, which has negative payoffs. It appears that our incentives still work when Sn-High nodes make up perhaps 10% of the population, but they fail sometime after that point.

V. Discussion

In this section we discuss some open issues regarding the routing game and the social norm strategy. We describe some attacks that could be carried out in a real system, and some solutions. Note that many of our remarks apply to general peer-to-peer systems, not just peer-to-peer routing. These issues are summarized in Figure 10. Finally, we sketch some different approaches to implementing a real reputation system, and we discuss some related work.

A. Open Issues in the Routing Game / Social Norm

1) Reputation System Does Not Track Resource Usage: A basic characteristic of our reputation system is that it does not keep track of how many requests each node sends. This is different from a monetary scheme, where each node pays for every request it sends. Intuitively, this means that our reputation system works best when the network is homogenous: all nodes send requests at the same, steady rate, and they all choose destinations uniformly at random. If these conditions do not hold, our reputation system can fail, and one may have to use a monetary scheme to obtain the desired incentives.

On the other hand, our reputation system does have an advantage over a monetary scheme: it only has to keep track of the guilty nodes, and only requires action when a request gets dropped; whereas a monetary scheme must keep track of all the nodes, and requires action each time a request is forwarded. So our reputation system may be faster in the common case (assuming most nodes are honest). This illustrates what seems to be a trade-off between the quality of the incentives and the amount of communication overhead. Here we will focus on the incentives; we will revisit the performance issues in section V-B.

We describe two attacks that exploit the above weakness in our reputation system. First, a node can cheat by sending its requests in batches, so that it only needs to maintain a good reputation when it is sending a batch. Note that this attack would not work in some applications, since it delays the servicing of requests.

One possible solution is to punish a guilty node by dropping a certain number of its requests. But this does not work; a guilty node can "fake" its punishment by sending a string of worthless requests to be dropped. Time-based punishment still seems to be the most effective.

A better 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, but without exact bookkeeping. This scheme is attractive because it punishes bad nodes that cooperate only when they have a batch of requests to send, and it tolerates good nodes that suffer from occasional bursts of noise. However, this scheme could be harder to implement, since the reputation system would have to maintain more state information for each node.

The second attack is even simpler: a node can increase its payoff simply by sending more requests (the Sn-High strategy). However, this attack is constrained by the fact that, in many real applications, each node only has a certain number of requests to send, and does not benefit by sending more than this number.

2) Social Norm is Not a Dominant Strategy: Recall that the social norm is only a (subgame-perfect) equilibrium, not a dominant strategy, and so it may not be in a node's best interest to follow the social norm if other nodes are deviating from it. In other words, misbehaving nodes can destroy their neighbors' incentives to cooperate.

This leads us to consider adaptive strategies, where a node observes the behavior of its neighbors, and chooses its own actions accordingly. For instance, a node might initially follow the social norm, but switch to Du (always defect) if one of its fingers turns out to be a Du node, or if it ends up being a finger of an Sn-High node.

In practice, however, this may not be such a serious problem. In a real peer-to-peer network, the same mechanisms that improve performance and reliability also help prevent misbehaving nodes from destroying the incentives of honest nodes. For example, load-balancing mechanisms distribute the burden of an Sn-High node over many other nodes, while failure recovery mechanisms allow good nodes to bypass or route around a Du node.

3) Untrusted Reputations: In a real network, reputations cannot be implemented entirely by a trusted third party. At the very least, one would have to rely on the nodes themselves to report when their requests are dropped. This creates incentive problems. For instance, one node might falsely accuse those nodes who are using it as a finger, so that it can drop their requests. Also, a group of nodes might collude to produce many independent reports against the same node. If the reputation system is not secure, nodes might even try to tamper with their reputations directly.

We think these problems can be solved using a combination of incentives (to encourage nodes to report truthfully), and a reputation system that resists tampering by small groups of nodes. We will discuss this further in section V-B.

4) Other Limitations: There are many details involving the implementation of a peer-to-peer network that are not modeled by our game. We assumed a very simple network topology and traffic model. We did not model nodes joining and leaving the network. 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.

5) Outside Communication: Finally, we did not consider how nodes might exploit information or communication channels that lie "outside" of the game. A well-known example is the Sybil attack, where a single entity controls multiple nodes on the network [9]. Alternatively, multiple entities could join together to form a "club," which controls a single node on the network. These techniques could be used for load-balancing, by splitting up high-rate nodes and joining together low-rate nodes to make the network more homogenous. But they can also be abused, for instance if several medium-rate nodes join together to form a high-rate node that floods the network with requests.

More generally, nodes might use outside communication to organize a collusion, or they might build their own overlay network on top of the peer-to-peer system. This kind of behavior is, almost by definition, hard to regulate, and so this could be a very serious issue.

B. Implementing a Reputation System

In this paper we have studied an abstract model of a reputation system, which uses a trusted third party. This model is convenient for analyzing incentives. We now sketch out some possible approaches to implementing this model in a real peer-to-peer system.

1) Central Authority: The first approach is to simply build a central authority which runs the reputation system. This is the most direct approach, and it avoids many of the difficulties with managing trust in a distributed system. Like any centralized solution, this approach may have problems with performance and scalability; but we think these problems are not as severe as many people assume. Here we present some ideas for improving performance and scalability.

The first task of the reputation system is to observe activity in the network. As mentioned earlier, the central authority would have to rely on the nodes to report when their requests get dropped. Since bad behavior is relatively infrequent, this should not consume too much bandwidth.

The problem of nodes submitting false reports can be dealt with in two ways. One approach is to partition the network in such a way that, whenever a node has an opportunity to accuse another node of misbehavior, it has no ulterior motive to do so [10]. Another approach is to require multiple independent reports before marking a node guilty; this protects against tampering by individual nodes or small coalitions. One can easily devise more elaborate rules along these lines to judge whether a node is guilty.

The second task of the reputation system is to disseminate reputation information to the nodes. We propose using a "blacklist," which is simply a list of the guilty nodes, and thus should be quite short. Observe that it is not necessary to distribute the blacklist to all the nodes: when a guilty node sends a request, the request takes multiple hops, and we only need to ensure that the request gets dropped on one of those hops. So, for instance, one could send the blacklist to a random subset of the nodes. Alternatively, one could exploit the topology of the network, and send the blacklist to the fingers of the guilty nodes. Yet another idea is to make use of the Chord stabilization protocol to eject guilty nodes from the Chord ring. These ideas have not been fully explored.

Finally, we point out that dropping requests is a good way to apply punishment, even in the context of incentive problems that have nothing to do with routing. For instance, in a file-sharing application, it may be easiest to drop requests while they are being forwarded through the network, rather than trying to deny those requests at their destinations. Because routing requires multiple hops, it can be easily disrupted, which provides a natural opportunity to carry out punishment.

2) Distributed Implementations: For distributed reputation systems, scalability is less of a concern, while trust management becomes a serious problem: how to build a trustworthy reputation system, when the individual nodes cannot be trusted? One approach is to replicate the reputation system over a set of independent nodes, the majority of which should be honest. This approach is used by Karma [14], described in the next section. Essentially, this amounts to using sets of nodes to simulate a trusted authority. It works, but it has significant complexity and performance overhead.

Another approach is to discard the notion of "reputation" in favor of some other mechanism to provide incentives. For example, Samsara [15] and Sharp [16], described below, are based on bartering economies. One could also imagine more exotic systems, where there are no well-defined reputations, but individual nodes can organize themselves to carry out large-scale punishment against misbehaving nodes. These are interesting ideas, and they could have good performance and scalability. Because they work so differently, however, they are not directly comparable to our simple reputation system.

C. Related Work

A preliminary version of this paper appeared in [11]. It had some simulations of malicious nodes and noise, but not the proof of equilibrium and not the simulations of adversarial strategies.

There have been many studies of peer-to-peer networks using game theory. Lai et al [12] use the evolutionary Prisoner's Dilemma as a model for file sharing. They study strategies based on private and shared history, and strategies that adapt to the behavior of strangers. Ranganathan et al [13] use the multiple-player Prisoner's Dilemma as a model for file sharing. They investigate some different reputation and monetary schemes.

Also, a number of systems that use incentives have been proposed. Karma [14] is a monetary scheme, used for peer-to-peer file sharing. Each node has a bank account, which is implemented by a set of the other nodes, called its "bank set." This approach is flexible and decentralized, but has substantial overhead. As far as we know, Karma was never actually implemented and tested.

Samsara [15] is a peer-to-peer storage system, that requires each node to contribute as much disk space to the system as it is using. Samsara allows "claims" to storage space, which are incompressible blocks of data. Claims can be traded, allowing the nodes to form asymmetric or transitive relationships. This solution requires no central authority, but it only works for storage applications.

Sharp [16] is a system for distributed resource management over the Internet. It allows trading of computational resources, which leads to a barter economy, without any central authority. However, Sharp is designed to manage resources at the level of autonomous systems, and is less suitable for peer-to-peer networks. In particular, Sharp assumes long-lived relationships among the nodes.

Finally, Castro et al [17] 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: they focus on constructing "hard" defenses against malicious nodes, whereas we focus on providing "soft" incentives for rational, self-interested nodes.

VI. 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. Under certain assumptions, this strategy is a stable subgame-perfect equilibrium. Our simulation results showed that this scheme is robust in the presence of malicious nodes and noise (though there is a tradeoff between having strong incentives and tolerating noise). We also 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. In addition, we ran simulations of some possible adversarial strategies. Finally, we described some attacks that could occur in a real system, and suggested some ideas for how our reputation system might be implemented in practice.

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 routing. One area of future work is to develop more realistic games that model different aspects of peer-to-peer systems. We also need to understand what exactly we are trying to accomplish: what kinds of attacks should we worry about, how rational are the users of peer-to-peer systems, and so forth.

Another area of work is to explore different approaches to implementing real reputation systems for peer-to-peer networks. We do not have much experience in this area. There are difficult tradeoffs in building a reputation system that is secure, provides the right incentives, and has good performance and scalability.

Acknowledgements: We thank the anonymous reviewers for their helpful and insightful comments.

References

[1] M. Kandori. "Social Norms and Community Enforcement." *Rev. 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, and Harvard, 2004.

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

[5] Vincent Crawford, personal communication.

[6] A. Blanc, Y. Liu, A. Vahdat. "Designing Incentives for Peer-to-Peer Routing." Technical report, available at http://www.cs.ucsd.edu/ vahdat/papers/routing-game-tr.pdf.

[7] 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/.

[8] 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.

[9] J.R. Douceur. "The Sybil Attack." IPTPS 2002.

[10] Z. Abrams, R. McGrew, S. Plotkin. "Keeping Peers Honest in EigenTrust." *P2PEcon 2004* [3].

[11] A. Blanc, Y. Liu, A. Vahdat. "Designing Incentives for Peer-to-Peer Routing." *P2PEcon 2004* [3].

[12] K. Lai, M. Feldman, I. Stoica, J. Chuang, "Incentives for Cooperation in Peer-to-Peer Networks." *P2PEcon 2003* [3].

[13] 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." *P2PEcon 2003* [3].

[14] V. Vishnumurthy, S. Chandrakumar and E. Gun Sirer, "KARMA: A Secure Economic Framework for P2P Resource Sharing." *P2PEcon 2003* [3].

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

[16] Y. Fu, J. Chase, B. Chun, S. Schwab, A. Vahdat. "SHARP: An Architecture for Secure Resource Peering." *SOSP 2003.*

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