International Journal of Advanced Science and Technology
Vol. 15, February, 2010

Using Incentives for Heterogeneous peer-to-peer Network

Maricel O. Balitanas and Taihoon Kim

Hannam University, Department of Multimedia Engineering, Postfach, 306 791
jhe_c1756@yahoo.com, taihoonn@empal.com

Abstract

Typically in a peer-to-peer network, nodes are required to route packets for each other. This entails to a problem of "free-loaders," nodes that use the network but refuse to route other nodes' packets. In this paper we presented designing incentives to discourage free-loading.

Keywords: *Heterogeneous Network, Incentive Meachanism,*

1. Introduction

The next generation network is a heterogeneous network, with different component networks which may be wired and wireless, each with very different characteristics in terms of transmission speeds, errors, and interference tolerance. P**2P networking** has generated tremendous interest worldwide among both Internet surfers and computer networking professionals. **P2P software** systems like Kazaa and Napster rank amongst the most popular software applications ever. Numerous businesses and Web sites have promoted "peer to peer" technology as the future of Internet networking. Although they have actually existed for many years, P2P technologies promise to radically change the future of networking. P2P file sharing software has also created much controversy over legality and "fair use." In general, experts disagree on various details of P2P and precisely how it will evolve in the future.

A. Related Work on Resource Discovery

Existing approaches on resource discovery can be categorized into three different groups: centralized, decentralized, and flooding [8]. In the centralized approach, a peer queries the centralized server for the requested resource. It then requests the resource from the peers reported as having the resource. This has been the original design of Napster [6]. The approach suffers from a single point of failure and performance bottleneck as all queries for resource discovery have to go through the only centralized server before any requested resource is located. In the decentralized approach, directory information is maintained at several servers or peers. In KaZaA [4], each peer is either a supernode or an ordinary node. Each supernode acts as a mini-Napster hub so as to keep track of the addresses and the resources maintained by its descendants. A peer first sends a query to its supernode, which responds with matches or forwards the query to a subnet of supernodes repeatedly until the requested resource is located. This technique may fail when a supernode leaves the system as each ordinary node is generally associated with a single supernode. The dynamic join or leave of a peer is, however, quite common in the heterogeneous environment. Distributed indexing service based on distributed hash tables has been proposed to facilitate the location of contents [9]. Peers and contents are mapped to a key space via a hash function. Thus, this method requires that peers and content indices are organized in a rigid structure according to their keys. This may not be practical in heterogeneous P2P networks since many peers may be mobile and join/leave the network unpredictably. In the flooding approach, a peer broadcasts its query to all of its neighbours. If the queried peer has the requested resource, it sends back a response to the originating peer through the reverse path. Otherwise, it forwards the query to its neighbors and the process repeats. This technique may generate excessive query traffic in the network and thus is not scalable with the number of peers. In [ ], limited scope query is applied to enhance scalability by limiting the number of neighbors repeating a broadcast and only to those within some given hops from the originating peer. However, these limitations may render the requested resource unallocated even if it is present in the system. As far as we know, the existing approaches do not take various characteristics of the heterogeneous P2P networks into account for resource discovery. First, they do not provide any mechanisms to handle dynamic join/leave of peers which is common in heterogeneous P2P networks. Second, the placement of directory information and the construction of the directory overlay do not take into account of the heterogeneity of the P2P networks. For example, it may not be beneficial to place the directory information at a mobile peer. Third, there is no mechanism to perform traffic analysis and caching of the directory information of some popular resources so as to optimize the network performance and the user response time for subsequent queries.

B. Related Work on Content Distribution

There have been a number of large-scale deployments of peer-to- peer content sharing systems in the Internet. BT [10] is perhaps the best known. A shared content is partitioned into multiple small chunks. Peers exchange information on which chunks they currently possess, and request missing chunks from others. A peer can maximize its download speed by requesting different chunks from different peers at the same time. A peer selection mechanism is employed to find to whom such a request should be made. In addition, a chunk selection technique is needed for a peer to decide which chunk(s) to request. Indeed, peer selection and chunk selection are two crucial issues that affect the global effectiveness of the content distribution process [1]. Existing approaches on peer selection [3, 14, 6 , 7] rely on the estimates of the round-trip times between peer pairs and peer response times. They use some bandwidth probing mechanisms to optimize the selection of "best" peers. In [15], the "best" peer is selected by incorporating the path length and transmission power constraints in wireless networks. Since the estimates are generally updated periodically, performance suffers. Furthermore, requests are only forwarded to the "best" peer based on these estimates, which do not favor spreading out the requests across a set of peers, thereby causing further performance degradation due to load imbalance.

In [16], we proposed a generalized application-layer any casting protocol, known as par casting, to advocate concurrent access of a subset of replicated servers to cooperatively satisfy a client's request. Each participating server satisfies the request in part by transmitting a subset of the requested content to the client. An analytical model was developed to study how to effectively download a shared content from a set of replicated servers. However, the study assumes that all peers are homogeneous. In addition, it focuses on the selection of a subset of peers for content downloading, but does not consider the redistribution of any downloaded chunks to other peers.

In [17], three different replication strategies based on size, bandwidth, or display time of the multimedia clips in wireless peer-to-peer networks have been investigated. The goal is to determine the number of replicas needed for each clip so as to enable all home-to-home online (H2O) devices to support continuous display of a clip. However, only the string network topology has been considered in the analysis. Besides, the effect of channel interference among these H2O devices was ignored. Hence, these replication strategies are not directly applicable to heterogeneous P2 networks with conflict-prone wireless links. The problem of broadcasting or multicasting a shared content as a single chunk in heterogeneous networks has been investigated in [18, 19]. It has been shown that the problem of finding an optimal broadcast schedule that minimizes the maximum completion time is NP-complete. In particular, the problem in [18] has been formulated in a network where peers have different processing times and the transmission times between different peer pairs also vary. It is shown that the

fastest node first (FNF) algorithm is worse than the optimal by an unbounded factor. Heuristic algorithms, such as fastest edge first (FEF) and earliest completing edge first (ECEF), have also been developed. [19] studies the problem in a network in which the transmission times of a peer to all its neighboring peers are assumed to be the same. It proves that FNF gives a 1.5-approximation schedule. It is worth noting that [18, 19] only analyze the problem of distributing just one chunk, whereas a shared content is generally divided into multiple chunks in peer-to-peer content sharing systems. In addition, [18, 19] also assume that the transmission times are known a priori and transmission errors are ignored. These assumptions are rather unrealistic in heterogeneous P2P networks with error-prone wireless links. BT employs the rarest element first (REF) algorithm, in which those chunks lacking in the most peers are downloaded first. REF is good at increasing the availability of different chunks in the network and is efficient at distributing all chunks from the original source to different peers across the network. However, the simulation results in [20] show that the performance of REF is far from optimal since REF does not take into account which peer(s) should have higher priorities as recipients based on their needs. Instead of finding the "best" peer to fulfill a peer's request, concurrent access to a set of peers for satisfying requests, also known as parallel downloading, has been proposed [21, 22]. The general idea is to satisfy a request by the involvement of all available peers. A peer may download a certain chunk from each of the available peers, where the decisions of how much and which chunk to be downloaded from which peer are determined before the download begins [21]. Alternatively, a shared content may be partitioned into a large number of smaller equal-sized chunks and chunk-based requests are forwarded to all peers until all chunks are received [22].

However, as discussed in [22], issues such as the conditions under which it is beneficial to apply the proposed dynamic parallel access and the optimal/preferred chunk size are still unresolved. Furthermore, the methods described in [21, 22] do not consider any redistribution of any downloaded chunks to other peers. In [20], we have investigated the collaborative file distribution problem in a homogeneous peer-to-peer network, where all peers have asymmetric upload and download bandwidths and the transmission times between any peers are the same. Several heuristic algorithms were developed to solve the problem. The performance of these algorithms was also studied. However, [20] focuses on wired peer-to-peer networks and assumes the same transmission time of a chunk on all logical links in the network. The scheduling process for data distribution is carried out in discrete cycles synchronously. As far as we know, none of the existing work accounts for the broadcasting nature and the error characteristics of the wireless media. In addition, most existing work, such as the "rarest element first" algorithm in BT, are "greedy" algorithms which attempt to optimize the objective in one step, but does not account for the effect of this short-term gain on subsequent steps.

C. Related Work on Testbed

There are many types of network testbeds. IBM has developed a wireless sensor network testbed with Bluetooth, WLAN and Zigbee modules to evaluate wireless mesh networking for range extension and reliability enhancement [2]. To access the diverse kinds of networks, multiple radio modules are equipped for each node, while for a single kind of network, only one radio module is deployed. [24-25] use wired cables to connect individual devices or equipments to construct a multiradio platform. In [24], Intel x86-based notebook computers and FALINUX EX-X5 + EXPCMCIA boards were used to implement mesh routers. [25] combines two Soekris net4801 stations with a cross-over cable to form a single logical mesh point to implement dual-radio mesh relay points. Software defined radio (SDR) device is a flexible solution for wireless node. However, some SDR devices have to be driven by computers, thus confining their flexibility in deployment. Independently operated SDR devices are available but tend to be expensive for a testbed of reasonable size [26]. In this work, the proposed portable and flexible testbed with 0 multi-radio wireless nodes can form a highly integrated and cost-effective heterogeneous P2P testbed for various operations. In this paper, we study peer-to-peer (P2P) applications in a heterogeneous wired/wireless environment, and describe the development of a testbed to evaluate two of the key P2P protocols, namely, the resource discovery and the content distribution protocols. In Section II, we shall describe our testbed. Section III describes novel protocols that will be developed and tested on the testbed, and we conclude in Section IV.

2. Heterogeneous Search Principles

What's the most essential property of resource discovery in unstructured P2P networks? We believe it's randomicity. Before performing a search, we don't know where the targets are, whether they exist, or which technique is the most suitable for locating them. This implies that no search technique applies to all situations. Some methods show powerful discovery capabilities but are more costly; others are cost-efficient but show varying performance. A specific technique might show better performance in some peers and worse in others during a single retrieving instance, or it might be efficient in exact-matching retrieving and inefficient in wildcard or rich searches. Therefore, to avoid worsening or unstable search performance, it might be rational in these situations to combine multiple search techniques among peers in some way.

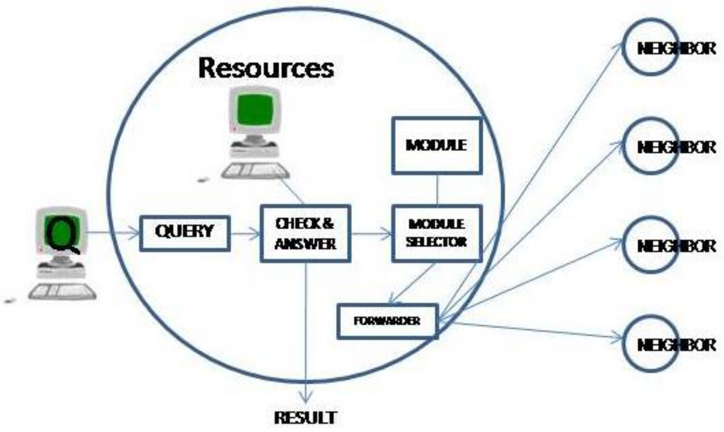

Figure 1. Internal structure of a peer using heterogeneous search (HES)

The process begins with a request message or query (Q). The Red-killer then filters out worthless queries. The Check & Answer parses Q with resources and forwards unfulfilled queries.

3. Mechanism Design for Peer-to-Peer Networks

3.1. Incentive Mechanism Design

Mechanism design has to deal with every configuration of agents' utility functions. For tractability, we can assume there is a known set of all possible utility functions, and each agent will be assigned a type based on which functions it is currently using. Knowing the type of other agents can improve one's decision process which in turns provides more possibilities for one to reach a better choice. However, it may not be beneficial for an agent to reveal its own type, or worse, an agent may be better off if it lies about its type. A mechanism is considered to be incentive compatible if agents have incentive to reveal their types. In other words, the best strategy for any agents in an incentive compatible system is to reveal their true identity and follow the protocol.

There has been various attempts to design a P2P system that is incentive compatible [4], [5], [6], [7], [8]. Among them, BitTorrent [9] is one of the most popular P2P-based file distribution systems that is incentive-compatible. It uses a "tit-for-tat" mechanism to reward peers that contribute more of their capacity in assisting the distribution process. However, these protocols either suffer from a high cost of reputation management, or a course granularity in classifying client types. In [10] and [11], we have proposed an incentive compatible mechanism for neighbor-selection that does not require maintenance of reputation, and a "seeing-is-believing" resource sharing mechanism where the best strategy for clients is to comply with the protocol and contribute as much as they can.

3.2. Mechanism with human intervention

It is until recently that stocks and other derivatives trading have been deploying automated trading agents to represent the clients. These agents are programmed with client-defined strategies, and react to the market based on their prediction or other modeling outcome. The upsides of these agents include fast responses (to a short arbitrage opportunity, for example) and do not make emotional decisions. An agent will follow its strategy regardless of how many times it has failed in the past, but the same thing may not be said about human traders. Even though trading agents have these advantages, they are also incapable to make adjustments1 or take into account other features that are not described in their model. For large investment banks and hedge funds, trading without the supervision of human being is very risky. As opposed to a completely automated black-box mechanism, where the agent makes all the decision and the process is virtually transparent, the client (the trader in this scenario) is given mechanisms to veto a trade or propose a new one if deemed necessary. Such actions, if not taken into account, may harm the social welfare of the system, as strategies are now not simply derived by agent, but also with human intervention. With the above innovation, we present our theoretical framework for human-centric mechanism design (HMD). One major difference between HMD and conventional mechanism design is that human are generally less rational than machines and contains emotions, which violates the rationality assumption in the first place. In the case of automated trading, for example, human also have a tendency to either settle too soon or hold on too long. Nonetheless, machine makes decisions (and take corresponding actions) in milliseconds, which is something infeasible for human to do.

It is very difficult, if not impossible, to model a hybrid of human behavior and machine behavior using a single, unified model. The bright side is, in a grey-box scenario, most of the decisions are still made by the agents, and human interventions appears only on critical situations. Our design has taken into account serval of those situations, and incorporated them into the overall mechanism design process. Design considerations such as computational cost, stability, and symmetry are particularly important in our framework, and each challenge has been properly addressed.

3.3. The random matching game

3.3.1 The Prisoner's Dilemma: First, we review some basic notions from game theory. Consider a game with n players, labelled 1, 2, . . . , 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 (s1, s2, . . . , sn) such that, for each player i, if all the other players j follow their assigned strategies sj , then strategy si 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:

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, R = b − c, P = 0, 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.

3.3.2. 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.

3.3.3. 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 subgameperfect 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].

3.1.4. 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].

3.1.5. 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].

3.4. 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.

3.4.1. 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 2i, i = 0, 1, 2, . . . , lg(N)− 1. That is, the fingers are at distances 1, 2, 4, . . .,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.

3.4.2. 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 1/2j . 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 1/2j . 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 _ = 1, 2, . . . , lg(N) − 1, the probability of at some point taking a hop of length 2_ is 1/2;

for _ = 0, the probability of taking a hop of length 2_ = 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 ≈ 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.

3.4.3. 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 −→ B −→ 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 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 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.

3.4.4. Proof of Equilibrium: First, some preliminary remarks: Suppose that some event occurs at random times specified by a Poisson process with rate λ. Let Yi be the time of the i’th event, and Ti 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 = δY1p + δY2p + · · · and our expected payoff is [1]

E[P] = E[δY1]p + E[δY2]p + · · ·
Since Yi = T1 + · · · + Ti, and the random variables
T1, . . . , Ti are independent and identically distributed, we have
E[δYi] = E[δT1 · · · δTi] = E[δT1] · · · E[δTi] = E[δT1]i
We define the “effective discount factor” to be δeff = E[δT1].
This lets us write the expected payoff as
E[P] = δeffp + δ2
effp + · · · = p
δeff
1 − δeff
The probability density function for T1 is p(t) = λe−λt,
and so we have
δeff = E[δT1] = _ ∞
0
δtλe−λtdt = λ
λ − log δ
Also, let P[0,τ] be the payoff during some time interval
[0, τ]. Then
E[P[0,τ]] = (1 − δτ)E[P] = (1 − δτ)p Δ eff 1 – δeff

The proof involves checking the incentives of each node. Each request has cost c for the nodes that services it, and benefits 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 λs, and receives requests at rate λr. For routing, λr 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). [1]

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 T1, 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 τ +T1. This will affect any requests that we send during the interval [τ, τ +T1]. 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 T1 between following the social norm and deviating from it is $\geq -c$ + E[# of requests sent in

$[\tau, \tau + T1]]\delta\tau\, b = -c + \lambda sE[T1]\delta\tau\, b = -c + \lambda s$
λr
$\delta\tau\, b$
it needs this to be ≥ 0, that is,
$\delta\tau \geq$
λrc
λsb

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, τ]. 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). [1]

Let δeffs and δeffr be the effective discount factors for sending and receiving requests. The expected payoff difference between following the social norm and deviating from it is

$\geq -c + (1 - \delta\tau)b\, \delta effs$
$1 - \delta effs - (1 - \delta\tau)c\, \delta effr$
$1 - \delta effr = -c + (1 - \delta\tau)b$
$\lambda s - \log\delta - (1 - \delta\tau)c$
$\lambda r - \log\delta = -c + (1 - \delta\tau)\lambda sb - \lambda rc - \log\delta$

It needs this to be ≥ 0, that is,

$1-\log \delta \geq c(1 - \delta\tau)(\lambda sb - \lambda rc)$

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:

$\gamma = 1/2$

$1-\log \delta \geq 2$ =□ $\delta \geq e-1/2$, so choose $\delta = 2/3$

$\tau = -1/ \lg \delta \approx 1.71$

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:

$\gamma = 1/2$

1

$-\log \delta \geq 1/5$ =□ $\delta \geq e-5$, so choose $\delta = 2/3$

$\tau = -1/ \lg \delta \approx 1.71$

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

9. Related Studies

Ralitsa Kostadinova and Constantin Adam [5] analyze epidemic algorithms, focusing on disseminating information. Epidemic algorithms offer effective solutions for disseminating information in large scale and dynamic systems. They are easy to deploy, robust and provide high esilience to failures, thus make them particularly useful in P2P systems. Yair Amir and Claudiu Danilov [ ] discuss reliable point-to-point communication usually achieved in overlay networks by applying TCP on the end nodes They achieved reliability on hop-by-hop basis thus reducing the latency and jitter considerably. There approach is feasible and beneficial in overlay networks that do not have the scalability and interoperability requirements of the global Internet. Though theory scheme produces overhead at the nodes but does not contribute much in the overall gain. Junghee Han, David Watson, and Farnam Jahanian [4] presented an idea of topology aware overlays networks by using the inherent redundancy of the Internet's underlying routing infrastructure to redirect packets along an alternate path when the given primary path is not available (link is down or due to congestion). However, the performance efficiency of these overlay networks depends on the availability of alternate path between the two hosts in terms of physical links, routing infrastructure, administrative control, and geographical distribution. There analysis shows that a single-hop overlay path provides the same degree of path diversity as the multi hop overlay path for more than 90% of source and destination pairs. Finally, they also validate that their architecture is able to provide a significant amount of resilience to real-world failures. Meng- Jang Lin, TX Keith Marzullo, San Diego presenting concept of directional gossip [1], in which there is a gossip server. A gossip server is aware of the network and it is responsible for directing direct path of one node to its neighbors. The protocol that a gossip server *s* executes is the following. Initially, gossip server *s* knows only the direct paths connecting itself to its neighbors. It will give initialization and assign an initial weight of one to each of its neighbors. The weights of initialization vector may be low, but as the gossip server learns more paths by sending new messages to all of its neighbors, it will compute more accurate weights. In directional gossip deterministic topological node selection mechanism is utilized to optimize query processing in WAN. This result in, low to moderate overhead by having a node identify the critical directions it has to forward gossip messages. Therefore, directional gossip helps in reducing overhead with time, achieves good reliability. David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris, worked on Resilient Overlay Network[2] (RON) an architecture that allows distributed Internet applications to detect and recover from path outages and periods of degraded performance within several seconds,. They found that forwarding packets via at most one intermediate RON node is sufficient to overcome faults and improve performance in most cases. These improvements, particularly in the area of fault detection and recovery, demonstrate the benefits of moving some of the control over routing into the hands of end- systems.

Communication bandwidth is often a scarce resource during the attacks, so the attack information sharing should involve only small messages. Recently, gossip based protocols have been developed to reduce control message overhead while still providing high reliability and scalability of message delivery [9]. Gossip protocols are scalable because they don't require as much synchronization as traditional reliable multicast protocols. In gossip based protocols, each node contacts one or a few nodes in each round (usually chosen at random), and exchanges information with these nodes. The dynamics of information spread bears a resemblance to the spread of an epidemic, and leads to high fault tolerance. Gossip-based protocols usually do not require error recovery mechanisms, and thus enjoy a large advantage in simplicity, while often incurring only moderate overhead compared to optimal deterministic protocols.

10. Conclusion

The Internet is a shared resource, a cooperative network built out of millions of hosts all over the world. Today there are more applications than ever that want to use the network, consume bandwidth, and send packets far and wide. In this paper, we have described a heterogeneous peer-to-peer network and used as a random-matching game to model routing in peer-to-peer networks as previously studied and authored by [1]

References

[1] Alberto Blanc1, Yi-Kai Liu2, Amin Vahdat3," Designing Incentives for Peer-to-Peer Routing",.

[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*

[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.