First Workshop on the Economics of Networked Systems (NetEcon06)

A workshop of **EC-06: ACM Conference on Electronic Commerce**
http://www.cs.duke.edu/nicl/netecon06

June 11, 2006
Ann Arbor, Michigan

NetEcon merges two workshops held in previous years: P2PEcon (Economics of Peer-to-Peer Systems) and PINS (Practice and Theory of Incentives in Networked Systems). The goal of the workshop is to promote a cross-disciplinary exchange of ideas on the role of game-theoretic and economic principles in the design and analysis of networked systems.

The influence of incentives is fundamental when the users of a system have competing interests and may behave selfishly. In particular, networked systems are often sustained by resources contributed and controlled by their participants, and their resources are consumed by individual user choice but are managed as a commons for the benefit of the group. Contexts of particular interest for this workshop include Internet routing and traffic control, peer-to-peer services, distributed hosting platforms (utilities or grids), and wireless mesh networks.

We reviewed 26 position papers for NetEcon06 and selected 12 papers to present and discuss at the workshop. The final versions of the papers are included in this volume and are available on the workshop website at *http://www.cs.duke.edu/nicl/netecon06*.

Program Committee

Program for the First Workshop on Economics of Networked Systems (NetEcon06)

Assessing the assumptions underlying mechanism design for the Internet

Steven J. Bauer
Massachusetts Institute of Technology
77 Mass Ave
Cambridge, MA
bauer@mit.edu

Peyman Faratin
Massachusetts Institute of Technology
77 Mass Ave
Cambridge, MA
peyman@mit.edu

Robert Beverly
Massachusetts Institute of Technology
77 Mass Ave
Cambridge, MA
rbeverly@mit.edu

ABSTRACT

The networking research community increasingly seeks to leverage mechanism design to create incentive mechanisms that align the interests of selfish agents with the interests of a principal designer. To apply mechanism design, a principal designer must adopt a variety of assumptions about the structure of the induced game and the agents that will be participating. (We focus in this paper on assumptions regarding agent preferences and non-repeated vs. repeated games.) As we demonstrate, such assumptions are central to understanding the degree to which theoretical claims based upon mechanism design support architectural design decisions or are useful predictors of real-world system dynamics. This understanding is central to integrating the theoretical results from mechanism design into a larger architectural discussion and engineering analysis required in networking research. We present two case studies that examine how the valid theoretical claims of [7, 18] relate to a larger, architectural discussion. We conclude with a discussion of general criteria for designing and evaluating incentive mechanisms for complex real-world networks like the Internet.

Categories and Subject Descriptors

J.4 [**Social and Behavioral Sciences**]: Economics; C.2.1 [**Network Architecture and Design**]: Packet-switching networks

General Terms

Economics, Theory

Keywords

network architecture, mechanism design, agent preferences, repeated games

1. INTRODUCTION

The networking community has often designed architectures and protocols that rely on the cooperative behaviors of participants (e.g. TCP). The field of mechanism design, though, suggests that network architectures and protocols can be designed that align the interests of non-cooperative selfish agents with the interests of a mechanism designer. This then would seem to be a powerful theory in which strong claims could be made of agent and system behaviors. However, strong claims are contingent upon many assumptions about selfish agents that may not hold in practice. Claims are much weaker if a mechanism only aligns the incentives of the subset of selfish agents that happen to match a principal's underlying assumptions.

While adopting simplifying assumptions can enable a network architect or protocol designer to prove theoretical properties such as the incentive compatibility or efficiency of a mechanism, these results may not always be useful predictors of actual agent behaviors or system dynamics when a mechanism is deployed in practice. While the simplifying assumptions of any theory are often easy to criticize from a practical perspective, the point of this paper is to not to criticize. Rather we hope to further the use of mechanism design by the networking community by promoting a better understanding of the contexts in which mechanism design succeeds (or fails) in practice to improve network and protocol designs.

While mechanism design requires simplifying assumptions – rationality, common knowledge – we focus on what mechanism designers can know about agent preferences and what they assume about whether the induced game will be a single-shot or repeated game. We focus on these assumptions because they strongly influence how applicable the theoretical results of mechanism design are in practice for the networking community.

The first class of assumptions we examine is the structure and type of agents' preferences. We consider how realistic various assumptions about agents' utilities are in practice. While the majority of agents participating in a game induced by a mechanism may match a designer's assumptions, it is likely that at least some agents will fail to conform to a mechanism designer's expectations in networks as large, complex and diverse as the Internet.

The second class of assumptions we examine deals with whether the game induced by a mechanism is part of a larger repeated game. In mechanism design the induced game is typically analyzed as a single-shot game. However, in networking, agents will interact repeatedly with mechanisms experiencing, over time, multiple mechanism outcomes. We therefore consider mechanisms which induce an outcome in a stage-game of a larger repeated game. It is well known that the equilibria of repeated games can be different than the equilibria of single-shot games [11]. Indeed, previous research in the networking community has noted the effect of repeated play on various routing mechanisms [2]. This paper considers more broadly the implications of the folk theorem [11] for mechanism design in any repeated context – a context that is very common in the real-world networking environments.

The rest of the paper is organized as follows. In §2 we discuss mechanism design for the Internet. In §3 we examine the assumptions about agent's preferences and provide an example of a mechanism, Re-Feedback [7], in which assumptions about the agents play a critical role in understanding the incentive compatibility claims. In §4 we examine assumptions about the number of times an agent plays the game induced by a mechanism and examine the implications for an ad-hoc routing and forwarding protocol [18] that is designed to be incentive compatible. In §5 we discuss the impact of our arguments on mechanism design for the Internet. In §6 we conclude with a summary.

2. MECHANISM DESIGN

Since this paper is targeted primarily at the networking community we begin with a brief review of mechanism design. As described in Fudenberg [11], mechanism design can be viewed as a multi-step game of incomplete information where agent "types" are private information. In the first step of the game the mechanisms designer, or principal, designs a "mechanism", "contract" or "incentive scheme." The objective of this mechanism is to illicit "messages" or "behaviors" from agents such that the mechanism's designer, or principal's expected utility is maximized. In the case of a benevolent principal, the expected utility that is maximized is some notion of social welfare. As network architects we often optimistically view ourselves as such benevolent principals, designing mechanisms to improve some notion of overall social welfare for the network.

In next step of the game, each agent either accepts or rejects the mechanism designed by the principal. Agents that accept enter the third step and play the game induced by the mechanism. Playing the game entails sending messages that are selected based upon an agent's private "type." In a networking context, one can interpret sending a message to a mechanism as engaging in a behavior that is observable to the network providers or other network participants.

The outcome of the game induced by the mechanism is called an "allocation" or "decision" k which is computed by the mechanism from the agent messages. The allocation consists of an assignment of goods and transfers of numeraire [13]. The allocation, for instance, in a VCG-based lowest-cost routing mechanism [8] consists of a selection of routing path and numeraire transfers of monetary payments to each of the nodes on the lowest-cost path. More generally, numeraire can be in terms of anything the agent values; often these are monetary transfers, but they can also be tokens that are valuable within the context of the mechanism. For instance, in mechanisms designed for the Internet these tokens might represent the right to transmit in a wireless network or they might be tokens employed in a traffic-shaping token-bucket.

The problem facing the mechanism designer is how to construct the message space and allocation rules such that it is in the interest of agents to truthfully reveal the private information that the principal conditions it's allocation decisions upon. Said another way, the mechanism must be designed to align the interests, behaviors, and actions of the agents with the interests of the mechanism designer.

In designing a mechanism, the principal is assumed to have some leverage over the agents that influences the agents' choice of messages or behaviors. This leverage is rooted in the principal's control over how goods and transfers are allocated to the agents. When designing a mechanism for the Internet then, an important question is what can a principal assume with confidence about agent preferences? The answer to this question is critically important to both the design of the mechanism as well as the equilibria that will result in the induced game. These are the topics that we consider in the following section.

3. ASSUMPTIONS ABOUT PREFERENCES

In the game of mechanism design, each agent has a private type θ_i that determines the agent's preferences over different allocations.[1] Agent types are assumed to be drawn from a known set of types $\theta_i \in \Theta_i$. The vector of all agents types is denoted as $\theta = (\theta_1, \ldots \theta_I)$ drawn from a vector of possible types $\theta \in \Theta_1 \times \cdots \times \Theta_I$ with a probability density function $\phi(\cdot)$. Each agent is also assumed to be an expected utility maximizer where the agent's utility function for an allocation k from the mechanism is denoted $u_i(k, \theta_i)$.

Many mechanisms designed by the networking community have further assumed that the structure of agents' utility functions have a quasilinear form:

$$u_i(k, \theta_i) = v(k, \theta_i) + (m_i + t_i) \tag{1}$$

where m_i is the agents i's initial endowment of the numeraire, k is the allocation of the good, $v()$ is the agent's valuation function for the allocation, and t_i is transfer of numeraire to/from agents. Quasilinear utility functions are popularly adopted because utility can be transfered across agents through transfers of the numeraire.

A key underlying assumption is that the probability density function $\phi(\cdot)$, the complete sets of possible types for each agent $\Theta_1, \ldots, \Theta_I$, and the structure of the utility functions $u_i(\cdot, \theta_i)$ are all common knowledge. In other words, all of this information is known by all agents and the principal. The only private information is the actual type of each agent θ_i.

By assuming that the complete set of possible types for each agent and structure of all utility functions are known, the principal can theoretically anticipate the effect of incentive mechanisms upon agent behaviors. If all these assumptions are sound, i.e. the assumptions accurately represent agents' utilities and types in the real world, then the mechanism designer can with confidence predict and describe the behaviors and equilibria in the game induced by a mechanism.

3.1 Assessing utility assumptions

The question, from a network architect's perspective, is what should be assumed about agents' utilities in network environments? In this section we consider mechanisms that assume agent utility functions are composed of terms representing the value of monetary and/or network goods to an agent.

3.1.1 Monetary utility terms

A monetary term that captures an agent's increase in utility with increased monetary assets seems fairly safe to assume in a utility function. Most selfish agents would seemingly prefer a larger amount of monetary goods to a smaller amount.

However, even this relatively safe assumption may not hold when considering the time value of money. An agent may be willing to incur a smaller short-term loss for a larger long-term gain. If agents are willing to incur loses within the induced game for longer-term gains realized in a different larger or repeated game that includes the induced game, a mechanism designer has more limited leverage to shape the agents behaviors through monetary incentives. In effect, the agents will not be playing the game the mechanism designer intended.

This is interesting because it suggests that even monetary rewards or penalties may not create the incentives expected by a principal. While perhaps most likely to occur over monetary goods, a tolerance for short-term loses for longer-term benefits also potentially effects how agents value network characteristics such as the ones discussed in the next section.

3.1.2 Network-based utility terms

The next class of terms in an agent's utility function we consider are ones that represent the value of network goods. These are

[1] See (23.B) "The Mechanism Design Problem" [16], for a more complete introduction to the assumptions summarized in this section.

terms that represent the value of network performance characteristics such as throughput, latency, and loss as well as more general goods such as transmission or access privileges on a network. It is often assumed that agents' utilities are an increasing function of improvements in the network good. Assuming that agents have traffic they want to send or receive on the network, such assumptions seem at first plausible.

However, a lesson from years of quality of service research in the networking community is that simplistic models of agent utilities are inadequate in the real-world [5]. Assuming that agents always value improvements in any one metric of network performance, such as throughput, latency, or network access fails to describe any one individual agent let alone being a good model for all agents on the network [5].

Moreover, in most networks today, agents are actively seeking to send or receive traffic only a small fraction of the time. During these periods, incentives leveraging the fact that agents value improvements in network performance will be effective. But, this raises the question of what governs and motivates an agent's behavior during other times? One is tempted to answer that agents will just cooperate during periods when they themselves are not selfishly invested in how the network is performing. But such a response weakens the strength of claims that can be made regarding a mechanism, particularly in networked environments, such as we have today, where actively malicious behaviors are common.

Finally, we note that a mechanism deployed in the real world cannot selectively admit only those agents that are well modeled by the utility functions assumed by the principal. Agent utilities and types are inherently private information. Agents that do not conform to a principal's assumptions may participate in the induced game. Therefore, the claims that can be made for a practical mechanism must be seen as limited to the subset of selfish agents that match a principal's underlying assumptions.

3.2 Case study: Re-Feedback

In this section, we provide a concrete example of a mechanism, Re-Feedback [7], in which assumptions about the agents play a critical role in understanding the incentive compatibility claims. We selected Re-Feedback because it was presented at one of the premier networking conferences and leverages mechanism design in support of a proposed real-world network architecture. We first provide a brief overview of the mechanism.

3.2.1 Objectives and incentives

The Re-Feedback framework attempts to add accountability for causing congestion to transport protocols. It is a technical network architecture that enables a "receiver-aligned" view of the downstream congestion along a path. How this congestion information is employed is left to the discretion of the network operators. They could directly charge or shape traffic based upon the congestion information.

We present a slightly abstracted description of the Re-Feedback mechanism; for details readers should consult the literature [7]. The framework leverages the Explicit Congestion Notification (ECN) bits of the IP header and the congestion feedback in the transport header in a novel manner to expose to network providers along a path the sender's "true" expectation of downstream congestion.

In Re-Feedback, the sender of a flow of traffic declares the expected congestion along the path (i.e. the rate at which the congestion-experienced (CE) bit will be set by ECN-capable routers) by setting selected bits (the ECT(0) code point) in the IP header. From replies sent back from the receiver in the transport protocol, which indicate the actual congestion experienced rate, the sender is capable of accurately adjusting it's expectation of downstream congestion. Routers in the framework are unmodified; they are simply assumed to set the CE bit in packets during periods of congestion.

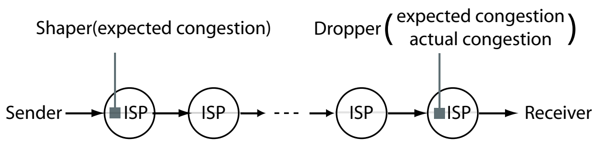

Shaper(expected congestion)
Dropper (expected congestion / actual congestion)
Sender
ISP
ISP
ISP
ISP
Receiver

Figure 1: The incentives of Re-Feedback are designed to induce senders to truthfully report the expected congestion along a path. A shaper creates an incentive for senders to not *overstate* their view of expected congestion. A dropper creates an incentive for senders to not *understate* their view of downstream congestion.

Of course, without additional components, senders would have no incentive to accurately state their expectation of the downstream congestion. Re-Feedback tries to create this incentive through the addition of shapers and droppers in the network (see Figure 1). A shaper creates an incentive for senders to not *overstate* their expectation of downstream congestion by shaping the traffic flow to conform to a TCP-friendly rate given the stated expected congestion rate. A dropper creates an incentive for senders to not *understate* their expectation of downstream congestion by dropping enough traffic to make the rate of congestion marked packets (CE) equal to the rate of packets marked with expected congestion (ECT(0)) in a flow of traffic.

Any deviation from truthfully stating the expected downstream congestion leads to shaping or dropping the sender's traffic so that the overall throughput to the receiver is reduced. These incentives are designed to lead to a maximization of social welfare where the social welfare function is the global aggregate throughput in the network.

3.2.2 Claims and analysis

A series of claims are made regarding the behaviors that the Re-Feedback framework will induce [7].

1. "We have introduced an incentive framework which ensures that the dominant strategy of selfish parties around the feedback loop will be to declare Re-Feedback honestly." (p.11)

2. "The ingress edge network can rely on downstream congestion declared in the packet headers presented by the sender." (p.5)

3. "Inter-domain congestion charging ensures that any network that harbors compromised "zombie" hosts will have to pay for the congestion that their attacks cause in downstream networks." (p.9)

We now examine the implicit agent assumptions underlying these claims. The first assumption is that all agents' utilities are an increasing functions of throughput (throughput is the good being allocated by the mechanism). If the ingress network charges for a sender's declared expected congestion, the agent utility function $u()$ has a classic quasilinear form with a valuation function $v()$ increasing with increased throughput and the transfer charges t_i increasing as well with increases in the expected congestion rate. The question, then, is are these sound assumptions for agents on the Internet?

Perhaps, in some scenarios these assumptions would be sound. But if Re-Feedback is positioned as a *general* purpose incentive architecture for the Internet this may not always be the case. Consider, for instance, a prevalent problem on today's Internet of denial of service attacks. Agents that participate in such attacks not only do not value their own throughput, they also want to diminish others' throughput as well.

As indicated by the zombie claims above, the Re-Feedback framework seeks to address such malicious behavior. But consider the following behavior of an agent that wanted to launch a denial of service attack on the network infrastructure. All the agent would have to do would be send network traffic at any rate (up to full access line rate) declaring no expected downstream congestion (e.g the ECT(0) code point is not set on any packets).

But note that this strategy could reasonably be employed by non-malicious agents as well. An application might be designed that employed a loss resistance encoding (e.g. erasure coding) and streamed a large data set across the network. Declaring no expected congestion would result in the Re-Feedback dropper discarding many packets if the network were congested, but the application would be able to make use of any available non-congested network periods.

Note the result of this strategy in the Re-Feedback framework. Even if the ingress network is charging for declaring downstream congestion, the strategy is cost free as no congestion is ever declared. As no packets are marked with an ECT(0) code-point, the shapers also have no effect. The flow of traffic will be discarded by a dropper in the Re-Feedback framework, but, since droppers necessarily maintain flow state (and thus will always likely be closer to one of the network edges), this may not be until the egress edge network. (Egress droppers are depicted in the Re-Feedback paper [7].) This means that no network element before the first dropper can ever rely on the expected congestion declared for a flow if such agents are indeed present on the network.

Now consider that to combat this strategy, a dropper is added to the ingress edge of the network. An agent only has to declare an expected level of congestion equal to the congestion on the path from the sender to the ingress dropper. But still no element on the path from the ingress dropper to the egress dropper can rely on the congestion information declared by the sender. Adding additional droppers along the path raises the costs to the sender, but no guarantee can ever be made that all elements along a path can rely upon the declared expected congestion rate being representative of the actual downstream congestion.

The fundamental issue is that only the agents that seek to maximize throughput to a receiver will exhibit the social welfare maximizing behaviors. Given that many selfish agents on the Internet may not conform to these assumptions, the mechanism claims for Re-Feedback should perhaps be interpreted more narrowly.

Note, though, that we do not consider it a failure that Re-Feedback does not accommodate all the types of selfish agents on the Internet. Creating an incentive mechanism that aligns the interests of a subset of agents may be a worthwhile improvement over the current Internet that largely assumes full cooperation from all agents. It is, however, important to understand the limitations.

4. MECHANISM DESIGN FOR REPEATED GAMES

Classically, mechanism design is viewed as inducing a single-shot game. However, when mechanism design is applied to the construction of protocols and architectures for networking problems, it is actually more likely the same agents will be repeatedly playing the game induced by the mechanism.

From this perspective, the outcome of a game induced by a mechanism must be seen as the outcome of a stage-game i.e. one iteration of a single-shot game, in a larger repeated game. However, the effect of incentives in a single-shot game can be different then in a repeated game. The classic example of this is the prisoners' dilemma game. The only equilibrium in the single-shot game is for each prisoner to defect and take a plea bargain. However, in the repeated game, staying silent can also be an equilibrium strategy [11].

In general, any mechanisms designed by the networking community that are repeatedly played must be analyzed as repeated games. This entails that important theoretical results in repeated games must be considered – namely the "folk theorems" for repeated games (see Fudenberg [11] for formal statements of the folk theorems in repeated games).

The folk theorems assert that, if players are sufficiently patient, any individually rational, feasible outcome can be enforced by an equilibrium. To be individually rational, players select actions in each stage game that minimizes the maximum possible loss that they will face in the overall repeated game. A feasible outcome is one in which the rationality condition is satisfied for all agents. Thus, in a repeated game, almost any outcome can be an equilibrium outcome [11].

But since any feasible outcome can be supported for the repeated game, this raises the question of how much influence the incentives of the mechanism designer inducing each stage game has over the overall equilibrium of the repeated game. Consider again the classic prisoners' dilemma cast as a mechanism design problem. The principal representing the justice system wants to allocate prison sentences in such a way that induces guilty suspects to defect from their partners and tell the truth about their crimes i.e. the principal wants to design an incentive compatible mechanism.

However, in the context of a repeated game, prisoners will always maximize their utility by continuously remaining silent in each stage-game. Such an equilibrium can be enforced, for instance, if agents adopt tit-for-tat or grim trigger strategies that punish any agent that ever defects. The principal representing the justice system in this repeated game cannot design an incentive compatible mechanism for a single stage game if the penalty allocated to two prisoners that both remain silent must always be lower than the penalties if they both defect.

In different contexts, though, a principal can, to a degree, influence the equilibrium of the repeated game through the design of the messages that each agent can send to the mechanism. The work of Afergan [2], for instance, considers the effect of protocol periods and field granularity on the equilibrium price computed by a routing protocol. We are unaware of general results in this area; it appears that each mechanism must be analyzed individually in the context of a repeated game to understand what effect the control of incentives in each stage game will have over the equilibria in the repeated game.

4.1 Case study: ad-hoc networking

To illustrate the importance of considering repeated games in the engineering of network protocols we consider the incentives created in ad-hoc wireless networks by the protocols described by a 2005 paper in one of the premier conferences on wireless and mobile networking [18]. We focus in particular on the protocols for the routing stage. Space limitations preclude us from presenting a longer overview of the protocol. For specific details, consult the paper [18].

4.1.1 Ad-hoc routing and forwarding protocols

The goal of the routing stage is to compute the true costs, i.e. the power levels required for transmission, for each link along a path in the ad-hoc network. Based upon these costs, the price paid to each node on the lowest cost path is computed similar to the VCG-like mechanism presented in [4].

The challenge in wireless ad-hoc networks, the authors note, is that the cost of a link cannot be determined by the sender alone. Receivers are an integral part in reporting what power levels the sender must employ. In certain circumstances, which the authors describe, receivers have an incentive to misreport the power levels that a sender requires for transmission. To address this challenge the protocol designers employ a cryptographic solution that involves sending multiple messages, encrypted under a key shared with a third party, at increasing power levels. The receiver transmits all received messages to the third party that decrypts the messages and computes the true cost of each transmitting node.

4.1.2 Claims and analysis

A series of claims are made regarding the behaviors that these routing and forwarding protocols will induce [18].

- “We ... design the first incentive-compatible, integrated routing and forwarding protocol in wireless ad-hoc networks.” (p.13)
- “We show that following the protocols is a dominant action for [the routing stage.]” (p.13)

These claims, however, are based upon an analysis of a single-shot game induced by the protocols. However, routing and forwarding in an ad-hoc network will unquestionably be a repeated game. As Afergan [2] notes, agents can advantageously deviate from the behaviors they would exhibit in a single-shot version of a VCG-based routing game. Namely, agents that collude (either implicitly or explicitly) have an incentive to reveal costs that are higher than their true costs so that they can enjoy larger payments from the mechanism. Thus, the principal’s goal of truthful revelation of costs is potentially thwarted in a repeated game. Indeed, collusion between agents can only be supported in the context of a repeated game.

This is interesting because the cryptographic approach taken in this purposed protocol represents considerable engineering effort to align the incentives of a single-shot game. If, in fact, the game is repeated, admitting other agent behaviors, the considerable effort at aligning the incentives in the single-shot game appears potentially less worthwhile in the context of an engineering cost analysis.

5. DISCUSSION

This paper can be seen as an examination of applying theory to practice. While simplifying assumptions are crucial to employing theory and models, this necessarily entails that any model will not capture all details of the real-world. What is crucial is to understand when theory and models provide support and understanding of a system design versus when they are no longer applicable.

The theory of mechanism design can “raise the bar” of networking and protocol designs even if it does not accommodate all the types of selfish agents on the Internet or perform exactly as expected in a repeated game. We do not consider a mechanism designed for the Internet to be a failure simply because one can construct agents that do not meet the principal’s assumptions. Creating an incentive mechanism that aligns the interests of a subset of agents is an improvement over a design that assumes full cooperation.

But understanding the real-world limitations of theoretical claims is important from an architectural and system engineering perspective. It is this understanding of the real-world limitations that enables the theoretical claims to be integrated into a larger architectural discussion and engineering cost analysis. While there is not a rigorous framework in which to conduct this discussion and analysis, we offer the following criteria for designing and evaluating incentive mechanisms for complex real-world networks like the Internet.

1. *Explicitly state assumptions:* Understanding the implications and applicability of mechanism design requires the underlying assumptions to be explicitly stated so that they can be analyzed and tested for soundness.
2. *Design defensively:* Network architectures and protocols should not rely upon incentives derived from mechanism design alone to ensure that desirable system dynamics are achieved. At least some agents in any network environment will not conform to a mechanism designer’s assumptions.
3. *Understand the limitations of simple models of utility:* Assuming that agents always value improvements in any one metric of network performance, such as throughput, latency, or network access is not a realistic model of real-world agents.
4. *Analyze the repeated game:* Many mechanisms designed for networks will, in fact, be repeatedly played. The incentives created by this repeated play must be analyzed as a repeated game.

6. RELATED WORK

Our work was motivated by considering the implications of deploying, in practical networks, some of many mechanisms that have been proposed for network environments in recent years. This includes the work of [4, 7, 8, 10, 18].

While our work focuses on the underlying assumptions about agents, the work on Distributed Algorithmic Mechanism Design (DAMD) [9, 10] emphasizes the importance of the algorithmic properties of mechanisms designed for the Internet. They introduce the notion of protocol-compatibility which focuses on two aspects of the practical feasibility of a mechanism: the computational tractability and deployability of a mechanism.

The work of Afergan et al. [1, 2, 3] emphasizes the importance analyzing networking problems as repeated games. One of the focuses of this work is that the mechanism designer can influence the equilibria that occur in an incentive-based routing mechanism by controlling some of the protocol parameters such as the period lengths and granularity of protocol fields [2]. These results are dependent on other agent assumptions such as the adoption of “trigger price strategies.”

Practical experiences applying mechanism design and game theory to networking problems are reported in Mahajan et al. and Huang et al. [12, 15]. Both note that theory does not necessarily apply as completely or easily as one might initially have hoped.

Potentially more realistic models of agents utilities are considered in [6]. The work of [17] considers how to prove, under certain assumptions, that an implementation of a mechanism in real-world system will match a designer’s specification.

Finally, if a mechanism designer can re-implement a mechanism repeatedly then any outcome the designer cares about can be implemented in dominant-strategies [14]. This becomes possible because the principal can learn agents’ preferences by observing their past behaviors.

7. CONCLUSION

This work focuses attention on the underlying assumptions about agents and how they will interact with mechanisms in complex networks like the Internet. We have emphasized that strong claims are contingent upon assumptions about the selfish agents and how they will interact with a mechanism. We have suggested that claims should perhaps be interpreted more narrowly if mechanisms only aligns the incentives of a smaller subset of selfish agents that match a principal's underlying assumptions. But we emphasize that creating an incentive mechanism that aligns the interests of a subset of agents can still be seen as an improvement over a design that assumes full cooperation from all agents. Finally, we emphasize that mechanism design cannot be a substitute for a systems engineering perspective.

In summary, the contributions of this paper are the following:

- A study of two classes of assumptions (agent preferences and repeated vs. single-shot induced games) and their impact on mechanisms designed for complex, real-world networks
- A consideration of what the folk theorem entails for any mechanisms that induce an outcome in a stage-game of larger repeated game
- Example case studies that illustrate understanding and evaluating theoretical claims based upon mechanism design in the context of larger architectural and engineering discussions
- A list of architectural and design criteria to consider when evaluating or applying mechanism design for networking problems

8. REFERENCES

[1] M. Afergan. *Applying the Repeated Game Framework to Multiparty Networked Applications*. PhD thesis, Massachusetts Institute of Technology, August 2005.

[2] M. Afergan. Using repeated games to design incentive-based routing systems. In *Proceedings of the 2006 IEEE Infocomm Conference*. IEEE, 2006.

[3] M. Afergan and R. Sami. Repeated-game modeling of multicast overlays. In *Proceedings of the 2006 IEEE Infocomm Conference*. IEEE, 2006.

[4] L. Anderegg and S. Eidenbenz. Ad hoc-VCG: a truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *MobiCom '03: Proceedings of the 9th annual international conference on Mobile computing and networking*, pages 245–259, New York, NY, USA, 2003. ACM Press.

[5] G. J. Armitage. Revisiting IP QoS: why do we care, what have we learned? ACM SIGCOMM 2003 RIPQOS workshop report. *SIGCOMM Comput. Commun. Rev.*, 33(5):81–88, 2003.

[6] F. Brandt, T. Sandholm, and Y. Shoham. Spiteful bidding in sealed-bid auctions. In D. Lehmann, R. M"uller, and T. Sandholm, editors, *Computing and Markets*, number 05011 in Dagstuhl Seminar Proceedings. Internationales Begegnungs- und Forschungszentrum (IBFI), Schloss Dagstuhl, Germany, 2005.

[7] B. Briscoe, A. Jacquet, C. D. Cairano-Gilfedder, A. Salvatori, A. Soppera, and M. Koyabe. Policing congestion response in an Internetwork using Re-Feedback. In *SIGCOMM '05: Proceedings of the 2005 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 277–288, New York, NY, USA, 2005. ACM Press.

[8] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. In *Proceedings of the 2002 ACM Symposium on Principles of Distributed Computing.*, 2002.

[9] J. Feigenbaum, C. H. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63(1):21–41, 2001.

[10] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications.*

[11] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, Cambridge, MA.

[12] E. Huang, J. Crowcroft, and I. Wassell. Rethinking incentives for mobile ad hoc networks. In *PINS '04: Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 191–196, New York, NY, USA, 2004. ACM Press.

[13] M. Jackson. *Optimization an Operations Research*, chapter Mechanism Theory. EOLSS, Oxford, UK, 2003.

[14] E. Kalai and J. O. Ledyard. Repeated implementation. *Journal of Economic Theory*, 83(2):308, Apr. 1998.

[15] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Experiences applying game theory to system design. In *PINS '04: Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 183–190, New York, NY, USA, 2004. ACM Press.

[16] A. Mas-Colell, M. Whinston, and J. Green. *Microeconomic Theory*. Oxford University Press, New York, NY.

[17] J. Shneidman and D. C. Parkes. Specification faithfulness in networks with rational nodes. In *PODC*, pages 88–97, 2004.

[18] S. Zhong, L. E. Li, Y. G. Liu, and Y. R. Yang. On designing incentive-compatible routing and forwarding protocols in wireless ad-hoc networks: an integrated approach using game theoretical and cryptographic techniques. In *MobiCom '05: Proceedings of the 11th annual international conference on Mobile computing and networking*, pages 117–131, New York, NY, USA, 2005. ACM Press.

Punishment in Selfish Wireless Networks: A Game Theoretic Analysis

Dave Levin

Abstract— **In currently deployed wireless networks, rational participants have no incentive to cooperatively forward traffic for others. Though much work has focused on providing such incentives, few has done so with adequate focus on the ease of deployment; often, these systems require trusted third parties or tamper-proof hardware. In this paper, we use game theory to analyze two *internal incentive mechanisms*, which rely only on the primitives made available by standard 802.11 hardware. We show that isolating free-riders (*i.e.*, refusing to forward through or for them) is not sufficient in all scenarios, and we motivate a new incentive mechanism: punishment via channel jamming. We also show that jamming yields a fair Nash equilibrium for all nodes, *i.e.*, that all nodes can provide incentives to their neighbors to forward their packets. Lastly, we discuss some of the emergent behaviors in these equilibria, as well as guidelines for the design of a jamming strategy.**

I. Introduction

Each participant in a wireless ad hoc network is both end-host (it generates its own data and routing traffic) and infrastructure (it forwards traffic for others). Forwarding others' traffic can consume a considerable amount of battery life, yet no currently deployed wireless routing protocols provide incentives for participants to route or forward. Indeed, *rational*, self-interested nodes will free-ride from currently deployed protocols. To ensure cooperation, protocol designers should build incentives directly into the protocols [15].

Internal vs. External Incentives: Building incentives for nodes in a wireless ad hoc network to cooperate is not a new problem, but most existing systems make assumptions that are simply too strong for a reasonable deployment. These assumptions generally include introducing one of two new components to the wireless network: trusted third parties (*e.g.*, banks) or trusted, tamper-proof hardware. Since these are not inherently part of existing wireless networks, we refer to them as *external incentive mechanisms*. Few systems have focused on what one can call *internal incentive mechanisms*, which make sole use of the primitives already available in deployed wireless (802.11) networks. Yet, systems which use such mechanisms are more likely to experience a timely deployment.

To better understand internal incentive mechanisms, we develop in this paper a game theoretic framework in the form of repeated games played on a strongly connected graph (Section III). Each player's strategy set is limited to what can be feasibly done with standard 802.11 hardware (as opposed to, say, interacting with a bank). Using this model, we analyze the predominant internal incentive mechanism: isolating a node by refusing to forward to or through it [4], [12], [13]. We show in Section IV that *isolation does not always yield system-wide cooperation*.

A New Internal Incentive: Motivated by this observation, we introduce a new incentive mechanism in Section V: punishment via channel jamming. Unlike isolation, jamming does not require any cooperation among nodes to punish a free-rider; a single jammer suffices. This fundamental difference leads us to a proof that *jamming is a sufficient mechanism for encouraging cooperation in all conditions, without requiring any trusted components*. We further extend our framework in Section VI to model noise in wireless networks, which is fundamentally different from the standard game theoretic notion of trembles.

Our game theoretic model shows that jamming is a viable means of punishment, but there are of course several considerations that must be taken into account when designing a system with jamming, such as: When should a node punish another node? How should a node react to another node's punishment? How should a node act to *avoid* being punished? We discuss these in Section VII. Designing a system that addresses these questions is the main focus of our ongoing work; the model presented in this paper is intended to guide this design.

II. Model and Assumptions

We first formalize our assumptions about the network and the nodes' preferences over potential outcomes.

A. Network Model

We assume the network to be an arbitrary, connected graph, $G = (V, E)$, of selfish ad hoc nodes, V. By *selfish* we mean that any $i \in V$ will act in whatever *rational* way that will maximize i's utility over time. Formally, if (u_i^t) and (w_i^t) are sequences representing i's payoffs at time $t = 1, \dots, T$, then i will prefer u to w if and only if there exists some $\epsilon > 0$ such that $\frac{1}{T}\sum_{i=1}^{T}(u_i^t - w_i^t) > \epsilon$. This condition is also known as the *limit of means criterion* [14].[1]

Edge (u, v) is in E if and only if u and v are within transmission range of each other. We can safely assume that edges are bi-directional, since 802.11 requires link-level acknowledgments [10]. Also, as assumed in the watchdog mechanism [13] and the Catch system [12] systems, when a node $u \in V$ sends a message (be it broadcast or unicast), all nodes in its one-hop neighborhood, $\mathcal{N}^1(u)$, overhear the message. We make this assumption so that we can analyze the resulting equilibria of systems such as Catch.

In terms of end-to-end connections, any two nodes $u, v \in V$ can communicate via some multi-hop path (*i.e.*, G is strongly connected). We assume that each node u knows its active connections and acts in a way that maximizes the sum goodput across these connections.

dml@cs.umd.edu. Department of Computer Science, University of Maryland, College Park. This work is supported by NSF-ITR Award #0426683 and a Sloan Foundation Fellowship awarded to Bobby Bhattacharjee.

[1]Please see [14] for thorough definitions of the game theoretic terms used in this paper.

B. Selfish Nodes' Preferences

It is not possible to formulate a general utility function to accurately capture to what degree each node prefers, say, connectivity over being disconnected. However, here, we present a reasonable set of preferences and assign nominal numeric values to different outcomes. A selfish node u can experience one of four outcomes: being disconnected or not, or (orthogonally) forwarding for other nodes or not. (We extend this to include punishment in Section V.) If the cost to forward is F, the benefit from being connected is C, and the utility gained from being disconnected is D, we have the following preferences:

- $C > D > F (< 0)$: Connectivity is the best outcome, but being disconnected at least does not expend resources, unlike forwarding.
- $C + F > D$: Nodes gain more benefit from being connected than what they lose by forwarding.

We can capture these properties by letting $C = 2$, $D = 0$, and $F = -1$; we assume each of these for the remainder of the paper only for ease of exposition, but these specific numbers do not change any of our fundamental results.

III. Ad Hoc Routing Games

We begin by formulating an *ad hoc routing game* which captures selfish nodes' preferences in a multi-hop wireless network. Such a game has many similarities to the well-known iterated prisoner's dilemma [14], repeated infinitely.[2] However, when modeling an ad hoc network, the game differs from most formulations of N-player games in the following ways:

1) Each node i plays a game with nodes in $\mathcal{N}^1(i)$.
2) The game $G(i, j)$ played between i and $j \in \mathcal{N}^1(i)$ is not necessarily independent of the games played between other nodes in i's two-hop neighborhood, $\mathcal{N}^2(i)$.
3) The payoffs of $G(i, j)$ (and therefore the dynamics of the game itself) depend on whether or not i has any interest in having j forward i's packets, and vice versa.

Hence, each game $G(i, j)$ must have an ever-changing set of payoffs, determined by others' actions and the desired end-to-end connections. We now motivate these three differences from the standard prisoner's dilemma.

Games are played between neighbors: Standard game theoretic models of N-player games generally assume that all N players may (or often must) play against one another. The network extension of games (see [1] for a nice survey) allows for a more suitable model of most networking problems, such as incentives-compatible BGP [8] and network planning [9]. Such a game includes an additional parameter to a game: a graph $G = (V, E)$, such that $|V| = N$ and games are only played between i and j if $(i, j) \in E$. We must therefore define a game for each pair of neighbors, $(i, j) \in E$. As we will see, games between different neighbors can vary significantly.

[2]Although, strictly speaking, players would not be expected to play infinitely, nodes generally do not know how long they will be in the network, so the game can be treated as infinite [14].

All of node i's games are interdependent: Let $\mathcal{U}_i^t(j)$ denote the utility i gains from game $G(i, j)$ at time t.[3] If all such games are independent, then the utility i gains from the system at time t is simply $\sum_{j \in \mathcal{N}^1(i)} \mathcal{U}_i^t(j)$. However, such games are not generally independent. For instance, i cannot forward packets for more neighbors at time t than the capacity of the wireless network allows. We assume for the remainder of this paper that the capacity of the wireless network is enough that all interfering nodes may successfully transmit their data in a given game, though in Section VI, we approximate interference with noise.[4] We make use of this interdependence when we introduce the notion of channel jamming as a punishment mechanism; when i is jammed, its utility for time t, $U_i(t)$, is forced to at most zero, regardless of the benefit that would have been gained from the sum of i's other games at time t.

Neighbors' interests may be asymmetric: Consider the example network in Figure 1(a). Node A gains utility from the system only if nodes B and C forward A's packets to node D. However, since B already has its end-to-end connection established (with C), B has no reason to ask A to forward its packets. Hence, there is an asymmetry of desire between A and B; A would gain utility with B's cooperation, but B gains no additional connectivity (and therefore no additional utility) by forwarding for A. Conversely, in the network of Figure 1(b), B and C have a mutual interest in one another, as they both would gain benefit from cooperating.

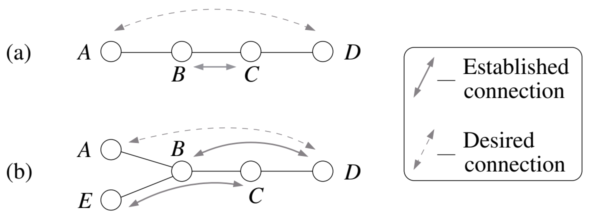


Fig. 1. Sample networks that motivate symmetric and asymmetric versions of the ad hoc routing game.

To capture players' varying desires, we use a different game for each scenario of interests: both are interested, only one is interested, or neither is interested. When both i and j are interested in having the other forward their packets, $G(i, j)$ is the standard prisoner's dilemma:

	Cooperate	Defect
Cooperate	$1, 1$	$-1, 2$
Defect	$2, -1$	$0, 0$

Game 1: The symmetric ad hoc routing game is the prisoner's dilemma. The pure strategy Nash equilibrium is (Defect, Defect).

When neither have interest, all payoffs are zero, since neither would have to spend any utility in forwarding (the other node will not request it), and neither will gain anything from having the other forward (since they have no interest), hence the

[3]For ease of exposition, we are making the simplifying assumption that time is slotted and that at each slot, a single round of each $G(i, j)$ is played.

[4]A model that more accurately models capacity would require a nonlinear program with constraints across all of i's games, and is an area of future work.

weakly dominant strategy is (Defect, Defect). Lastly, consider Game 2, where there is an asymmetry of interest; player 1 wants player 2 to forward but player 2 has no interest in player 1. For the uninterested player 2, Defect is a dominant strategy, since pl. 2 would gain no benefit from pl. 1 for performing this favor. Hence, the weakly dominant strategy is (Defect, Defect), meaning that any node i will not have its packets forwarded by any node who has no interest in i.

	Cooperate	Defect
Cooperate	2 , -1	0 , 0
Defect	2 , -1	0 , 0

Game 2: An asymmetric ad hoc routing game; pl. 1 wants pl. 2 to forward, but pl. 2 has no packets to forward through pl. 1. Defect is a dominant strategy for pl. 2.

Games 1 and 2 are sufficient to analyze systems that use isolation (defined in the next section) as a means of punishment [12], [13]. We show that isolation does not sufficiently account for the asymmetric game , and we introduce a new mechanism that provides incentive for all nodes to cooperate, independent of their interest in their neighbors.

IV. Punishing with Isolation

An intuitive strategy for enforcing cooperation in an ad hoc routing game is to *isolate* a free-rider f by ensuring that all nodes in $\mathcal{N}^1(f)$ play Defect in games against f, such as in Catch [12]. However, isolation (Defection) is not always a rational strategy for a node $i \in \mathcal{N}^1(f)$ to play in game $G(i, f)$. In particular, if any such $G(i, f)$ is the symmetric game (1), then i will be able to yield greater short-term utility by not isolating (Cooperating with) f. In Figure 1, B has no incentive to forward for A, hence A will attempt to isolate B. However, since C has no incentive to isolate B (at least, not in the short term), A's isolation will fail.

One could argue that, in some cases, there may exist greater long-term gain for $i \in \mathcal{N}^1(f)$ by participating in f's punishment. For instance, if the other neighbors of f were able to detect that i was not participating in f's isolation, then they could subsequently punish i. There are trivial cases where this does not work, *e.g.*, when f and i have an end-to-end connection with one another. Also, there are more general solutions that f and i could employ to make it appear that i is never forwarding for f. For instance, f could simply communicate with i over an encrypted channel, in essence resulting in a one-hop mix network [6].

Hence, as long as free-rider f has at least one neighbor with a mutual interest, (with whom it plays the symmetric Game 1), isolation is not a viable punishment. In fact, the only nodes who are guaranteed to gain no utility once isolation is in effect are the nodes for whom f was not forwarding in the first place.

V. Punishing by Jamming

We have shown that isolation does not guarantee cooperation by all rational nodes. Further, deployable isolation systems, such as Catch [12], seem to require rather strong assumptions: no collusion among nodes, MAC-level authentication, and MAC-level sender anonymity (*e.g.*, that nodes cannot use transmission power measurements to distinguish amongst its neighbors). As protocol designers, we are interested in the question: do there exist punishment strategies that *guarantee* cooperation by all rational nodes and can these assumptions be relaxed?

To this end, we consider channel jamming as a punishment mechanism. A node jams the channel by sending many broadcast packets (generally with no meaningful payload), thereby occupying the channel for all nodes within carrier sense range of the jammer (*e.g.*, its two-hop neighbors). Playing Jam costs J; we require that jamming costs more than forwarding ($|J| > |F|$), and assign J a nominal value of -2. To incorporate jamming into the ad hoc routing games (Games 1 and 2), we must capture the fact that whenever node i jams, none of the nodes in $\mathcal{N}^2(i)$ can receive any packets, and hence none gain utility from their neighbors' Cooperation. Let $c_f(t)$ denote the number of games in which f cooperatively forwards at time t, and recall that $\mathcal{U}_f^t(i)$ is the utility f gains from game $G(f, i)$ at round t. Then the ad hoc routing game with jamming is:

$$U_f(t) \overset{\text{def}}{=} \begin{cases} -2 & \text{if } f \text{ is Jamming} \\ -c_f(t) & \exists i \in \mathcal{N}^2(f) \text{ Jamming} \\ \sum_{i \in \mathcal{N}^2(f)} \mathcal{U}_f^t(i) & \text{otherwise} \end{cases}$$

Game 3: The ad hoc routing game with jamming. When no one in $\mathcal{N}^2(f)$ is jamming, the normal (symmetric or asymmetric) games are played.

Note that, although f cannot gain utility if any $i \in \mathcal{N}^2(f)$ jams, f can still pay the cost of forwarding for others (the second condition). Of course, f has no incentive to forward in this case; indeed, f achieves its minmax payoff (0) by playing Defect whenever any $i \in \mathcal{N}^2(f)$ jams:

Theorem 1: Any node i forces $j \in \mathcal{N}^1(i)$ to j's minmax payoff by Jamming; j in turn will Defect in all of its games.

Proof: The set of j's feasible payoffs when being punished is $(-\infty, 0]$, with 0 being obtained when j plays Defect in all of its symmetric games (Game 1) and asymmetric games (Game 2) where j is the node without interest. When being jammed, the asymmetric game where j is the node with interest have the same outcome, since j will never be asked to forward a packet; w.l.o.g., we can say j Defects in this case, as well. ■

Theorem 1 allows us to apply the well-known folk theorem, but first we require two definitions. A payoff profile (*i.e.*, a vector of utilities), $\mathbf{p} \in \mathbb{R}^N$, is said to be *feasible* if there exists a set of strategies that, when each node i plays its assigned strategy, its payoff is $\mathbf{p}(i)$, the i^{th} component of $\mathbf{p}$. Further, $\mathbf{p}$ is *strictly enforceable* if, for all i, $\mathbf{p}(i)$ is greater than i's minmax payoff; in effect, $\mathbf{p}$ is enforced by punishing nodes (forcing them to their minmax payoff) whenever they deviate from the strategy that would yield $\mathbf{p}$.

Theorem 2 (Folk Theorem [14]): Every feasible, strictly enforceable payoff profile of a game G is a subgame perfect Nash equilibrium payoff profile of the infinitely repeated version of G with the limit of means criterion (Section II).

Theorem 3: There exist subgame perfect Nash equilibria

(SPNE) with payoffs greater than system-wide defection that use jamming to punish free-riders.

Proof: By Theorem 1, jamming yields a minmax payoff. Any feasible payoff profile with payoffs greater than system-wide defection is therefore enforceable by jamming. Hence, by Theorem 2, such a profile is the payoff of at least one SPNE in which jamming is used as punishment. ■

Although Theorem 3 states that punishment can yield SPNE, it (like the folk theorem in general) does not specify precisely *how* to obtain these equilibria. Designing protocols (and punishment strategies) that yield these SPNE is a main focus of our ongoing work, and we discuss some of the necessary considerations in Section VII.

The Price of Jamming: To the best of our knowledge, jamming has only been studied as an attack, and this is not without reason. Even as a punishment mechanism, it incurs a loss of efficiency, since it pauses all connections within the jammer's two-hop neighborhood for the duration of the punishment. Additionally, it decreases the expected lifetime of the network as a whole, as nodes must expend additional energy to jam. Designing a punishment strategy that balances between this loss of efficiency and the gain of cooperation is the goal of our future work.

VI. Noisy Games

When all nodes act rationally, *and when all actions taken by i are viewed perfectly by $\mathcal{N}^1(i)$*, each node will cooperatively forward for others, and Jam will never be played. However, since wireless networks are inherently noisy, j will not overhear some of the packets i forwards on j's behalf. In the terms of our model, this means that with some probability, when i plays Cooperate in game $G(i, j)$, j will view i's action as Defect, even though i has paid the cost to Cooperate, F. For example, in Figure 1(b), B could have forwarded a packet to E for C at the same time that D sent a packet to C, resulting in a collision at C. Hence, although B cooperated, C is not able to verify; if this happens significantly more than the noise itself would cause, C must assume that B is defecting.

Nodes may not always cooperate, since they know that some of their defections could be interpreted as noise. In this section, we incorporate noise into the ad hoc routing game, and examine some of the resulting emergent behaviors.

A. Ad Hoc Routing with Noise

The notion of noise in a wireless network is fundamentally different from the standard game theoretic notion of *trembles*. In a game with trembles, when a player i chooses a strategy, there is some probability p that i tremble, *i.e.*, play a different strategy instead. If node i chose to Cooperate but trembled, it would simply play Defect instead, and vice versa, giving us the following game.[5]

	Cooperate	Defect
Cooperate	$1-p$, $1-p$	$3p-1$, $2-4p+p^2$
Defect	$2-4p+p^2$, $3p-1$	p , p

Game 4: The symmetric ad hoc routing game with standard game theoretic trembles with probability p.

However, Game 4 does not accurately capture the notion of noise in a wireless network. To see this difference, observe that when i's Cooperation is not viewed by j, i still has to pay the cost of forwarding, but does not gain the benefit of cooperation. Let U_c and U_d denote the utility that a node gains when it views its *opponent* playing Cooperate or Defect, respectively. In the symmetric game and for the interested node in the asymmetric game, $U_c = 2$ and $U_d = 0$ (because they will have to retransmit), whereas for the uninterested node in the asymmetric game, $U_c = U_d = 0$. We will modify slightly the definition of p: in our model, p represents the probability that node j views i's action as Defect, given that i actually played Cooperate.[6] Then the expected utility of cooperation and defection are:

$$\begin{aligned} E_c &\overset{\text{def}}{=} \text{E[Cooperate]} &= F + (1-p)U_c + pU_d \\ E_d &\overset{\text{def}}{=} \text{E[Defect]} &= U_d \end{aligned}$$

Note that when $p = 1$, $E_c = F + U_d \leq E_d$, so player i should always defect; the obvious correlation of this is that nodes ought not attempt to forward packets when the error rate is 1. We will assume for the remainder of the paper that $p < 1$. We can now formulate the following ad hoc routing games with noise; we derive these values by plugging in the values for U_c and U_d above.

	Cooperate	Defect
Cooperate	$1-2p$, $1-2p$	-1 , $2-2p$
Defect	$2-2p$, -1	0 , 0

Game 5: The symmetric ad hoc routing game with a more accurate model of noise. When $p = 0$, this is Game 1.

	Cooperate	Defect
Cooperate	$2-2p$, -1	0 , 0
Defect	$2-2p$, -1	0 , 0

Game 6: The asymmetric ad hoc routing game with a more accurate model of noise. When $p = 0$, this is Game 2.

Regardless of p, the minmax payoff for node i is achieved with (Defect, Jam), resulting in a total of 0 payoff for i at time t (similarly for j). When $p < 1$, the pure strategy Nash equilibrium of Game 5 does not differ from the corresponding games without noise: (Defect, Defect).

B. Playing with a Watchdog

In practice, detecting a neighbor's strategy requires a watchdog-like system [13]. The watchdog mechanism makes the standard assumption that whenever j forwards a message, all nodes $i \in \mathcal{N}^1(j)$ overhear the message, and can therefore determine if and when j has forwarded a packet on i's request. One way to implement a watchdog is as follows: each node i stores the weighted averages of $r_i(j)$, the number of unique packets that i requested j to forward, and $f_i(j)$, the number

[5]Note that, again, the strategies listed correspond to those *chosen* by the players, not the strategies that are necessarily played.

[6]To be precise, p would be a function of i and j, depending on the available capacity at the two nodes' respective locations in the network.

of these packets that j actually did forward. In a game with no noise (Game 3), $f_i(j) = r_i(j)$ for all i, j, since all nodes will cooperate to avoid punishment by jamming. However, when there exists noise, nodes will drop as many packets as they can while still avoiding punishment.

In a watchdog and in Catch [12], each node maintains a parameter θ, the threshold value that, if $f_i(j)/r_i(j) < \theta$, then node i considers j misbehaving. This parameter could change depending on the capacity of the wireless network, which depends in part on the bandwidths of neighboring links and the two-hop neighbors' desired flow rates [11]. We use the threshold value θ in defining nodes' punishment strategy. First, we show how a low threshold value (or high amount of noise) can lead to obsequious behavior in the network.

C. Avoiding Punishment with Forward Error Correction

The more a node i's neighbor perceives a defection from i, the greater the risk i has of being punished. Nodes can react to a high error rate (p) by employing some form of forward error correction (FEC). For ease of exposition, and to gain insight into what effect FEC has on nodes' strategies, we consider a naïve form of FEC, in which node i sends each packet multiple times, thereby "replacing" p with a smaller value. Of course, in practice, such a scheme would fail in the presence of high levels of congestion, but, again for clarity, we will assume failures are independent and, as stated in Section II, that the capacity of the network is infinite. Under these assumptions, if a node retransmits a packet r times (the *r-FEC strategy*), the probability that the previous hop will not see any of these is p^r. The expected utility from cooperating with r-FEC is thus:

$$E_c^r \overset{\text{def}}{=} \text{E[Coop w/ } r\text{-FEC]} = rF + p^r U_d + (1-p^r)U_c$$

Since U_c is gained at most once, this captures the fact that i's neighbor will not compensate a forwarded packet multiple times. The r-FEC strategy strictly dominates[7] the normal, single-transmission Cooperation when $E_c^r > E_c$, or

$$\begin{aligned} rF + p^r U_d + (1-p^r)U_c &> F + pU_d + (1-p)U_c \\ (U_c - U_d)(p - p^r) &> |F|(r-1) \qquad (1) \end{aligned}$$

In other words, nodes will employ forward error correction whenever the cost to forward the extra $r-1$ times (r.h.s.) is compensated by a greater expected value of utility (l.h.s.).

D. Emergent Behaviors

The resulting system-wide behavior of nodes depends on the punishment strategies they employ. A punishment strategy is a tuple (θ, δ), where θ is the threshold of free-riding at which to begin punishing (larger is more generous), and δ is the duration of the punishment (smaller is more generous). Hence, the *strength* of the punishment is proportional to δ/θ. For instance, a small θ and a large δ correspond to harsh, long punishments after the slightest noise or defection. Conversely, a high θ and low δ correspond to a generous node that punishes rarely and, if at all, for short durations. Here, we consider the behaviors that result from three different regimes of punishment strength.

[7] r-FEC weakly dominates under equality of Eq. (1).

Generosity Leads to Free-Riding: A node can be *generous* toward its neighbors by assuming a considerable amount of noise (*i.e.*, choosing a large θ) and punishing for a short duration (a small δ). Increased generosity allows for free-riding, as nodes exploit the large difference between how much they must forward and how much they are requested to forward ($f_i(j)$ and $r_i(j)$ from our watchdog). They do so without having to pay much price, even when they are discovered (since δ is small). However, generosity may be the best strategy when a level of trust is established between neighbors; this course of action will be the most resilient to spikes in noise or non-stop failures.

Stronger Punishment Leads to Obsequiousness: When δ/θ is high, there can be a large difference between U_c and U_d in Eq. (1), thereby making even our naïve version of FEC a viable strategy. As δ/θ continues to grow, the obvious price is the efficiency of the network as a whole; rampant jamming can vastly degrade capacity, and, since it expends more energy, the lifetime of the network will decrease, eventually leading to a disconnected network. One potential method to keep nodes from excessive jamming is to punish them by jamming in return, but this of course carries its own loss of efficiency (at least in the short term, until the nodes react to the punishment and change their strategy).

Efficiency by Matching Noise Levels: These two extreme punishment strategies (very low and very high δ/θ) incur a loss of efficiency: the former due to free-riding, the latter due to excessive jamming. We expect that the ideal outcome would be one in which the punishment strategy is *as tightly coupled to the given noise level as possible.* Studying such strategies is a focus of ongoing work.

VII. Choosing a Jamming Strategy

Varying punishment strategies can yield vastly different system-wide behaviors, ranging from incurring low overhead while allowing rampant free-riding, to punishing beyond any reasonable expectation of cooperation. Clearly, these two extreme points ought to be avoided, but the fundamental question as protocol designers is: *what punishment strategy yields the most system-wide efficiency and/or fairness?* We do not present a formal punishment strategy here, but we briefly discuss some guidelines worth consideration. Recall that a punishment strategy is a tuple, (θ, δ), consisting of the threshold θ (as defined by our watchdog) at which to begin punishing, and the duration of the punishment, δ.

The strategy should be adaptive: A viable punishment strategy must be adaptive, allowing θ to change as the available capacity of the wireless network changes to reflect new (or completed) connections. Along these same lines, the punishment strategy should ideally incorporate measurements of the available capacity into its calculation of θ. Available capacity can be measured locally at each node by calculating the link-level error rates and bandwidths, as well as the fraction of time for which the wireless channel is idle.

Punishments ought not echo: When node i punishes some $j \in \mathcal{N}^1(i)$ by jamming, all of the nodes within carrier sense range ($\mathcal{N}^2(i)$) are affected and, worse yet, the nodes in $\mathcal{N}^3(i) \setminus \mathcal{N}^2(i)$ do not necessarily know that i is even punishing. Hence, node $m \in \mathcal{N}^3(i) \setminus \mathcal{N}^2(i)$ could perceive the defection of $k \in \mathcal{N}^2(i)$ as a response to $G(k, m)$, and not as k playing its minmax strategy against i's punishment. Node m may therefore punish k, which then raises the same issue for i, since $i \in \mathcal{N}^3(m)$, and so on.

In effect, a single node's jamming can *echo* throughout the network, potentially indefinitely. To address this, it may be necessary to only jam with some probability small enough to limit the extent of such an echo. Addressing the echo of jamming is an area of future work.

Sharing one's views: Given issues such as the hidden node problem, it is not always the case that i knows when noise even takes place between itself and its neighbor. To help nodes understand the level of noise, p, each node i could forward to each of its neighbors, j, the values i's watchdog is storing to compute j's level of defection: $f_i(j)$ and $r_i(j)$. This is precisely what is done to compute link-level error rates for path metrics such as ETT [7]. Based on $f_i(j)$ and $r_i(j)$, j could estimate p and, if need be, choose an r with which to play r-FEC. An open question is how to ensure truthfulness in reporting $f_i(j)$ and $r_i(j)$.

VIII. Related Work

We briefly review existing systems that provide incentives to forward in wireless networks, as well as some known results about games played in noisy environments.

Systems with Incentives-Compatible Forwarding: Previous such systems can be categorized into two classes:
Payment schemes generally involve a trusted third party (TTP) [2], [17] or tamper-proof hardware [5] that generates digital currency. Peers pay others with tokens to forward data and route requests.
Detection and avoidance systems consist of two parts: (i) a *watchdog* that each node runs locally to determine when one of its neighbors is not forwarding data its data, and (ii) a policy to avoid sending to or forwarding on behalf of these defectors [4], [12], [13].

Neither of these types of systems can be used to apply incentives in a general setting. For instance, TTP-based payment schemes generally assume that wireless nodes have access (albeit infrequent) to the TTP itself. Perhaps future hardware will contain trusted, tamper-proof components that would remove the need for TTPs in payment schemes, but recent trends in digital rights management (DRM) indicate that this deployment would be slow, expensive, and hardly trusted after all [3].

Theory of Games in Noisy Environments: Previous game theoretic work on noise in the prisoner's dilemma (Game 1) has focused on the notion of trembles, some of the most influential work by Axelrod et al. Wu and Axelrod experimentally analyzed several strategies in an environment where nodes trembled, *i.e.*, when a player chose to play an action, the other action was, with some probability p, played instead [16]. Wu and Axelrod showed that, in the presence of more trembles, it is better for nodes to accept their punishment (*i.e.*, play "contrite tit-for-tat") when they defect than it is to simply act generously. This result is reflected in our proposed solution for stopping the echo of punishments. However, since the notion of a tremble is so different from that of noise in wireless networks (Section VI), it is not clear to what extent Wu and Axelrod's results apply.

IX. Discussion and Future Work

In this paper, we have developed a game theoretic model to analyze existing internal incentive mechanisms in wireless networks (*i.e.*, mechanisms that require only the primitives available in 802.11), and have introduced a new mechanism: punishment via channel jamming. We showed that isolation does not always ensure cooperation. On the other hand, jamming, though seemingly malicious, is a viable means by which to enforce cooperation of each node in the system, even when there are neighbors acting in a collusive manner by communicating only with one another. The price of jamming, if not engineered in a careful manner, can be high; jamming could, for example, echo throughout the network, resulting in a significant loss of efficiency. The main focus of our future work is in developing a viable punishment strategy that balances between the price of jamming and the gain of provable system-wide cooperation.

Acknowledgments

The author thanks Seungjoon Lee, Vijay Gopalakrishnan, and Bobby Bhattacharjee for their helpful discussion and comments throughout this work.

References

[1] E. Altman, T. Boulogne, R. Azouzi, and T. Jimenez. A survey on networking games in telecommunications, 2000.
[2] L. Anderegg and S. Eidenbenz. Ad Hoc-VCG: A Truthful and Cost-Efficient Routing Protocol for Mobile Ad Hoc Networks with Selfish Agents. In *Proc. of MobiCom*, 2003.
[3] R. Anderson. 'Trusted Computing' Frequently Asked Questions. Online: http://www.cl.cam.ac.uk/˜rja14/tcpa-faq.html.
[4] S. Buchegger and J.-Y. L. Boudec. Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes — Fairness In Dynamic Ad-hoc NeTworks. In *MobiHoc*, 2002.
[5] L. Buttyán and J.-P. Hubaux. Enforcing Service Availability in Mobile Ad-Hoc WANs. In *Proceedings of ACM MobiHoc*, pages 87–96. IEEE Press, 2000.
[6] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security*, 2004.
[7] R. Draves, J. Padhye, and B. Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. In *Proc. of Mobicom*. ACM Press, 2004.
[8] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based Mechanism for Lowest-Cost Routing. In *PODC*, 2002.
[9] A. Gupta, A. Srinivasan, and É. Tardos. Cost-Sharing Mechanisms for Network Design. In *APPROX-RANDOM*, 2004.
[10] IEEE. Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. IEEE 802.11 Standard, 1999.
[11] V. S. A. Kumar, M. V. Marathe, S. Parthasarathy, and A. Srinivasan. Algorithmic Aspects of Capacity in Wireless Networks. In *ACM SIGMETRICS*, 2005.
[12] R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan. Sustaining Cooperation in Multi-hop Wireless Networks. In *NSDI*, 2005.
[13] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *MobiCom*, 2000.
[14] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.
[15] C. Papadimitriou. Algorithms, Games, and the Internet. In *STOC*, 2001.
[16] J. Wu and R. Axelrod. How to Cope with Noise in the Iterated Prisoner's Dilemma. *Journal of Conflict Resolution*, 1995.
[17] S. Zhong, J. Chen, and Y. R. Yang. Sprite: A Simple, Cheat-Proof, Credit-Based System for Mobile Ad-Hoc Networks. In *Infocom*, 2003.

Why Share in Peer-to-Peer Networks?

Lian Jian* and Jeffrey MacKie-Mason†

May 26, 2006

Abstract

Prior theory and empirical work emphasize the enormous free-riding problem facing peer-to-peer (P2P) sharing networks. Nonetheless, many P2P networks thrive. We explore two possible explanations that do not rely on altruism or explicit mechanisms imposed on the network: direct and indirect private incentives for the provision of public goods. The direct incentive is a traffic redistribution effect that advantages the sharing peer. We find this incentive is likely insufficient to motivate equilibrium content sharing in large networks. We then approach P2P networks as a graph-theoretic problem and present sufficient conditions for sharing and free-riding to co-exist due to indirect incentives we call *generalized reciprocity*.

1 Introduction

Studies of peer-to-peer (P2P) networks as static games predict these systems should suffer from enormous free-riding— peers download but do not upload—in the absence of altruism or explicit incentive mechanisms to encourage content uploading [Ranganathan et al., 2003]. The fact that many peers free ride is empirically confirmed by [Saroiu et al., 2002] and [Adar and Huberman, 2000]. However, in practice P2P networks such as eDonkey, KaZaa, and Gnutella, persist and flourish despite free-riding. One possible explanation for this puzzling phenomenon is that altruism might sustain these networks. Rather than rely on this deus ex machina, we explore two alternative explanations: direct and indirect incentives for the private provision of public goods.[1] [2] Our immediate goal is to understand economic conditions under which networks of self-interested participants might be sustainable despite equilibrium free-riding. Our ultimate goal is to develop a plausible model of P2P behavior in order to evaluate various proposed mechanisms to increase sharing, and to develop our own mechanism for the same.

In the next section, we investigate a direct private incentive to provide public goods proposed by Krishnan et al. [2004], who suggest that sharing redistributes traffic in the network to the advantage of the sharing peer. We explore, in Section 3, generalized reciprocity as an indirect incentive explanation of both sharing and free riding on P2P networks. We close with a discussion of limitations in our work, and plans to continue this research.

2 Direct Private Incentives

Providing files (*sharing*) to a P2P network is an instance of the *private provision of public goods* [Bergstrom et al., 1986].[3] Sharing provides direct benefits to others for which, in the absence of an explicit incentive mechanism, the sharing peer is not compensated. One suggested explanation for the nonetheless observed sharing is that in

*School of Information, University of Michigan, Ann Arbor, MI 48109 USA; `ljian@umich.edu`

†School of Information, and Dept. of Economics, University of Michigan, Ann Arbor, MI 48109 USA; `jmm@umich.edu`

[1]We are not claiming that altruism does not exist or is unimportant. Rather, taken as a primitive, it is not susceptible to analysis, and does not help answer design questions. For example, if sharing occurs solely due to axiomatic tastes for altruism, we will have nothing to say about how to encourage increased sharing, unless we have a story about *why* people want to be altrustic: that is, what incentives do they have for sharing?

[2]Some P2P protocols impose "altruistic" (always on) sharing as a default setting. It may be a good design principle to encourage people to act as if they were altruistic, but that leaves open the incentives question: why do they not change the default setting?

[3]We adopt the widely-used convention of referring to uploading as *sharing*.

the process of providing a benefit to other users, a sharing peer is simultaneously obtaining a direct private benefit, similar to the personal incentive to donate, for example, to support a public radio station. Krishnan et al. [2004] model a particular form of this, which we call the "offload effect": sharing redistributes traffic in the network to the advantage of the sharing peer. In a P2P network, suppose peers A and B each want a different file, but both files are available from peer C. If peer A has the file that B desires, by offering to share with B agent A may get her file sooner from C, by offloading some of the demand on C.[4]

Krishnan et al. show that an offload effect could support a network in which all peers share is a dominant strategy equilibrium. After modeling the traffic redistribution and network congestion more precisely, we find it implausible that the offload effect alone is sufficient to motivate the amount of sharing seen on successful P2P networks.

2.1 Modeling offloading

We construct a four-period game. In period 1, $n \geq 3$ players join the network. In period 2, each player chooses whether to share or not, at a fixed cost of c or zero, respectively. Capacity is fixed and scaled so that a player can share at most one unit in a sharing period. In period 3, each player requests a unit of content from the network. The network protocol randomly assigns each request to one player who has decided to share in period 2. For consistency we adopt the important simplifying assumption made by Krishnan et al. that every node has at least one file wanted by any other node. Suppose $k \geq 2$ players have decided to share their content in period 2; then the probability that player i's request is assigned to sharing player s is $\frac{1}{k}$. If multiple requests are assigned to a sharing player, she randomly chooses one to serve. In period 4, files are shared, and payoffs are realized.

Suppose in period 3, i's request, together with m other requests, has been assigned to player s. s will pick i's request to serve with probability $\frac{1}{m+1}$. Given that the event of any peer's request being assigned to a sharing peer is $\frac{1}{k}$, this event follows a Bernoulli distribution, and the event that m other players will be assigned to node s follows a binomial distribution, $m \sim b(n-2, p)$.[5]

Consider an arbitrary node i deciding whether to share. She calculates the expected value from sharing or not (v_i^S, v_i^N). These values are defined as the probability of obtaining one unit of content. If i shares, the total number of sharing nodes is $k+1$; if she doesn't, it's k. Thus, the probability that another peer will choose the same source node as i is $p = \frac{1}{k+1}$ if i shares, and $q = \frac{1}{k}$ if i does not share. Now we need to calculate the expected value for i of being served by s, or $\mathrm{E}[(m+1)^{-1}]$. We calculate this as sum of the probabilities of m taking on each possible value on $\{0, \ldots, n-2\}$, times the probability that i gets a file from s when there are exactly m other demanders on s. Thus, the expected values, v_i^S and v_i^N, are:[6]

$$\begin{aligned} v_i^S(n,p) &= \Sigma_{m=0}^{n-2} C_{n-2}^{m} p^m (1-p)^{n-2-m} \frac{1}{m+1} \\ &= \frac{1-(1-p)^{n-1}}{(n-1)p} \qquad (1) \end{aligned}$$

and similarly,

$$v_i^N(n,q) = \frac{1-(1-q)^{n-1}}{(n-1)q}, \qquad (2)$$

where C_x^y is the number of combinations "x choose y". We define the marginal benefit of sharing (MBS) as the difference between v_i^S and v_i^N:

$$MBS_i(n,p,q) = v_i^S(n,p) - v_i^N(n,q). \qquad (3)$$

2.2 Privately provided public good

If the sharing cost is low enough, nodes will share to obtain the offloading benefit.

Lemma 1. $MBS_i(n,p,q) > 0.$

[4]We take a game-theoretic approach to studying incentives in P2P networks, and will use graph theory in the next section, so we use *peer*, *node* and *player* interchangeably.

[5]The number of trials is $n-2$ because there are n downloading agents, but the set of possible *other* agents than i downloading from s excludes i and s.

[6]Equations (1) and (2) are a simplified approximation. The difference is qualitatively unimportant; see `http://www-personal.umich.edu/\~jmm/papers/NetEcon06-supp-appendix.pdf`.

Proof. See appendix. □

Lemma 1 implies if the cost of sharing, c, is less than $MBS(n,p,q)$, there is a dominant strategy equilibrium in which all peers choose to share. This verifies that our model is consistent with the results in Krishnan et al. [2004].

2.3 How large is the offloading benefit?

It is straightforward to show that MBS is decreasing in the number of other sharing nodes, k, so the incentive for a marginal node to share decreases in larger networks. But the equilibrium outcome of the game depends on the relative values of c and MBS. Without an empirical estimate for c, it is difficult to determine whether the offload effect is meaningful for a P2P network. We can, however, gain an appreciation for the magnitude of the offloading effect by analyzing it as a percentage increase in a non-sharing peer's utility. Denote this increase by G_i:

$$G_i(n,p,q) = \frac{MBS(n,p,q)}{v_i^N(n,q)} \tag{4}$$

Lemma 2 below characterizes the asymptotic properties of G_i. As $n \to \infty$, G_i converges to $\frac{1}{k}$. For example, when $k = 30$, by sharing her content, a player can only increase the probability of obtaining one unit of content by 3.3%. It seems implausible that in medium or large networks this small gain would motivate many peers to share their content. Further, since $k \leq n$ by definition, we see that G_i converges to zero as k increases, which means the benefit of sharing vanishes the more other peers are sharing.

Lemma 2.
$$\lim_{n\to\infty} G_i = \frac{1}{k}.$$

Proof. See appendix. □

In Figure 1 we plot G_i against n, for various small values of k. This illustrates our point that the gain of sharing becomes independent of the number of peers in large networks, and it decreases in the number of nodes that are sharing their content. We conclude that although the offloading effect may play some role in P2P networks, the private incentives it suggests are likely insufficient to motivate equilibrium content sharing in large networks.

3 Indirect Private Incentives: Generalized Reciprocity

We turn to a different candidate explanation for sharing: generalized reciprocity in a repeated game. In BitTorrent, a form of direct reciprocity is implemented by embedding a tit-for-tat type of strategy in the client software [Cohen, 2003]. This provides a form of direct incentive for uploading, similar to the offloading incentive we studied in the previous section. One type of indirect incentive for contributing to the public good is "generalized reciprocity" [Putnam, 2000]:

> I'll do this for you without expecting anything specific back from you, in the confident expectation that someone else will do something for me down the road. (p. 21)

In a P2P network, generalized reciprocity may be loosely described as a cycle in the directed graph in which each peer contributes to second peer, but receives a contribution from a third peer. We shall formally characterize conditions on the topology of the graph such that it has an equilibrium in which some self-interested peers contribute while others free-ride. Generalized reciprocity can arise when direct reciprocity is impossible, for example when demands between node pairs are very asymmetric.

We suppose there is no private benefit from sharing (i.e., no altruism, and no offloading effect), but that peers are interconnected through a network topology, and anticipate participating for an indefinite length of time. Feigenbaum and Shenker [2002] suggested graph theory

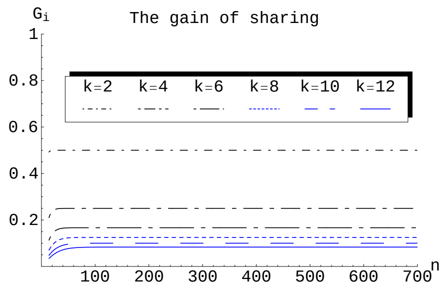


Figure 1: The gain of sharing for player i.

to model incentives in network problems because peer incentives might affect the formation of the graph. We follow Shneidman and Parkes [2003] who suggest graph configuration may affect incentive structures. In particular, we characterize a family of graphs that support a generalized reciprocity equilibrium. Like us, Afergan and Sami [2006] use repeated games theory to study problems on network topologies.

We illustrate with the simple graph in Figure 2. Each labeled, directed link represents the direction and volume of the traffic between the two end nodes of the link. In a repeated game of indefinite duration, B and C want a file from each other and both want a file from A. A, however, is a free rider. Suppose peers restrict themselves to either sharing with every node or not sharing at all.[7] Suppose further that the benefit of receiving one unit of content is significantly higher than the cost of sharing it. With these assumptions B and C sharing is an equilibrium as long as they are receiving enough content. If say, B stops sharing, C will find it unprofitable to share, hence will also stop sharing. Thus the network breaks down due to B's deviation.

3.1 Definitions and assumptions

We model peers' interactions as an infinitely repeated game with a fixed time discount factor δ adopted by all peers.[8] A set of demand relationships among the n peers in the network is given exogenously, and remains constant through out. These relationships can be represented as a directed graph, D. Loosely speaking, a directed graph is a set of *nodes* connected by directed *edges* [Deo, 1974]. A *path* is a sequence of consecutive nodes and edges, with no nodes repeated. A path which ends at the node it begins is called a *cycle*. Two cycles are *independent* if they do not share any nodes in common. A graph is *connected* if there is an undirected path connecting every pair of nodes. A graph that is not connected can be divided into connected *components*, each of which is a connected subgraph. For example, Figure 3 is a directed connected graph with independent cycles $(1, a_{12}, 2, a_{23}, 3, a_{34}, 4, a_{41})$ and $(10, a_{10_1 1}, 11, a_{11_1 0})$. The link label values are the demand quantities. The graph has an equivalent representation as an $n \times n$ adjacency matrix with each element σ_{ij} the demand from peer j to peer i.

3.2 Game setup

For simplicity, we define matrix B as a binary demand matrix obtained by converting the positive link intensities in D into 1, with element b_{ij} the demand from j to i. We focus on an arbitrary *component* of the graph with the adjacency matrix K associated with it. K is thus connected and consists of k players. K remains constant through out the game and its member peers have complete information of K. In each stage game, permissible actions for player i, a_{ij}, are defined as follows,

$$a_{ij} = \begin{cases} 1 & \text{if } \sigma_{ij} \neq 0 \text{ and } i \textit{ shares} \text{ to } j \\ 0 & \text{if } \sigma_{ij} = 0 \text{ or } i \text{ does not } \textit{share} \text{ to } j \end{cases}$$

In each round, the stage game is played and then pay-

[7]We remove this assumption for Proposition 2.

[8]With an appropriate increase to the discount rate, we can accommodate a finite but random time in the network, rather than an infinite horizon.

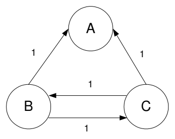


Figure 2: A simple example

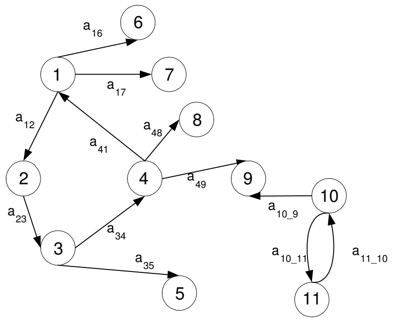


Figure 3: An example of equilibrium action graph

offs are realized. Players observe all players' actions in the previous round before the next round starts. Each peer receives a positive value of v when she obtains a unit of file, and incurs a positive cost of c when she shares a unit of file to any other peer.

In round t, the actions chosen by the peers constitute a $k \times k$ action matrix, A, which again corresponds to a directed graph. For node i, $\Sigma_{i=1}^{k} a_{ij}$ is its out-degree and $\Sigma_{i=1}^{k} a_{ji}$ is its in-degree. When $\Sigma_{i=1}^{k} a_{ij} = 0$, which means peer i does not share her file to any other peer, she is considered a *free-rider*; otherwise she is considered a *sharing peer*. We also define a parameter ρ_i as the ratio between i's out-degree and in-degree, $\rho_i = \frac{\Sigma_{i=1}^{k} a_{ij}}{\Sigma_{i=1}^{k} a_{ji}}$.

3.3 Equilibrium analysis

We propose that an action matrix A^* can be sustained in equilibrium, if its corresponding graph satisfies the following properties,

P.1 No nodes have an in-degree of zero, and $\max_i\{\rho_i\} < \frac{\delta v}{c}$.

P.2 Any two cycles in the graph are independent.

P.3 Any leaf node is connected to a node that participates in a cycle.

Figure 3 shows a graph that satisfies properties P.1–P.3. The condition on the out-degree/in-degree ratio in P.1 limits its consideration to networks in which users get enough net benefit that participating is better for them than is dropping out of the network. This condition can surely be relaxed to accommodate altruistic users. Property P.2 is purely to simplify the analysis, and we know from examples that there are networks in which users participate in more than one cycle and yet the result of our proposition still holds. We are working to relax this condition in ongoing research. Property P.3 rules out an agent who does not provide anything that "comes back around"; the node uploading to such an agent (e.g., node 6 uploading to some other leaf, say 12, in Figure 3) would always be better off to stop contributing since there is no generalized return on the contribution.

We claim that free-riding on such a graph may exist in equilibrium even without altruistic players or the offloading effect. We formalize this intuition in Proposition 1, as a *subgame perfect Nash equilibrium* (SPNE). A profile of strategies is a SPNE if it is a Nash equilibrium of the game itself, and if it induces a Nash equilibrium in every subgame [Fudenberg and Tirole, 1991].

Proposition 1. *In an infinitely repeated game with the afore-mentioned stage game, if the action matrix A satisfies property P.1–P.3, there exists a SPNE which can have both sharing peers and free-riders.*

Proof. See appendix. □

Thus even without altruistic peers, an offloading effect or an explicit incentive mechanism to encourage sharing, sharing can exist due to generalized reciprocity. Moreover, free-riding may exist too. The intuition is simple: peers care a lot about fulfilling their demands, and the cost of sharing is low, so they can tolerate free-riding to a certain extent. More free-riding does not occur because of the threat of a *local grim trigger strategy* (LGTS)[9]: if a node stops uploading to A, node A will leave the network forever, which through the generalized reciprocity cycle punishes the miscreant node, discouraging it from free-riding in the first place (see proof).

We derived Proposition 1 under restrictive conditions: all nodes could observe all flows (the *flow topology*), and nodes may only choose from a strategy space restricted to either upload to every requestor, or upload to none. These two assumptions taken together are clearly not very general: if nodes know the entire flow topology then why punish all requestors when a single node deviates? Likewise, if node 4 knows node 8 is a free-rider in Figure 3, why not cut off only node 8 rather than all nodes?

We are working on a model of generalized reciprocity with incomplete information about the flow topology, and with an unrestricted space of strategies. These assumptions seem reasonable for the pseudonymous Internet. We have one preliminary result that illustrates how generalized reciprocity can support P2P networks with equilibrium free-riding in more general settings. We assume peers only know the flows in which they participate and each peer selectively shares to other peers to maximize her value.

[9]Or others; LGTS is sufficient to support the equilibrium, but may not be unique.

Proposition 2. *In an infinitely repeated game, with only local knowledge of the flow graph and selective strategies, the flow graph depicted in Figure 4.(a) is a weak perfect Bayesian equilibrium.*

Proof. See appendix. □

4 Discussion

We have shown the existence of an equilibrium in a constrained family of network topologies, under two different game forms. Both cases are restrictive. We would like to characterize the *set* of equilibria to assess the plausibility of outcomes with a mixture of sharing and free-riding. Further, we would like to characterize other families of network topologies, to uncover those features (size, connectedness, overlapping cycles, etc.) that affect the equilibrium configurations. Of course, we would also like to address the question we asked about the offloading effect: is generalized reciprocity important enough to explain the amount of sharing we see in large networks?

Our ultimate goal is to use the model as a principled foundation to explore the design and performance of various incentive mechanisms to encourage sharing in P2P networks.

References

Eytan Adar and Bernardo A. Huberman. Free riding on Gnutella. *First Monday,* 5, 2000.

Mike Afergan and Rahul Sami. Repeated-game modeling of multicast overlays. In *INFOCOM*. IEEE, 2006.

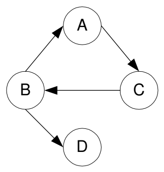


Figure 4: An equilibrium example

Theodore Bergstrom, Lawrence Blume, and Hal R. Varian. On the private provision of public goods. *Journal of Public Economics*, 29:25–49, 1986.

Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, CA, June 2003. Berkeley.

Narsingh Deo. *Graph Theory with Applications to Engineering and Computer Science*. Prentice-Hall, Englewood Cliffs, N.J., 1974.

Joan Feigenbaum and Scott Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *6th Intl Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 1 – 13, New York, NY, 2002. ACM Press.

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

Ramayya Krishnan, Michael D. Smith, Zhulei Tang, and Rahul Telang. The virtual commons: Why free-riding can be tolerated in file sharing networks. Carnegie-Mellon Univ., November 2004.

Robert D. Putnam. *Bowling Alone : The Collapse and Revival of American Community*. Simon & Schuster, New York, 2000.

Kavitha Ranganathan, Matei Ripeanu, Ankur Sarin, and Ian Foster. To share or not to share : An analysis of incentives to contribute in collaborative file-sharing environments. In *Workshop on Economics of Peer-to-Peer Systems*, CA, June 2003. Berkeley.

Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networking 2002*, January 2002.

Jeffrey Shneidman and David C. Parkes. Rationality and self-interest in peer to peer networks. In *2nd Int. Workshop on Peer-to-Peer Systems*, 2003.

APPENDIX

Proof of Lemma 1

$$\begin{aligned} v_i^S(n,p) &= \frac{1-(1-p)^{n-1}}{(n-1)p} \quad (5)\\ &= \frac{1-(1-p)^{n-1}}{(n-1)(1-(1-p))}\\ &= \frac{1+(1-p)+\cdots+(1-p)^{n-2}}{n-1}. \end{aligned}$$

Here we used the sum of a geometric series $\sum_{k=0}^{n} r^k = \frac{1-r^{n+1}}{1-r}$. Let $I_{n,p} = 1+(1-p)+\cdots+(1-p)^{n-2}$. Then $v_i^S(n,p)$ simplifies to $v_i^S(n,p) = I_{n,p}/(n-1)$. Similarly, let $I_{n,q} = 1+(1-q)+\cdots+(1-q)^{n-2}$, then $v_i^N(n,q)$ can be written as $v_i^N(n,q) = I_{n,q}/(n-1)$. Since $p = \frac{1}{k+1} < q = \frac{1}{k}$, $1-p > 1-q$ and $(1-p)^x > (1-q)^x$, $x = 0,\cdots,n-2$. Therefore $MBS(n,p,q) > 0$.

Proof of Lemma 2

$$\begin{aligned} &\lim_{n\to\infty} \frac{MBS(n,p,q)}{u_i^N(n,q)}\\ &= \lim_{n\to\infty} \frac{\frac{1-(1-p)^n}{p}}{\frac{1-(1-q)^n}{q}} - 1 = \frac{1}{k}. \end{aligned}$$

Proof of Proposition 1

Two types of nodes in any graph A satisfy properties P.1 $\sim$ P.3. We label the nodes on the cycle as *cycle nodes* and the nodes that do not have child nodes as *leaf nodes*. We restricts peers in each round to play either Share or Not Share with all demanding nodes: if i plays Share, $a_{ij} = b_{ij}, \forall j \neq i$; and if i plays Not Share, $a_{ij} = 0, \forall j$.

We consider two peer strategies, Not Share and the local grim trigger strategy (LGTS). In LGTS peer i plays Share in the first round and continues sharing as long as $\rho_i < \frac{\delta v}{c}$. We show that a strategy profile in which the cycle nodes play LGTS and the leaf nodes play Not Share is a SPNE. First, leaf nodes, by playing Not Share while receiving value from their parent nodes do not have any incentive to deviate.

Second, cycle node i can either follow LGTS or deviate by playing Not Share. We calculate the continuation payoffs of each from round t onwards, as u_i^t. If she follows the equilibrium strategy, LGTS, her continuation payoff is,

$$u_i^t = v\Sigma_{i=1}^k a_{ji} - c\Sigma_{i=1}^k a_{ij} \quad (6)$$

If player i deviates from LGTS in round t, the other nodes in the same cycle will know before round $t+1$ that she has deviated. Therefore in round $t+1$ no nodes will share to her. This is due to P.2, which implies that a peer belongs to no more than one cycle, such that once one peer deviates from LGTS, the cycle is going to be broken. Foreseeing this happening, no peer in the cycle will share in round $t+1$. Thus player i's continuation payoff is,

$$u_i^t = (1-\delta)v\Sigma_{i=1}^k a_{ji} \quad (7)$$

The cycle nodes will choose to follow LGTS if the following inequality holds,

$$v\Sigma_{i=1}^k a_{ji} - c\Sigma_{i=1}^k a_{ij} > (1-\delta)v\Sigma_{i=1}^k a_{ji}, \quad (8)$$

which is equivalent to ,

$$\rho_i < \frac{\delta v}{c} \quad (9)$$

Inequality (9) is satisfied by property P.1. For completeness, we can easily verify that once a cycle node or a leaf node has deviated, there is no incentive for her to return to the equilibrium strategy. In this equilibrium, the cycle nodes are sharing peers and the leaf nodes are free-riders.

If P.3 is not satisfied, then there will be a third type of node in the graph: a sub-leaf node that receives a file from a leaf node, but does not participate in a cycle.[10] A graph with sub-leaf nodes will not be an SPNE because the leaf node providing only to sub-leaves will be unambiguously better off not uploading any files, and thus will deviate from the proposed equilibrium.

Proof of Proposition 2 (sketch)

Suppose all the peers adopt individual grim-trigger strategy (IGTS). Whenever a pair of peers each demand one

[10]The sub-leaf may upload a file to another node, but at some point a node in that chain will be a terminal leaf node.

unit of content from each other, IGTS requires they start by sharing with each other and stop sharing forever if one has deviated in the previous period. Suppose further that a node has diffuse (uniform) priors over the distribution of possible flow topologies (each possibility is equally likely). These beliefs will be sustained in equilibrium because there are no moves by nature and the problem is stationary so there are no changes in flows that are informative about the unobserved links. The proof follows 5 steps:

Step.1 Show that node B in Figure 5(a) will choose to stop sharing to C. This can be done by examining Figure 5(c), (d), (e), and (f), which represent all of B's possible beliefs. Given that each peer is individually rational, in (c), (e), and (f) B can gain by cutting off C. In (d), B can gain by cutting off either A or C. Thus it is profitable for B to stop sharing to C.

Step.2 Following the same logic in Step. 1, show that node B in Figure 5.(b) will choose not to deviate and continue sharing to A.

Step.3 To show that Figure 4 can be sustained in a weak perfect Bayesian equilibrium, we only need to show that node B in both Figure 6.(a) and (b) will not deviate, since these two cases represent scenarios for all nodes in Figure 4.

Step.4 In Figure 6.(a), B only knows the links that she participates in, and the total number of nodes in the graph. There are 64 possible flow graphs in total, out of which only 28 are rational according to the results of Step.1 and 2.[11] In 19 scenarios B gains by cutting off either A or D. And in 2 cases B gains by not sharing to any nodes — she free-rides. Thus the gain of cutting off A or D is $\frac{19}{28} \times (v-c) - (v-2c)$.[12] And the gain of free-riding is $\frac{2v}{28} - (v-2c)$. Apparently the gain of free-riding is too small to be interesting. And only if $c > \frac{9v}{37}$, it is profitable for B to cut off A or D. Assuming that v is sufficiently larger than c, this condition is not satisfied. Hence the most profitable strategy is not to deviate.

Step.5 Similar logic applies to Figure 6.(b).

[11]see http://www-personal.umich.edu/\~jmm/papers/NetEcon06-supp-appendix.pdf for an analysis of all 64 graphs.

[12]This is an approximate calculation for illustration purpose only. For detailed calculations please refer to http://www-personal.umich.edu/\~jmm/papers/NetEcon06-supp-appendix.pdf.

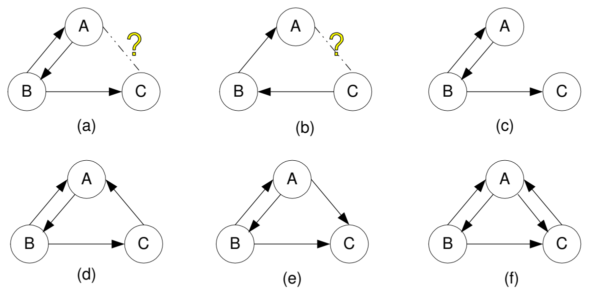


Figure 5: Three Node Scenarios

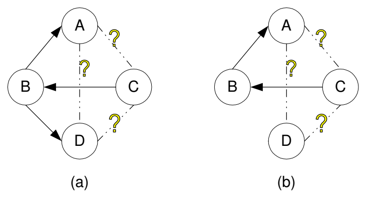


Figure 6: Node B's Beliefs

Peer-to-Peer Filesharing and the Market for Digital Information Goods[*]

Ramon Casadesus-Masanell[†] Andres Hervas-Drane[‡]

May 8, 2006

Abstract

Existing models of peer-to-peer (p2p) filesharing assume that individuals are concerned with each others' wellbeing. Without social preferences (*i.e.*, altruism or reciprocity), peers are better off freeriding whenever the cost of sharing content is larger than that of not sharing. In the absence of social preferences this public-goods problem results in the collapse of the p2p network. Because p2p networks are composed of millions of individuals who interact anonymously, we find inadequate the assumption that peers care about each others' utility. We present microfoundations for a stylized model of a p2p network where all peers are endowed with standard preferences and show that the resulting endogenous structure of the p2p network is conducive to sharing content by a significant number of peers, even if sharing is costlier than freeriding. Selfish utility-maximizing peers are better off sharing because by doing so they face less congestion. We characterize the endogenous level of sharing and present comparative statics results. We build on this framework to analyze the optimal strategy of a profit-maximizing firm, such as Apple's i-Tunes, that offers the same content available on the network. Contrary to the p2p network, the firm offers downloads on a traditional client-server architecture and sells content at positive prices. We show that the firm may be better off setting high prices, allowing the network to survive, and that the p2p network may work more efficiently in the presence of the firm than in its absence.

1 Introduction

Peers in peer-to-peer (p2p) networks face a fundamental choice between sharing content or freeriding. When a peer decides to share content –a costly activity– she is effectively supplying two different goods. On the one hand, she provides *content*. Obviously, the peer who shares does not benefit from the content that she is sharing as she already owns it. On the other hand she also supplies *upload bandwidth* and this may result in lower network congestion. Sharing results in lower congestion if upload bandwidth is a scarce resource. Based on the available empirical evidence, in this paper we assume this to be the case. The nature of peer-to-peer networks warrants that the provision of bandwidth benefits all peers equally in expected terms. In sum, peers face a trade off: by sharing they bear costs that could be avoided by freeriding, but sharing also reduces average network congestion and this benefits every peer, including the peer who shares.

Building on this insight, we construct a model where peers provide bandwidth in addition to content when they decide to share. Specifically, we consider a finite population of agents that derive positive and homogenous utility from digital content. Peers suffer disutility from the costs associated with downloading content. These costs are proportional to the time required to complete downloads, the level of *congestion*, which in turn depends on the bandwidth provision available in the network. Peers may reduce their expected congestion by providing upload bandwidth to other peers. We model this decision as a binary choice: share content or freeride. By having agents differ in their disutility of congestion (impatience or opportunity cost of time) we show that an endogenous level of sharing emerges in the network. Selfish utility-maximizing peers are better off sharing because by doing so they face less congestion. To the best of our knowledge, there is no earlier model of p2p filesharing with endogenous congestion where peers concerned solely about the impact of their actions on their own utility decide to share content.

We build on this framework to analyze the optimal strategy of a profit-maximizing firm that offers the same content available on the network at positive prices employing a traditional client-server architecture. In the absence of altruism towards artists, it is an interesting question why consumers pay to purchase licensed content online. Towards answering this question, we derive the shape of the demand function the firm faces and characterize its optimal

[*]This abridged draft was prepared for the 2006 Workshop on the Economics of Networked Systems (NetEcon06). An expanded version is available from the authors.
We thank seminar participants at the 2005 Bosphorus Workshop on Economics Design, the 2005 CoCombine conference on the Economics of the Internet, London Business School, Harvard Business School and MIT. We are grateful for financial support from the HBS Division of Research, the Fundación BBVA and the FPU program of the Spanish Ministry of Education and Science.
[†]Harvard Business School. (casadesus@gmail.com)
[‡]Universitat Autònoma de Barcelona. (andresonline@gmail.com)

pricing strategy. In essence, our framework points to the central role of 'convenience' when accessing and consuming digital content through the Internet.

The model captures important stylized facts identified by the literature. First, Asvanund et al. [2] show that congestion worsens with size as peer-to-peer networks grow. Our model endogenously generates this result. In fact, the effect of network size on congestion helps explain the coexistence of multiple different p2p networks. Second, many studies have shown that heavy users of p2p filesharing networks are more prone to purchase content online. Our framework not only suggests that there is no contradiction in this observed behavior, but also sheds light on the factors that explain the demand for online content in the presence of a p2p network. Third, we provide insights on content pricing and the effectiveness of industry initiatives such as suing heavy sharers. Finally, our model shows that filesharing networks strictly benefit from bandwidth infrastructure improvements. This suggests that filesharing is indeed a driver for broadband demand and helps explain why Internet service providers have not taken action to limit the spread of p2p applications and filesharing traffic load. We believe that our results should be of interest to all participants in markets for digital information goods.

The paper is organized as follows. Section 2 introduces the building blocks of our model of peer-to-peer filesharing and describes the game (in the absence of a profit maximizing firm). In Section 3 we present a simple approximation to the average congestion in an arbitrary peer-to-peer network. Section 4 derives the equilibrium network configuration and studies its properties. In Section 5 we further characterize the equilibrium under the assumption that peers' time preferences are independently drawn from a uniform probability distribution. Finally, in Section 6 we introduce a profit maximizing firm that competes against the p2p network and analyze the interdependencies that arise in the competition between both business models.

2 The model

We consider a population of M agents that derive utility from the consumption of digital information goods. They all value content equally and differ only in their disutility of congestion. We model the formation of a peer-to-peer network in two stages. In the first stage, agents choose (simultaneously) whether or not to join the network. Agents who choose to belong to the network can either share their content or freeride. Sharers offer their content on the network for download by other peers while freeriders do not. While sharing content is costly, some sharing is required for the network not to collapse as downloads can only be realized from other sharers. We will refer to agents in the network as *peers* and those outside as *outsiders*. We let $N \leq M$ denote the number of peers. $M - N$ is the number of outsiders.

In the second stage peers interconnect and downloads are realized. The utility of a peer that freerides is given by

$$u_i^f = u_d - (c_n + \rho_i)t_d, \tag{1}$$

and that of a peer who shares his content is

$$u_i^s = u_d - (c_n + c_s + \rho_i)t_d, \tag{2}$$

where $i \in \mathbf{N} = \{1, 2, ..., N\}$. Outside utility is normalized to zero.

The utility derived from content once a download has been completed is u_d and it is common across all agents. The time required to complete a download, t_d, is endogenous and depends on the level of congestion. A lower bandwidth transmission speed implies higher level of congestion resulting in higher download time. Every peer suffers a positive cost c_n of pertaining to the network. This captures the costs of the computing resources and the bandwidth for signalling traffic required to remain connected to the network until a download completes. Sharers additionally bear cost c_s. This is the cost originating from offering content for download on a public p2p network (including expected costs of legal action against the peer) as additional computing resources (storage space) and upload bandwidth is required.

Parameter $\rho_i \geq 0$ reflects the disutility of congestion experienced by peer i. The larger ρ_i is, the higher the disutility the peer obtains from an increase in the time required to complete a download. Hence ρ_i can alternatively be interpreted as impatience or opportunity cost of time: how much peer i values quick accessibility to content. Without loss of generality we choose indexes i so that $\rho_i \leq \rho_{i+1}$ for all i. All other costs being equal, peers would prefer to obtain the downloadable content immediately avoiding congestion delays. An increase in the time required to complete a download reduces the utility obtained from the network by increasing both the network costs and the disutility of congestion experienced by all peers.

To solve the second stage we let $\mathbf{S} \subset \mathbf{N}$ be the set of sharers in the network (given the agents' first-stage strategies) and denote by S the number of sharers (the cardinality of $\mathbf{S}$). A downloader exclusively served by a sharer will download a unit of content in time $\theta > 0$; that is, $t_d = \theta$. This can be interpreted as θ capturing the relation between the filesize of content and the bandwidth capacity available to peers. Thus an improvement in either encoding efficiency reducing filesizes or broadband infrastructure increasing bandwidth amounts to a reduction in θ. Download bandwidth is assumed not to be a limiting factor. If more than one downloader is connected to a given sharer, bandwidth is shared evenly amongst them. This can be interpreted as downloading taking place simultaneously or, alternatively, the sharer serving download queues for fractions of content by turns.

A set of links connecting peers to sharers where every peer connects to one sharer only and no sharer connects to herself is called a *network allocation.* A *stable* network allocation is one where no peer can be made strictly better off by connecting to a different sharer. We assume that following the first stage (where peers decide whether to share or to freeride) a stable network allocation ensues. Clearly, if the network allocation was not stable, at least one peer would have an incentive to connect to a different sharer.

The following mild assumption is required for the results: $u_d > (c_n + c_s + \rho_i)\theta$ for all i. This ensures that a p2p network with minimum congestion is always preferred to the outside option of not pertaining to the network. With the notation in place, we now proceed to solving the game by backwards induction.

3 Network foundation

Interconnection occurs in the second stage, after each peer has decided whether she will share or freeride. Congestion plays a crucial role in our development as peers choose to share taking into consideration the effect that their sharing has on congestion. Given a network allocation, the bandwidth obtained by peer $i \in \mathbf{N}$ can be computed as follows: if the peer is connected to a sharer to which k other peers are connected to, then peer i obtains effective bandwidth $1/(k+1)$.

Freeriders can connect to every sharer and, thus, have S possible links available to choose from. Sharers, on the other hand, cannot connect to themselves. As a consequence, sharers have $S-1$ possible links available. This implies that, in general, the expected congestion of sharers and freeriders will differ. To compute the *expected bandwidth* for freeriders and sharers in a network with N peers and S sharers, we begin by computing each peer's effective bandwidth in every stable network allocation. We then average these effective bandwidths assuming that every stable network allocation is equally likely. *Expected congestion*, the delay required to complete downloads, t_d, is the inverse of the expected bandwidth.

In another paper [3] we derive an exact expression for the expected effective bandwidth of sharers and freeriders. There, we show that S/N is a good approximation to the expected bandwidth of both sharers and freeriders. The accuracy of this approximation increases with the size of the network. In fact, already in a network of size $N = 10$, the expected effective bandwidth of sharers and freeriders differs from S/N by, at most, 0.0012. Given this result, we conclude that all peers obtain an expected download bandwidth close to S/N. This implies that the expected time to complete a download for all peers can be approximated by $t_d = \theta/\frac{S}{N} = \theta\frac{N}{S}$. It should be noted that although the expected bandwidth depends linearly in the number of sharers, the time required to complete a download does not. This property is crucial to our results. Technically, it ensures that our objective function is concave in S, allowing for interior equilibria in which sharing and freeriding may coexist for certain ranges of N.

4 Equilibrium network configurations

In this section we analyze the first stage of the game. Every peer i chooses whether to freeride or to share content (at additional cost c_s). In making their decision, peers consider the effect of their choice on expected download time $\theta\frac{N}{S}$. Equations (1) and (2) imply that if expected download time was *not* affected by the sharing decision, no peer would ever share and the peer-to-peer network would not be viable.

In this section we take N as given. This amounts to assuming that all N peers in the network obtain positive utility. In general, this will depend on S and the distribution of ρs. In the following section we relax this assumption and let peers decide whether or not to join the network.

Let $P = \{\mathbf{F}, \mathbf{S}\}$ be a partition of $\mathbf{N}$. We refer to P as a *network configuration.*[1] $\mathbf{F}$ is the set of freeriders and $\mathbf{S}$ the set of sharers. Obviously, P constitutes a Nash equilibrium if no $i \in \mathbf{S}$ prefers to (unilaterally) become a freerider and no $j \in \mathbf{F}$ prefers to become a sharer.

Proposition 1 *Every equilibrium network configuration $P = \{\mathbf{F}, \mathbf{S}\}$ has the following form: $\mathbf{F} = \{1, 2, ..., n-1\}$ and $\mathbf{S} = \{n, n+1, ..., N\}$ for some $n \in \mathbf{N}$. The system of equations given by Γ_s identifies the set $\mathbf{S}$ for all equilibrium network configurations,*

$$\Gamma_s = \{i \in \mathbf{I} \,|\, H(\rho_i) \subset G(\rho_i)\},$$

where

$$\begin{aligned} G(\rho_i) &= \left\{ k \in \mathbf{I} \,\middle|\, \frac{c_f + \rho_{i-1}}{c_s} \leq k \leq \frac{c_f + c_s + \rho_i}{c_s} \right\} \\ H(\rho_i) &= N + 1 - i. \end{aligned}$$

Proof. All proofs are in the appendix. ■

The proposition says that if peer i is a sharer in equilibrium network configuration P, then peer $i+1$ must also be a sharer. Moreover, if peer j is a freerider, then peer $j-1$ must also be a freerider. Thus, the most impatient peers prefer to share while

[1] Notice that a network configuration can be mapped to many different network allocations.

the more patient peers are better off freeriding. The reason is simple: by sharing content, peers reduce congestion and the (positive) marginal effect on peer utility implied by lower congestion is proportional to the value of ρ_i. Peers for whom the opportunity cost of time is high, are more inclined to share. This is true even though given any *fixed* level of congestion, all peers (regardless of the value of ρ) are better off freeriding than sharing.

The system of equations $\{G(), H()\}$ characterizes the equilibrium network configurations by pinning down to the fullest possible extent the set of sharers $\mathbf{S}$. Note that certain parameter constellations may exhibit multiple equilibria and Γ_s may not be a singleton.

Let $\mathbf{S} = \{n, n+1, ..., N\}$ be the set of sharers in an equilibrium network configuration. We refer to the case $n = 1$ as a *full-sharing network configuration* (or *full-sharing equilibrium*) and to the case $n > 1$ as a *partial-sharing network configuration* (or *partial-sharing equilibrium*). In a full-sharing network configuration all peers are sharers. In this case, congestion is minimized as the expected download time for all peers (t_d) is equal to θ.

Remark 2 *Full-sharing holds in the network if and only if*

$$N < \frac{c_n + c_s + \rho_1}{c_s}.$$

Therefore, if N is sufficiently small, the unique equilibrium network configuration has all peers sharing content. Notice that as the incremental cost of sharing c_s approaches zero, the maximal network size that supports full sharing grows without bound. When N is large, the equilibrium network configurations will typically entail partial sharing. In this case, expected download time will be larger than θ for all peers.

5 Equilibrium with $\rho_i \sim U[0, \bar{\rho}]$

In Section 4 we have characterized all equilibrium network configurations for the general case, without specific assumptions on the distribution of ρ_is or the cardinality of $\mathbf{N}$. In order to ensure tractability when we introduce a profit maximizing firm (Section 6), we make the additional assumption that ρ_is are i.i.d. $U[0, \bar{\rho}]$. This allows us to further characterize the set of equilibrium network configurations.

The first result shows that if the network has many peers, then the set of equilibrium configurations is a singleton.

Remark 3 *For N large enough, there is a unique equilibrium network configuration.*

The next result identifies the most patient sharer as a function of the parameters. Identifying precisely the most patient sharer will allow us to easily analyze how the different parameters affect network congestion. In particular, we are interested on the effect that N has on congestion. If congestion decreases when N grows, then the p2p network becomes gradually more valuable as the number of peers expands. If, in contrast, network congestion grows with N, then the p2p network exhibits negative (network) externalities.

Proposition 4 *Let $\rho_{s(N)}$ be the most patient sharer in equilibrium. Then, for N large,*

$$\rho_{s(N)} \simeq \frac{\bar{\rho}\left((N-1)\, c_s - c_n\right)}{\bar{\rho} + N c_s},$$

and the cardinality of $\mathbf{S}$ *in equilibrium is given by*

$$S = \frac{\bar{\rho} + c_s + c_n}{\frac{1}{N}\bar{\rho} + c_s}.$$

Notice that $\rho_{s(N)}$ is increasing in N. This implies that the larger the cardinality of $\mathbf{N}$, the lower is the proportion of sharers in equilibrium. In other words, in our model the p2p network exhibits negative network externalities (past the threshold network size of full sharing): the larger the number of peers in the network, the lower the average utility that peers obtain. In fact, as $N \to \infty$, $\rho_{s(N)} \to \bar{\rho}$. Therefore, when the network is very large, only the most impatient peer winds up sharing content. Also note that $S < \infty$, even as $N \to \infty$. Therefore, not only the proportion of sharers dwindles as N grows, but the absolute number of sharers has a cap. As a consequence, the expected download time for all peers ($\theta \frac{N}{S}$) grows without bound as N increases. This means that as N grows, the peer-to-peer network becomes less and less attractive. We will now see that this has important implications for the equilibrium pricing strategy of a profit maximizing firm competing for customers against a p2p network.

6 The firm

We next introduce the problem of an online firm selling digital information goods also available on the peer-to-peer network. To the firm, the network is a competitor because peers that choose to download files from the network could otherwise become paying customers. Because content is free on the p2p network, for the firm to persuade users of digital content to purchase it at positive price, it must offer added benefits that the p2p network cannot match. In our view, the most important advantage of the firm is that it can offer lower download time than the network. Specifically, we let the firm offer content streaming based on traditional client-server architecture. That is, consumption of content acquired through the firm can be realized immediately at the moment of purchase. As a consequence, the utility of buyers is:

$$u_i = u_d - p. \tag{3}$$

Notice that (3) is the natural adaptation of (1) and (2) to the case of streaming. With streaming, the expected download time t_d falls down to zero. Thus, the terms $c_n + \rho_i$ and c_s do not appear in (3). On the other hand, the firm charges a positive price for content. We assume that the content offered by the firm is the same as that shared in the p2p network.

Agents may now choose to purchase content instead of downloading it off the network at zero price. We modify the timing of the game accordingly. In the first stage, the firm chooses the price at which content is sold. In the second stage, agents choose to either purchase from the firm, enter the network, or stay outside and not consume content. Agents who enter the network may share or freeride. In the third stage, agents in the network interconnect and downloads take place.

We assume that the firm faces zero marginal costs. All infrastructure and running costs related to the service are fixed and independent of the activity level. The assumption captures the fact that selling additional copies of digital content has negligible incremental costs.

The problem of the firm is to quote the price p that maximizes profits. The following proposition summarizes the firm's optimal strategy as a function of the parameters.

Proposition 5 *Let*

$$\begin{array}{ll} p_{fc} := \theta(c_n + c_s) & \textit{(full market coverage)} \\ p_{hc} := \frac{1}{2}\theta(\bar\rho + c_n + c_s) & \textit{(high market coverage)} \\ p_{lc} := \frac{1}{2}\theta(\bar\rho + Mc_s) & \textit{(low market coverage)} \\ p_{oc} := u_d & \textit{(outsiders only coverage)} \end{array}$$

The optimal pricing strategy is given by:

- *If* $\bar\rho > c_n + c_s$, *then*

$$\begin{array}{lll} p_{hc} & \textit{if} & M < M_b \\ p_{lc} & \textit{if} & M_b \le M < M_c \\ p_{oc} & \textit{if} & M \ge M_c \end{array}$$

- *If* $\bar\rho \le c_n + c_s$, *then*
  - *If* $u_d \le 2\theta(c_n + c_s)$, *then*

$$\begin{array}{lll} p_{fc} & \textit{if} & M < M_d \\ p_{oc} & \textit{if} & M \ge M_d \end{array}$$

  - *If* $u_d > 2\theta(c_n + c_s)$, *then*

$$\begin{array}{lll} p_{hc} & \textit{if} & M < M_b \\ p_{lc} & \textit{if} & M_b \le M < M_c \\ p_{oc} & \textit{if} & M \ge M_c \end{array}$$

where $M_a = \frac{4(c_n+c_s)-\bar\rho}{c_s}$, $M_b = \frac{(c_n+c_s)(2\bar\rho+c_n+c_s)}{\bar\rho c_s}$, $M_c = \frac{2u_d-\theta\bar\rho}{\theta c_s}$ *and* $M_d = \frac{u_d^2-\theta\bar\rho(u_d-\theta(c_n+c_s))}{\theta c_s(u_d-\theta(c_n+c_s))}$.

Equilibrium profits are:

$$\begin{array}{ll} \pi_{fc} = M\theta(c_n + c_s) & \textit{(full market coverage)} \\ \pi_{hc} = \frac{M\theta(\bar\rho+c_n+c_s)^2}{4\bar\rho} & \textit{(high market coverage)} \\ \pi_{lc} = \frac{1}{4}M\theta(\bar\rho + Mc_s) & \textit{(low market coverage)} \\ \pi_{oc} = u_d(1 - \frac{u_d}{\theta(\bar\rho+Mc_s)})M & \textit{(outsiders only coverage)} \end{array}$$

The firm's demand curve is downward sloping as expected. Agents with sufficiently high disutility of congestion prefer to purchase instead of belonging to the network. The lower the price the firm quotes, the higher the number of agents who prefer to purchase. The agents who obtain higher surplus from purchasing are those with larger values of ρ, agents who may have potentially remained outsiders in the absence of the firm. The size of the network is affected by the presence of the firm, as only those individuals with lower disutility of congestion remain as peers; agents who would otherwise share content may now leave the p2p network and purchase from the firm. As sharers leave, peers that might have otherwise been freeriders are now better off sharing.

The demand curve exhibits a non-derivability. Two ranges exist over which congestion in the network differs. In the lower price range, full sharing holds and congestion is not affected by peers entering or exiting to purchase. In the higher price range, only partial sharing holds. In this case congestion varies with the size of the network and this effect is taken into account by agents; the smaller the network, the lower the level of congestion. As an example, consider the effect of a reduction in price. In the full sharing range, peers who switch to purchase are not affecting the congestion experienced by those remaining. But in the case of partial sharing, peers leaving are (indirectly) reducing congestion by reducing the size of the network. This effect ensures that less peers will react to a price reduction in the case of partial sharing.

Proposition 5 shows that the optimal pricing strategy depends critically on market size. The firm will only quote a low price and cover the entire market if it is sufficiently small. The bigger the market, the more profitable it is for the firm to target agents with a higher disutility of congestion by quoting higher prices. If the market is sufficiently large, it is optimal for the firm to serve outsiders exclusively. The intuition for this result lies on the mechanisms that drive congestion in a peer-to-peer network. As the size of the network increases, so does the congestion experienced by all peers. As a consequence, the surplus that the firm can extract by targeting agents who most suffer congestion grows more than that obtained by covering the whole market by quoting a low price. The fact that this result arises under a uniform distribution of ρ suggests that it is quite general.

For the case of a large market, we should expect the firm to quote a high price, close to the actual valuation of content. Such a pricing strategy does indeed internalize the presence of the network. As a result, the firm will not directly affect the size of the network, as agents who purchase would otherwise choose to stay outside, although the profits of the firm would strictly increase if the p2p network did not exist. The strategic effect of the presence of a p2p network on the firm can be likened to a low quality firm competing against a vertically differentiated competitor. The firm has strong incentives to offer high quality service by investing to minimize congestion and to quote a high price. The net effect of the network's presence on consumer welfare is unambiguously positive.[2]

It is of interest to look at the effect on firm's profit of changes in the technology parameter θ and the marginal cost of sharing c_s. A technology improvement captured by a decrease in θ strictly decreases the firm's profit. All other factors equal, a technology improvement in either broadband infrastructure or the efficiency of digital encoding reduces effective download delays in the network making it more attractive. Similarly, profits are increasing in the marginal cost of sharing under all market configurations. Because c_s measures the efficiency of the network, or how well the network scales with size, a higher marginal cost of sharing implies higher levels of congestion. This benefits the firm by generating higher surplus to be extracted out of potential purchasers. These effects provide strong incentives for the firm to intervene. Strategies that may serve this purpose include traffic discrimination on broadband networks, prioritizing the firm's data, and randomly suing sharers thereby increasing the expected cost of sharing for all agents.

References

[1] Antoniadis, P. and Courcoubetis, C. and Mason, R. (2004), 'Comparing Economic Incentives in Peer-to-Peer Networks,' *Computer Networks*, Vol.46, pp.133-146.

[2] Asvanund, A. and Clay, K. and Krishnan, R. and Smith, M. D. (2004), 'An Empirical Analysis of Network Effects in Peer-to-Peer Music-Sharing Networks,' *Information Systems Research*, Vol.15, pp.155-174.

[3] Creus Mir, A. and Casadesus-Masanell, R. and Hervas-Drane, A. (2006), 'Bandwidth Allocation in Peer-to-Peer Filesharing Networks,' Mimeo, Harvard Business School.

[4] Cunningham, B. M. and Alexander, P. J. and Adilov, N. (2004), 'Peer-to-Peer File Sharing Communities,' *Information Economics and Policy*, Vol.16, Issue 2, pp.197-221.

[5] Feldman, M. and Lai, K. and Chuang, J. and Stoica, I., 'Quantifying Disincentives in Peer-to-Peer Networks,' 1st Workshop on Economics of Peer-to-Peer Systems (2003).

[6] Feldman, M. and Papadimitriou, C. and Chuang, J. and Stoica, I. (2004), 'Free-riding and Whitewashing in Peer-to-Peer Systems,' PINS '04: Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems, pp.228-236.

[7] Golle, P. and Leyton-Brown, K. and Mironov, I. and Lillibridge, M. (2001), 'Incentives for Sharing in Peer-to-Peer Networks,' *Lecture Notes in Computer Science*, Vol.2232, pp.75+.

[8] Krishnan, R. and Smith, M. D. and Tang, Z. and Telang, R. 'The Virtual Commons: Understanding Content Provision in Peer-to-Peer File Sharing Networks,' November 2004.

[2]It is important to stress that in our framework the production of content is unaffected by the free exchange of content between peers in the p2p network. If lower royalties imply lower incentive by artists to produce quality content, then the net effect of the presence of the p2p network is ambiguous.

A Appendix

Proof of Proposition 1. Sharer $i \in \mathbf{S}$ will not free ride iff:

$$u_d - (c_f + c_s + \rho_i)\,\theta \frac{N}{S} \geq u_d - (c_f + \rho_i)\,\theta \frac{N}{S-1}, \quad (4)$$

or

$$\frac{S}{S-1} \leq \frac{c_f + \rho_i}{c_f + c_s + \rho_i}.$$

Notice that

$$\frac{d\left(\frac{c_f+\rho_i}{c_f+c_s+\rho_i}\right)}{d\rho_i} = \frac{c_s}{(c_f + c_s + \rho_i)^2} > 0. \quad (5)$$

Therefore, if (4) is satisfied for sharer $i \in \mathbf{S}$ it is also satisfied for all sharers i' with $\rho_{i'} \geq \rho_i$. Thus, the more impatient a sharer is, the less the incentive to become a freerider.

A freerider $j \in \mathbf{F}$ will not want to become a sharer iff:

$$u_d - (c_f + \rho_j)\,\theta \frac{N}{S} \geq u_d - (c_s + \rho_j)\,\theta \frac{N}{S+1}, \quad (6)$$

or

$$\frac{S}{S+1} \geq \frac{c_f + \rho_j}{c_f + c_s + \rho_j}.$$

Notice that (5) implies that if (6) is satisfied for peer $j \in \mathbf{F}$ it is also satisfied for all peers $j' \in \mathbf{F}$ with $\rho_{j'} \le \rho_j$. Thus, the more patient a freeriding peer is, the less the incentive to become a sharer.

We now further characterize the equilibrium network configurations by pinning down to the fullest possible extent the cardinality of S. Let $P = \{\mathbf{F}, \mathbf{S}\}$ be an equilibrium network configuration. Let ρ_i be the most patient sharer in $\mathbf{S}$. Equations (4) and (6) imply that

$$S \le \frac{c_f + c_s + \rho_i}{c_s} \qquad \text{and} \qquad S \ge \frac{c_f + \rho_{i-1}}{c_s}.$$

Thus,

$$\frac{c_f + \rho_{i-1}}{c_s} \le S \le \frac{c_f + c_s + \rho_i}{c_s}. \tag{7}$$

Let $\mathbf{I}$ be the set of integers. The following two objects are useful in what follows:

$$G(\rho_i) = \left\{ k \in \mathbf{I} \,\middle|\, \frac{c_f + \rho_{i-1}}{c_s} \le k \le \frac{c_f + c_s + \rho_i}{c_s} \right\} \tag{8}$$

and

$$H(\rho_i) = N + 1 - i. \tag{9}$$

Correspondence G indicates the cardinality of $\mathbf{S}$ if the sharer with lowest impatience has time preference parameter ρ_i. Function H tells us the number of peers with parameter ρ_j larger than or equal to that of peer i. The solution to the system of equations given by G and H pins down the set of most patient sharers for all equilibrium network configurations:

$$\Gamma_s = \{ i \in \mathbf{I} \,|\, H(\rho_i) \subset G(\rho_i) \}.$$

Because $G(\rho_i)$ is a correspondence, Γ_s may not be a singleton. Clearly, the cardinality of the set of equilibrium network configurations, coincides with that of Γ_s. ■

Proof of Remark 2. For a full-sharing network configuration to obtain, every peer must realize higher utility sharing than freeriding. In particular, the most patient peer (ρ_1) must be better off sharing than freeriding (given that everybody shares):

$$u_d - (c_n + c_s + \rho_1)\,\theta \ge u_d - (c_n + \rho_1)\,\theta \frac{N}{N-1}.$$

Solving for N we obtain:

$$N \le \frac{c_n + c_s + \rho_1}{c_s}. \tag{10}$$

■

Sketch of Proof of Remark 3. Recall that the set of equilibrium network configurations is given by

$$\Gamma_s = \{ i \in \mathbf{I} \,|\, H(\rho_i) \subset G(\rho_i) \}.$$

H is a decreasing function of ρ_i. G is an increasing correspondence. However, when N is large ρ_{i-1} is close to ρ_i. In fact as $N \to \infty$, $|\rho_{i-1} - \rho_i| \to 0$. Thus, when N is large,

$$G(\rho_i) = \left\{ k \in \mathbf{I} \,\middle|\, \frac{c_f + \rho_i}{c_s} \le k \le \frac{c_f + \rho_i}{c_s} + 1 \right\}.$$

Given this, $G(\rho_i)$ is single-valued except at those ρ_i such that $\frac{c_f + \rho_i}{c_s}$ is a natural number. Suppose that at ρ_i', $\frac{c_f + \rho_i'}{c_s} \in \mathbf{I}$. Then, for all $\varepsilon > 0$, $\frac{c_f + \rho_i' + \varepsilon}{c_s} \notin \mathbf{I}$ and $\frac{c_f + \rho_i' - \varepsilon}{c_s} \notin \mathbf{I}$. Thus, $G(\rho_i)$ is a step function with 'continuous jumps' at $\rho_i \in [0, \bar{\rho}]$ such that $\frac{c_f + \rho_i}{c_s} \in \mathbf{I}$. As a consequence, Γ_s is a singleton. ■

Proof of Proposition 4. We look for $\rho_{s(N)}$ such that the set of peers with $i \ge s(N)$ all want to share. Because $\rho_{s(N)}$ is the most patient sharer, the cardinality of the set of sharers is $S = N - s(N) + 1$.

For $\mathbf{S}$ to be the set of sharers of a stable partition, we need that the most patient sharer does not want to freeride:

$$u_d - \left(c_n + c_s + \rho_{s(N)}\right)\theta \frac{N}{S} \ge u_d - \left(c_n + \rho_{s(N)}\right)\theta \frac{N}{S-1}$$

This expression implies that

$$\rho_{s(N)} \ge (N - s(N))\, c_s - c_n.$$

Therefore, for $S = N - s(N) + 1$ to be stable, $\rho_{s(N)}$ must satisfy $\rho_{s(N)} \ge (N - s(N))\, c_s - c_n$.

We also need that the most impatient freerider does not want to share:

$$u_d - \left(c_n + \rho_{s(N)-1}\right)\theta \frac{N}{S} \ge u_d - \left(c_n + c_s + \rho_{s(N)-1}\right)\theta \frac{N}{S+1}$$

This expression implies that

$$\rho_{s(N)-1} \le (N - s(N) + 1)\, c_s - c_n.$$

Suppose now that all ρ_is are drawn from a uniform distribution $\rho_i \sim U[0, \bar{\rho}]$. When N is large we have that $s(N) - 1 \simeq \frac{\rho_{s(N)-1}}{\bar{\rho}} N$. Furthermore, large N also implies that $\rho_{s(N)} \simeq \rho_{s(N)-1}$. Therefore $s(N) \simeq \frac{\rho_{s(N)-1}}{\bar{\rho}} N + 1 \simeq \frac{\rho_{s(N)}}{\bar{\rho}} N + 1$. Substituting in the expression above, we obtain

$$\begin{aligned} \left(N - \frac{\rho_{s(N)}}{\bar{\rho}} N - 1\right) c_s - c_n &\le \rho_{s(N)} \\ \frac{\bar{\rho}\left((N-1)\, c_s - c_n\right)}{\bar{\rho} + N c_s} &\le \rho_{s(N)}. \end{aligned}$$

When N is large we have that $s(N) - 2 \simeq \frac{\rho_{s(N)-2}}{\bar{\rho}} N$. Furthermore, large N also implies that $\rho_{s(N)-1} \simeq \rho_{s(N)-2}$. Therefore $s(N) - 2 \simeq \frac{\rho_{s(N)-2}}{\bar{\rho}} N \simeq \frac{\rho_{s(N)-1}}{\bar{\rho}} N$ or $-s(N) + 1 \simeq -\frac{\rho_{s(N)-1}}{\bar{\rho}} N - 1$. Now, substituting in the expression above, we obtain

$$\begin{aligned} \rho_{s(N)-1} &\le (N - s(N) + 1)\, c_s - c_n \\ \rho_{s(N)-1} &\le \frac{\bar{\rho}\left((N-1)\, c_s - c_n\right)}{\bar{\rho} + N c_s} \end{aligned}$$

So, when N is large we have that

$$\rho_{s(N)-1} \leq \frac{\bar{\rho}\left((N-1)\,c_s - c_n\right)}{\bar{\rho} + Nc_s} \leq \rho_{s(N)}.$$

We conclude that when N is large

$$\rho_{s(N)} \simeq \frac{\bar{\rho}\left((N-1)\,c_s - c_n\right)}{\bar{\rho} + Nc_s}.$$

To identify the cardinality of S, we have that $S = N - s(N) + 1$ and $s(N) - 1 \simeq \frac{\rho_{s(N)-1}}{\bar{\rho}} N$. Therefore,

$$\begin{aligned} S &= N - \frac{(N-1)\,c_s - c_n}{\bar{\rho} + Nc_s} N \\ &= N\left(\frac{\bar{\rho} + c_s + c_n}{\bar{\rho} + Nc_s}\right). \end{aligned}$$

■

Proof of Proposition 5. An agent with disutility of congestion ρ_i will only purchase from the firm if:

$$u_d - p \geq u_d - (c_n + c_s + \rho_i)t_d.$$

Because $t_d \geq \theta$ is positive, if the condition is satisfied for peer i it will also be satisfied for peer $i+1$. To solve for demand given a price p we proceed by identifying the indifferent buyer, denoted by ρ_b. If $p = u_d$, only outsiders buy from the firm, as all other agents obtain strictly positive utility in the network. If $p > u_d$ purchasing yields negative utility and the firm faces no demand. To obtain demand when $p \leq u_d$ we must solve for ρ_b, given by:

$$u_d - p = u_d - (c_n + c_s + \rho_b)t_d. \tag{11}$$

Because either full or partial sharing may hold in the network, we consider two separate cases. We begin with the partial sharing case. Substituting $t_d = \theta \frac{N}{S}$ in (11) and taking into account that congestion will depend on ρ_b, as only agents such that $\rho_i \leq \rho_b$ are present in the network:

$$u_d - p \simeq u_d - (c_n + c_s + \rho_b^{ps})\,\theta \frac{N\left(\rho_b^{ps}\right)}{S\left(\rho_b^{ps}\right)},$$

where

$$N\left(\rho_b^{ps}\right) = \frac{\rho_b^{ps}}{\bar{\rho}} M,$$

and

$$S\left(\rho_b^{ps}\right) = N\left(\rho_b^{ps}\right)\left(\frac{\rho_b^{ps} + c_s + c_n}{\rho_b^{ps} + N\left(\rho_b^{ps}\right) c_s}\right).$$

Solving for ρ_b^{ps} yields:

$$\rho_b^{ps} = \frac{p\bar{\rho}}{\theta(\bar{\rho} + Mc_s)}.$$

We next consider the full sharing case and solve for the indifferent buyer by substituting $t_d = \theta$ in (11):

$$u_d - p \simeq u_d - \left(c_n + c_s + \rho_b^{fs}\right)\theta,$$

thus

$$\rho_b^{fs} = \frac{p - \theta(c_n + c_s)}{\theta}.$$

The demand function for the firm is given by:

$$D = (1 - \frac{\rho_b}{\bar{\rho}})M. \tag{12}$$

Substituting ρ_b^{ps} we obtain the expression for demand in the partial sharing range:

$$D^{ps} = (1 - \frac{p}{\theta(\bar{\rho} + Mc_s)})M.$$

And substituting ρ_b^{fs} in (12) we obtain demand in the full sharing range:

$$D^{fs} = (1 - \frac{p - \theta(c_n + c_s)}{\theta\bar{\rho}})M.$$

Full market coverage is obtained when $\rho_b^{fs} = 0$, which implies:

$$p_{fc} = \theta(c_n + c_s).$$

A lower price will also ensure that the market is covered.

We next consider the optimal pricing strategy of the firm. Given that either full or partial sharing may hold in the network, the firm faces two separate cases. Profits in the lower price range, under full sharing, are given by D^{fs}:

$$\pi_{lr} = p(1 - \frac{p - \theta(c_n + c_s)}{\theta\bar{\rho}})M, \tag{13}$$

which has a maximum at

$$p_{hc} = \frac{1}{2}\theta(\bar{\rho} + c_n + c_s).$$

We denote the maximum by p_{hc}, as high coverage of the market is obtained in this price range. In the higher price range, under partial sharing, profits are given by D^{ps}:

$$\pi_{hr} = p(1 - \frac{p}{\theta(\bar{\rho} + Mc_s)})M, \tag{14}$$

which has a maximum at

$$p_{lc} = \frac{1}{2}\theta(\bar{\rho} + Mc_s).$$

As market coverage is lower in this range, we denote the maximum by p_{lc}

To solve the firm's optimal price strategy, profits given by the optimal price in both ranges need to be compared under all feasible parameter configurations. Solving the systems of inequalities implied determines the profit-maximizing price as a function of all the parameters. ■

Improving Robustness of Peer-to-Peer Streaming with Incentives

Vinay Pai Alexander E. Mohr

Stony Brook University
{vinay, amohr}@cs.stonybrook.edu

Abstract

In this paper we argue that a robust incentive mechanism is important in a real-world peer-to-peer streaming system to ensure that nodes contribute as much upload bandwidth as they can. We show that simple tit-for-tat mechanisms which work well in file-sharing systems like BitTorrent do not perform well given the additional delay and bandwidth constraints imposed by live streaming. We present preliminary experimental results for an incentive mechanism based on the Iterated Prisoner's Dilemma problem that allows all nodes to download with low packet loss when there is sufficient capacity in the system, but when the system is resource-starved, nodes that contribute upload bandwidth receive better service than those that do not. Moreover, our algorithm does not require nodes to rely on any information other than direct observations of its neighbors' behavior towards it.

1 Introduction

In recent years, BitTorrent [4], a peer-to-peer file sharing protocol has become one of the most widely used tools for bulk data dissemination to large numbers of nodes. BitTorrent allows a large number of nodes to simultaneously download a large file by breaking it into chunks and having different nodes exchange chunks among each other.

While BitTorrent is effective for transferring files, the file is not downloaded in sequence and is therefore generally unusable till the download is complete. Moreover the bandwidth delivered often varies over time, making it unsuitable for applications like streaming video.

Many peer-to-peer streaming protocols [2, 6, 5, 10] have been proposed. However, in order to perform well, most of them assume a resource-rich environment where there is sufficient upload capacity in the system to support all downloaders.

However, several studies [1, 11] have shown that users of peer-to-peer networks tend to be selfish and try to benefit from a system without contributing resources in return. Moreover, several nodes are *unable* to contribute as much upload bandwidth as the download bandwidth they consume because they are using an asymmetric Internet connection like a consumer cable-modem or ADSL line.

This could result in a system where the demand for download bandwidth exceeds the supply of available upload bandwidth, making it impossible to satisfy all demand. In such a system we would like to give nodes an incentive to upload as much as they can by making the probability of suffering packet loss inversely proportional to the upload bandwidth contributed.

To achieve this, we tried to design an incentive scheme for peer-to-peer streaming on top of the Chainsaw [9] streaming protocol. We found that due to the strict time and bandwidth constraints and intolerance to long delays in streaming, incentive schemes designed for file-sharing protocols perform very poorly in streaming systems. In this paper, we start by presenting some of our *unsuccessful* attempts along with our reasoning of why they failed to perform well, in the hope of generating discussion.

In addition, we present an incentive scheme called Token Stealing based on an Iterated Prisoner's Dilemma that appears promising based on our preliminary experiments on PlanetLab [3].

2 Background

We implement our incentive schemes on top of Chainsaw [9], a streaming protocol based on an unstructured mesh network. Chainsaw uses a simple request-response protocol which is briefly described here.

Chainsaw Streaming Protocol

Chainsaw is designed to deliver a stream of data from one node (called the *seed*) to a large number of recipients. While Chainsaw may be generalized to multiple seeds, as well as many-to-many multicast, in this paper we only consider one-to-many multicast with a single seed. We refer to the set of nodes to which a peer is connected as its *neighbors*.

Every node maintains a list of packets that its neighbors are willing to provide. To ensure that this list is updated properly, whenever a node receives a new packet, it broadcasts a NOTIFY message to its neighbors to inform them of the change. The seed obviously does not receive packets, but does generate new packets periodically. The seed sends out NOTIFY messages to its neighbors every time it generates a new packet.

Every node also maintains a *window of interest*, which is the set of sequence numbers that the node is interested in acquiring at the current time. The node slides its window of interest forward over time as new packets stream in. If a packet has not been received by the time it "falls off" the trailing edge of the window, the node will consider that packet lost and will no longer try to acquire it.

For every neighbor, a node creates a list of *desired packets*, i.e. a list of packets that the node requires that the neighbor is able to provide. It will then pick one or more packets from the list at random and request them via a REQUEST message.

A node keeps track of which packets it has requested from which neighbor to ensures that it does not request the same packet from multiple neighbors. It also limits the number of outstanding requests with a given neighbor to ensure that requests are spread out over all neighbors. Finally, when a node receives a REQUEST message, it responds with a corresponding DATA packet as bandwidth permits.

Chainsaw has no global routing tables, so it does not depend on any specific network topology. In this paper, we will assume a topology in which every node repeatedly connects to a randomly picked node from the list of known hosts until a predefined minimum number of neighbors (the *node degree*) is reached. This network has the advantage of being very easy to construct and maintain even in the face of the sudden departure of a large fraction of nodes. Many practical peer-to-peer networks like BitTorrent and Gnutella use an unstructured random graph topology.

3 Poor Performance: Naïve Tit-for-Tat

Chainsaw's request-response protocol is similar to that used by BitTorrent, as well as SWIFT[12], a pairwise currency mechanism for file-sharing that we showed to be more effective at providing fairness in file-sharing than BitTorrent's incentive mechanism.

In SWIFT, every node maintains *credit* for each of its neighbors and honors packets requests only when the neighbor has enough credit. Whenever it receives a packet from a neighbor, the node extends it α packets worth of credit. In addition, trading is jump-started by initializing neighbors with γ packets worth of credit instead of zero, and deadlocks are avoided by periodically extending nodes a small fraction β of their total upload capacity in credit every second, regardless of data received from it.

As long as nodes consistently upload, they will keep earning credit with their neighbors and be able to download. However, nodes that do not upload will soon deplete their credit with their neighbors and not be able to download anymore, except for small trickle of free credit they receive from their neighbors in the form of β.

While SWIFT was very effective at ensuring fairness in file-transfer applications, we found a similar mechanism to perform very poorly when applied to streaming. In our simulations we found that over time, a large fraction of nodes started to suffer severe ($> 50\%$) packet loss even in a system where every node tried to upload as much as their capacity allowed. This was caused by small imbalances between nodes (eg. due to different delay characteristics, distance from seed, number of neighbors) being amplified by an undesired positive-feedback loop.

For instance, consider a pair of nodes A and B, where A is closer to the seed than B. In this situation, node A is likely to receive new packets before node B. As a result, node B has fewer opportunities to upload packets to node A, resulting in a net loss of credit. Eventually, node B runs out of credit and is no longer able to download from node A. However, given its proximity to the seed, it is likely that node A was a source of packets that was of interest to node B's other neighbors. Therefore, the loss of node A as a trading partner puts node B in a less favorable position to trade with the rest of its partners. This creates a positive feedback loop where a slight disadvantage is ultimately amplified to the point where a node is unable to earn enough credit to avoid packet loss.

4 Partial Success: Compensating for Trading Imbalances

We experimented with a number of mechanisms for compensating for these small imbalances.

4.1 Preferential Uploading

In the naïve tit-for-tat experiments, we found that nodes ran out of credit because they were unable to upload enough packets to some of their neighbors to maintain a stable supply of credit. Therefore, we implemented a system where nodes aggressively tried to upload to neighbors they were running out of credits with by giving requests from those neighbors a higher priority. Quickly satisfying existing requests results in more requests for packets from that neighbor because nodes limit the number of requests outstanding with a given neighbor at any time.

Unexpectedly, this strategy made the problem *worse*. Some nodes were at an advantage with respect to most of their neighbors, which led to an "arms race" among neighbors to upload as quickly as possible to the advantaged node. This created a new positive feedback loop where advantaged nodes were put at an increasingly greater advantage by neighbors aggressively uploading to them. Eventually, the neighbors that lost the arms race ran out of credits as they did in the naïve tit-for-tat system.

4.2 Advantaged Nodes Back Off

Our next approach was to have advantaged nodes attempt to *reduce* the number of packets they uploaded rather than increase the number of packets their disadvantaged neighbors uploaded to them. We did this by having nodes keep track of every neighbor's balance ratio,

$$\text{balance-ratio} = \frac{\text{total-download}}{\text{total-upload} + \text{total-download}}$$

Note that this calculation is done purely based on direct local observations of a neighbor's behavior towards the node.

Nodes with balance-ratio < 0.5 have been uploading more than they have been downloading. By default, nodes send NOTIFY messages to all their neighbors when they receive a new packet to enable them to download it. However, as the balance-ratio fell below 0.5, we linearly reduced the number of neighbors notified. This ensured that the advantaged nodes only uploaded a small number of copies of every packet.

This is beneficial to the advantaged node, disadvantaged node, and the system as a whole. The advantaged nodes benefit by having some of the burden of uploading taken off them, while the disadvantaged nodes benefit by having a greater opportunity to upload packets to their neighbors and earn credit. The overall amount of upload bandwidth in the system is generally not reduced because some of the burden of uploading packets is shifted from the advantaged to the disadvantaged nodes. The advantaged node is then able to use its upload bandwidth to rapidly propagate new packets rather than multiple copies of old packets.

This scheme worked very well in our simulations, and we were able to maintain a balance-ratio between 0.45 and 0.55 across all pairs of neighbors in the network. Unfortunately, the algorithm failed to produce a significant benefit in real-world tests on real a implementation of the protocol on the PlanetLab testbed. We found that the variation in bandwidth capacity and round-trip delays between different pairs of nodes in the network so great that it was not possible to accommodate the slowest nodes without dragging down the performance of the entire system.

5 Promising: Token Stealing Algorithm

Our next attempt was an algorithm we call *Token Stealing*, which builds on the standard *token bucket* model commonly used to allocate limited bandwidth among competing processes. The Token Stealing algorithm sets up local markets at every node where neighbors compete for the node's upload capacity. When the demand for bandwidth from the node exceeds the node's capacity, nodes that upload receive preferred service, while this constraint is relaxed when there is adequate bandwidth to fulfill all requests.

We first outline the standard token bucket algorithm.

The token bucket algorithm works by having a virtual bucket into which tokens are added periodically. Whenever a packet is transmitted, an equivalent number of tokens must be removed from the bucket—packets may only be transmitted when there are a sufficient number of tokens available in the bucket. Thus, the overall bandwidth can be controlled by controlling the rate at which tokens are added to the bucket.

The number of tokens that may accumulate in the bucket is limited to some maximum value to prevent a large number of tokens from accumulating during periods when there is low demand for bandwidth.

The basic token bucket algorithm only ensures that the overall bandwidth doesn't exceed a specified limit. We augment this with the Token Stealing algorithm to give a higher priority to nodes that have been consistently uploading than those that haven not.

The **Token Stealing algorithm** is a simple extension of the token bucket algorithm. In this algorithm, every node maintains a standard token bucket that we refer to as the *shared bucket* into which tokens are added periodically. In addition, the node maintains a separate bucket for each of its neighbors. We refer to these as *private buckets*. Whenever a node receives a packet from one of its neighbors, it removes tokens from the shared bucket and transfers them to that neighbor's private bucket. This has the effect of reserving a portion of the node's upload bandwidth to repay the neighbor for the packets it has uploaded.

To prevent neighbors from reserving large amounts of bandwidth that they never utilize (for example, because they are connected to other nodes with large upload capacities), there is a limit on the size of the private buckets. Tokens that overflow the private buckets are returned to the shared bucket.

Which Bucket First?

The question of which bucket to deduct tokens from when a neighbor requests a packet is interesting. One may choose to first deduct tokens from the private bucket and dip into the shared bucket only if there are not enough tokens in the private bucket, or one may use up tokens from the shared bucket first.

In our experiments we found that the both strategies give the neighbors that upload (and therefore have tokens in their private buckets) an advantage, but that advantage is considerably greater in the latter case. When tokens are deducted from the private buckets first, neighbors that upload do not compete in the market for the shared tokens unless their private buckets are empty. This makes it easier for neighbors that do not upload to receive a portion of the bandwidth.

When tokens are deducted from the shared bucket first, all neighbors compete equally in the market for shared tokens before dipping into their private buckets, which act as a "reserve". This amplifies the priority given to the nodes that upload.

Therefore, the strategy we choose is to deduct tokens from the shared bucket first and only dip into the private bucket when the shared bucket is empty.

5.1 Analysis

With the Token Stealing algorithm, the total upload capacity of the node is still limited by the rate at which tokens are added to the token bucket, i.e. the upload bandwidth limit. However, unlike a simple token bucket system where all nodes have an equal opportunity to use up tokens from the bucket, the Token Stealing algorithm favors neighbors that upload.

Whenever a neighbor uploads a packet to a node, the node reserves tokens for that neighbor's use. Every packet the neighbor uploads serves to a node increases the chances that the neighbor will be able to download a packet in the future.

If all neighbors upload equally, all private buckets will have the same number of tokens in them, which gives all neighbors equal priority. However, a neighbor that does not upload will not have tokens in its private bucket and will be limited to competing with other neighbors for tokens from the shared bucket.

Whether or not the non-uploading neighbor succeeds in downloading depends on the total supply and demand at that node:

5.1.1 Node has excess upload capacity

If the node has more than enough upload capacity to fulfill the demand of all of its neighbors, the shared bucket will have tokens in it and the neighbor that does not upload will still be able to download. This ensures that a node's upload capacity is utilized as much as possible.

It is possible for a few nodes, known as *free-riders* to try to leach off the system by selectively connecting to nodes with excess capacity. This strategy will work so long as the number of free-riders is small. If a large number of nodes attempt to leach off the system, they will compete among each other for tokens from the shared token bucket. This makes the effect of free-riders self-limiting.

5.1.2 Node has limited upload capacity

If the node does not have enough capacity to satisfy all requests, most of the tokens will be moved to the private buckets of the neighbors that do upload, and the shared bucket will generally be empty. As a result, the neighbors that upload will be able to use the tokens from their private buckets to download packets, but nodes that do not upload will be forced to compete for the scarce tokens from the shared bucket.

5.2 Prisoner's Dilemma

The Token Stealing Algorithm may be modeled as an Iterated Prisoner's Dilemma problem. If all of a node's neighbors defect (refuse to upload), they all share the common pool and none of the neighbors has an advantage. However, a neighbor that chooses to upload (cooperate) can "steal" tokens away from the shared bucket. The neighbor will still compete equally for the remaining tokens in the shared bucket, but will have a private reserve for itself in addition to the tokens it receives from the shared bucket. In this case, the best strategy for the other neighbors to upload in order to move tokens to their own private buckets. Thus, whenever the upload capacity at a node is scarce, the dominant strategy for every neighbor is to upload to that node, i.e. to cooperate.

6 Experimental Evaluation

To evaluate the performance of the Token Stealing algorithm, we build an application and deployed it on 350 nodes across the globe on the PlanetLab[3] testbed. All nodes joined the network before the seed (source node) started broadcasting data, and connected to an average of 15 neighbors each. The stream was divided into 4 kilobyte packets at a rate of 25 packet/second to give a total stream rate of a 100 kilobytes/sec.

We did not constrain the download capacity of nodes in any way, but capped the upload capacity of nodes to put them in one of two classes:

1. *Fast Nodes*: Maximum upload capacity = 200 kilobytes/sec

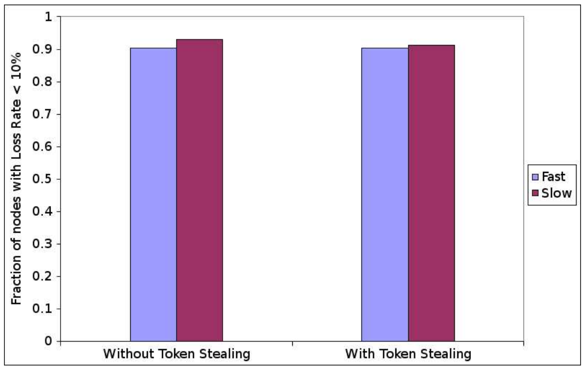


Figure 1: In a resource-rich system both fast and slow nodes are able to download with very little packet loss.

2. *Slow Nodes*: Maximum upload capacity = 25 kilobytes/sec

6.1 Resource-Rich System

In our first experiment, the system had 75% of fast nodes and 25% of slow nodes. This made the average supply of upload capacity 156.25 kB/node. The demand from every node regardless of their upload capacity was the full 100 kB/sec stream rate. Thus, the system had approximately one and a half time the supply as demand.

We ran this system for 300 seconds and measured the packet loss rate experienced by different nodes. Note that machines on PlanetLab are shared between many researchers and are often very heavily loaded, resulting in severe and unpredictable constraints on available bandwidth and CPU time. This causes some nodes to suffer severe packet loss even when there are adequate resources in the system.

In this system, over 90% of the fast nodes as well as slow nodes suffered less than 10% packet loss regardless of whether or not the Token Stealing algorithm was used. This shows that the Token Stealing algorithm does not harm the performance of a resource-rich system.

6.2 Resource-Starved System

In our second experiment, the system had 25% of fast nodes and 75% of slow nodes. This made the average supply of upload capacity 68.75 kB/node. Thus, the system had approximately two third the supply as demand.

In this system, there is a major benefit to having the Token Stealing algorithm enabled. Without token stealing, barely 3% of fast and 6% of slow nodes had less than 10% packet loss. With Token Stealing enabled, things improved dramatically for the fast nodes—68% of them had less than 10% packet loss. Clearly, in a resource-starved system with Token Stealing enabled, nodes have a big incentive to upload as much as they can.

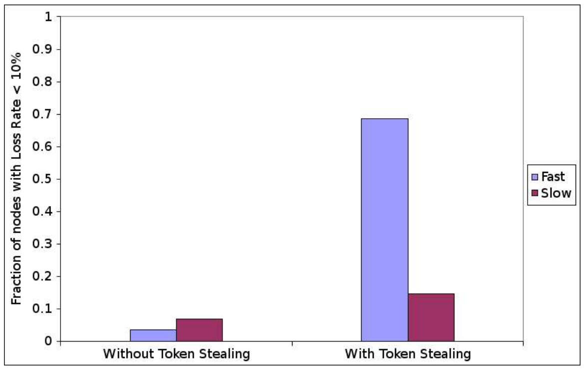


Figure 2: In a resource-starved system without Token Stealing, both fast and slow nodes suffered heavy packet loss. Barely 3% of fast nodes and 6% of slow nodes were able to download with less than 10% packet loss. With Token Stealing enabled, however, 68% of the fast nodes and 14% of the slow nodes suffered a packet loss rate under 10%.

7 Discussion

Our investigations have shown that despite the many similarities between BitTorrent-like file-sharing systems and mesh-based peer-to-peer streaming systems, incentive schemes used in file-sharing can not be easily applied to streaming.

Simple tit-for-tat schemes do not work well because of the additional constraints imposed by live streaming. For example, in a file-sharing network, every packet is useful until all nodes in the system have downloaded that packet. In streaming, however, packets quickly become obsolete.

Our preliminary work with the Token Stealing algorithm has shown promising results. We find that it allows nodes that are unable to contribute much upload bandwidth to still download the stream with low packet loss so long as the supply of bandwidth in the system exceeds the demand. This allows the system to take advantage of altruistic nodes that contribute more upload bandwidth than the stream rate, and to avoid imposing harsh penalties on nodes that are unable to upload (for example ADSL nodes).

However, when the system is resource-constrained because there aren't enough altruistic nodes to close the gap between the supply of and demand for bandwidth, it is impossible for all nodes to download the stream with no packet loss. Under these circumstances, nodes that contribute upload bandwidth to the system are given a higher priority and tend to suffer much lower packet loss.

So far we have only investigated this algorithm under very limited circumstances. While our initial results are promising, we still need to investigate the effect of many real-world conditions, such as the fact that node bandwidths do not fall into a small number of well-defined categories, and that available bandwidth varies over time. Moreover, we have yet to investigate the ways in which nodes may game the system.

We believe that every practically deployed peer-to-peer streaming system needs to give nodes an incentive to upload as much as they can in order to ensure that the system remains resource-rich and operates well. However, it is better to avoid shutting out nodes that are unable to upload as fast as they download unless there are insufficient altruistic nodes in the system to make up the deficit.

8 Related Work

Traditional multicast approaches have relied on building spanning trees over the network and pushing data over those trees in order to minimize delay. This creates parent-child relationships that make it hard to identify and penalize nodes that do not upload based purely on local observations.

However, Ngan, Wallach, and Druschel propose a general reputation-based system [8] to detect and penalize free-riders. Their solution can be applied to any tree-based multicast system. However, their solution require the multicast trees to be rebuilt continuously. We believe our system to be easier to implement in a decentralized manner in practice.

Levin, Sherwood and Bhattacharjee describe an interesting overlay structure[7] for file swarming that is a radical departure from BitTorrent and other tit-for-tat approaches. However, in the current form it has two severe limitations. Firstly they assume that every node in the system has exactly the same upload capacity. Secondly, the disincentive to defect comes from the *collapse of the entire system* when a single node defects. As the authors acknowledge, these drawbacks make the system impractical in its current form, but these problems may be alleviated with further research.

9 Conclusion

We argue that in order for a peer-to-peer streaming system to be robust, it is important to have an effective mechanism to give nodes an incentive to upload as much as they can. Our investigations show that naïve application of tit-for-tat mechanisms that work well in file-sharing systems do not perform satisfactorily in streaming systems due to additional bandwidth and delay constraints. We have outlined an algorithm called "Token Stealing" that runs locally on every node and relies only on direct observations without the need for network-wide or third-party coordination. Our experiments show that this algorithm helps reduce the packet loss for nodes that contribute upload bandwidth in a resource-constrained system, while not shutting nodes with poor upload capacities out of resource-rich systems.

References

[1] E. Adar and B. A. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), Oct 2000.

[2] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-Bandwidth Multicast in Cooperative Environments. In *SOSP*, 2003.

[3] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. *SIGCOMM Computer Communication Review*, 2003.

[4] B. Cohen. BitTorrent, 2001. http://www.bitconjurer.org/BitTorrent/.

[5] J. Jannotti, D. K. Gifford, K. L. Johnson, M. Frans Kaashoek, and J. O'Toole, Jr. Overcast: Reliable multicasting with an overlay network. In *OSDI*, 2000.

[6] D. Kostić, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: high bandwidth data dissemination using an overlay mesh. In *SOSP*, 2003.

[7] Dave Levin, Rob Sherwood, and Bobby Bhattacharjee. Fair file swarming with fox. In *Fifth International Workshop on Peer-to-Peer Systems*, 2005.

[8] T. Ngan, D. S. Wallach, and P. Druschel. Incentives-compatible Peer-to-Peer Multicast. In *Second Workshop on the Economics of Peer-to-Peer Systems*, 2004.

[9] Vinay Pai, Kapil Kumar, Karthik Tamilmani, Vinay Sambamurthy, and Alexander E. Mohr. Chainsaw: Eliminating trees from overlay multicast. In *Fourth International Workshop on Peer-to-Peer Systems*, 2004.

[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *ACM SIGCOMM*, 2001.

[11] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. *Proceedings of Multimedia Computing and Networking*, 2002.

[12] K. Tamilmani, V. Pai, and A. E. Mohr. SWIFT: A system with incentives for trading. In *Second Workshop on the Economics of Peer-to-Peer Systems*, 2004.

Dandelion: Cooperative Content Distribution with Robust Incentives

Michael Sirivianos Xiaowei Yang Stanislaw Jarecski
Department of Computer Science
University of California, Irvine
{msirivia,xwy,stasio}@ics.uci.edu

Abstract

Online content distribution has increasingly gained popularity among the entertainment industry and the consumers alike. A key challenge in online content distribution is a cost-efficient solution to handle demand peaks. To address this challenge, we propose Dandelion, a system for robust cooperative (peer-to-peer) content distribution. Dandelion explicitly addresses two crucial issues in cooperative content distribution. First, it provides robust incentives for clients who possess content to serve others. A client that honestly serves other clients is rewarded with credit that can be redeemed for future downloads at the content server. Second, Dandelion discourages unauthorized content distribution. A client that uploads to another client is rewarded for its service only after the server has verified the other client's legitimacy. Our preliminary evaluation of a prototype system running on commodity hardware with 120 KB/sec upload rate indicates that Dandelion can achieve aggregate client download throughput three orders of magnitude higher than the one achieved by an HTTP/FTP-like server.

1 Introduction

Content distribution via the Internet is becoming increasingly popular among the industry and the consumers alike. A survey showed that Apple's iTunes music store sold more music than Tower Records and Borders in the US in the summer of 2005 [18]. A number of key content producers, (e.g. CBS, Disney, Universal), are now selling films online [2, 3, 6].

A challenging issue for online content distribution is a cost-effective solution to handle peak usage by promotions or new releases. A 45-minute DVD-quality episode easily exceeds one GB. Even if each user is provisioned with a 1 Mbps, it takes more than two hours to download 1 GB. Overprovisioning for one additional user during peak usage may require at least an additional 1Mbps bandwidth, which often costs up to $100 per month [5, 13]. However, a TV episode is commonly sold at less than two dollars. One solution is to purchase service from a content distribution network (CDN) such as Akamai. However, CDNs' services are also costly, and free CDNs such as Coral, CoDeen, and CobWeb [11, 20, 26, 31] lack a viable economic model to scale.

This work explores a cost-effective approach for handling flashcrowds. We present the design and a preliminary evaluation of Dandelion, a cooperative content distribution system. Rather than using a third party service, a Dandelion server utilizes its clients' bandwidth. During a flash crowd event, a server redirects a request from a client to the clients that have already downloaded the same content. This approach is similar in spirit to previous work on cooperative content distribution [12, 16, 24, 27, 28], most notably BitTorrent [8]. However, with the exception of BitTorrent, the above approaches do not provide incentives for a client to upload to other clients. BitTorrent employs tit-for-tat incentives, but these are susceptible to manipulation [15, 17, 25] and does not motivate clients to upload content after the completion of their download (i.e., seeding).

The primary contribution of our work is that we provide robust incentives for clients to upload to others. By robust, we mean that the incentive mechanism does not rely on clients being altruistic or honest. Its secondary contribution is that Dandelion discourages unauthorized content sharing. Our design gives no incentives to clients to upload to unauthorized clients, but provides explicit rewards for them to upload to authorized clients, e.g., clients that have purchased content at a server.

Dandelion's incentive mechanism is based on a cryptographic fair exchange mechanism, which uses only efficient symmetric cryptography. A client uploads content to other clients in exchange of virtual credit. The credit can be redeemed for future service by other clients, or for service by the server itself, or other rewards. This incentive mechanism discourages unauthorized content exchange, because a client is rewarded for its service only after the server has verified that the client has uploaded to an authorized client.

We have implemented a prototype of a Dandelion client and server and conducted a preliminary evaluation on PlanetLab [7]. We compare the throughput of a Dandelion server with a server that runs a simple request-response protocol, such as HTTP. Our preliminary evaluation shows that Dandelion can improve the throughput of a commodity PC server with 1 Mbps bandwidth by three orders of magnitude. However, as a trade-off of providing robust incentives and discouraging unauthorized content distribution, a Dandelion server is less efficient than a BitTorrent tracker. As a result, a Dandelion system is less scalable than BitTorrent, with respect to the number of active clients supported by a single server/tracker.

The rest of this paper is organized as follows. Section 2 describes the design of Dandelion. Section 3 briefly discusses our implementation and its performance. Section 4 compares our work with related work. We conclude in Section 5. In the Appendix we provide a detailed description of our protocol and discuss its security.

2 Design

This section describes the design of a Dandelion server and client at a high-level. Please see the Appendix for the detailed protocol description.

2.1 Overview

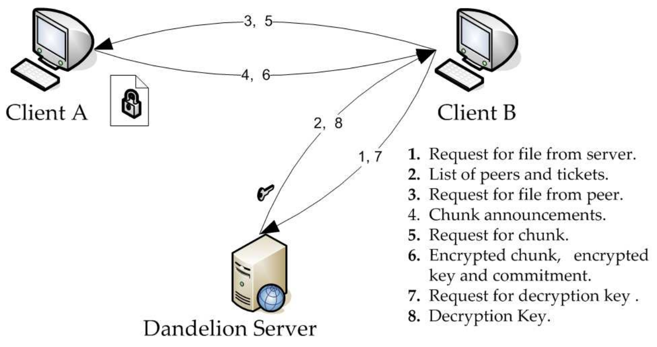
A Dandelion server is optimized to provide large static files. It behaves similar to a web/ftp server under normal work load, responding to clients' requests with content. When a Dandelion server is overloaded, it enters a *peer-serving* mode. Upon receiving a request, the server redirects the client to clients that are able to serve the request.

A Dandelion server maintains a virtual economy. It rewards cooperative clients that upload to others with virtual credit to provide robust incentives. The credit is used as "virtual money" to purchase future downloads from other clients or from the server itself (at a high credit cost when the server is overloaded), or used as other types of rewards.

Similar to BitTorrent, a Dandelion server splits a large file into multiple chunks, and disseminates them independently. This allows clients to participate in uploading chunks as soon as they receive a small portion of the file, increasing the efficiency of the distribution pipeline. Furthermore, this incentivizes clients to upload chunks to others, as they need credit to acquire the missing ones.

2.2 Robust incentives

A key challenge in designing a credit system is to prevent client cheating, while keeping both a server and a client's processing and bandwidth costs low. A dishonest client that does not upload to others or uploads garbage may attempt to claim credit at the server, and to be robust, the server must not award credit to such cheating behavior. To address this challenge, Dandelion employs a cryptographic fair exchange mechanism. A Dandelion server serves as the



Client A
Client B
Dandelion Server
3, 5
4, 6
2, 8
1, 7
1. Request for file from server.
2. List of peers and tickets.
3. Request for file from peer.
4. Chunk announcements.
5. Request for chunk.
6. Encrypted chunk, encrypted key and commitment.
7. Request for decryption key .
8. Decryption Key.

Figure 1: The peer-serving protocol. The numbers on the arrows correspond to the listed protocol messages. The messages are sent in the order they are numbered.

trusted third party mediating the exchanges of content for credit among its clients. When a client *A* uploads to a client *B*, it sends encrypted content to client *B*. To decrypt, *B* must request keys from the server. The requests for keys serve as the "proof" that *A* has uploaded some content to *B*. Thus, when receiving a key request, the server credits *A* for uploading to *B*, and charges *B* for the content.

A problem occurs if a malicious client *A* sends invalid content to *B*. *B* can discover that the content is invalid only after receiving the decryption key and being charged. To address this problem, our design includes a non-repudiable complaint mechanism. If *A* intentionally sents garbage to *B*, *A* cannot deny it. In addition, *B* is prevented from falsely claiming that *A* has sent it garbage. For clarity, we describe the complaint mechanism after we describe the normal message exchange in a Dandelion system.

Figure 1 shows how messages are exchanged in a Dandelion system. We assume that each client has a password-protected account with the server and that it establishes a secure channel (e.g SSL), over which it obtains shared session keys with the server. During a flash crowd event, the Dandelion server keeps track of the clients that are currently downloading or seeding offered files. The message exchange proceeds as follows:

Step 1: A client (*B* in Figure 1) sends a request for a file to the server.

Step 2: When the server receives the request, it returns digests of the file chunks for integrity checking [8], a random list of other clients that can serve the file, and cryptographic authorizations, namely tickets that enable *B* to request chunks from these clients.

Step 3: Upon receiving the server's response, *B* connects to the listed clients to request the file. We use client *A* as an example in Figure 1.

Step 4: If *B*'s tickets verify that the server has authorized *B* to request chunks from *A*, *B* and *A* will run a chunk selection protocol similar to that in BitTorrent [8]. *A* reports periodically to *B* what chunks it has. *B* determines which chunks it wishes to download and from which peers according to a chunk scheduling algorithm such as *rarest first*.

Step 5: *B* sends a request for the chunk to *A*.

Step 6: If *B*'s ticket verifies, *A* chooses a random key *k*, and encrypts it with the session key K_{SA}, it shares with the server. Client *A* sends to *B* the chunk encrypted with *k*, the encryption of the key *k*, and its cryptographic commitment to the encrypted chunk. *A* generates the commitment by computing a message authentication code (MAC), keyed with the shared session key K_{SA}, over the digest of the encrypted chunk and the encryption of *k*

Step 7: To retrieve *k*, *B* sends a decryption key request to the server. The request contains the encryption of the key *k*, a digest of the encrypted chunk, and *A*'s commitment.

Step 8: Upon receiving *B*'s request, the server checks whether *A*'s commitment matches the one computed over *B*'s digest of the encrypted chunk and the encryption of key *k*, using K_{SA}. If the commitment verifies and *B* has sufficient credit, the server sends the key to *B*. At the same time, it rewards *A* with credit and charges *B*.

If *A*'s commitment does not verify, the server cannot determine whether the discrepancy is caused by a transmission error, or client *A* or *B* is misbehaving. The server simply warns *B* of the discrepancy, and does not return the encryption key *k*. It updates neither *A*'s or *B*'s credit. *B* can re-request the chunk from *A* or try another client.

If *B* repeatedly receives invalid commitments from *A*, it should disconnect from *A* and blacklist it. Similarly, if the server repeatedly receives decryption requests from *B* with invalid commitments from a specific *A*, the server knows that *B* is misbehaving because *B* should have blacklisted *A*. The server will blacklist *B*.

Next, we explain the complaint mechanism. After *B* receives the key *k*, it decrypts the chunk and validates its integrity. If the chunk is invalid, *B* can complain to the server, and *A* cannot repudiate it. This is because *B*'s complaint message contains *A*'s commitment, the digest of the encrypted chunk, and the encryption of key *k*, all received in the message from *A* in Step 6. The server can easily validate whether *A* has sent the commitment, as the commitment is a MAC computed with the session key K_{SA} shared between the server and *A*. *B* cannot forge a valid commitment. If the commitment fails, the server knows that *B* is misbehaving, since it should have abandoned the transaction in step 8. If the commitment verifies, *A* cannot repudiate that it has sent the commitment to *B*. All the server needs to check is whether *A* has computed the commitment over a valid chunk. To verify this, the server retrieves and encrypts the chunk that *B* complains about, using the key *k* and computes the MAC using the shared key K_{SA}. If this recomputed commitment matches *A*'s commitment, it proves that *A* has sent the valid content, and *B* is framing *A*; otherwise, it proves that *A* has sent invalid content to *B*.

2.3 Credit Management

Dandelion can be used to distribute both free and paid content. Each usage dictates a distinct credit management policy. In the free-content case, we require that a client must have sufficient credit units to download either from a server or from a peer client. This is to incentivize a client to accumulate credit units by uploading to others. Each client is given an initial small amount of credit when it first registers at the server. This initial credit enables a new user to download a few chunks when it joins the Dandelion swarm. Clients spend $\Delta_c > 0$ credit units for each chunk they download from a peer client and earn $\Delta_r > 0$ credit units for each chunk they upload to a peer. Consequently, clients are forced to upload chunks proportionally to the number of chunks they want to download. In our system, a client can obtain a chunk only if its credit is greater or equal to the chunk's cost. To prevent collusions we set $\Delta_c = \Delta_r$, so that two colluders cannot increase the sum of their credit.

In the paid content case, if a client has already spent real money at the server to purchase content, a client may not be charged credit units to download file chunks. That is, a server rewards a client that uploads a chunk with $\Delta_r > 0$ credit units, but may charge $\Delta_r = 0$ for a client that downloads a chunk. To incentivize a client to accumulate credit units, a server may redeem a client's credit for monetary awards, such as discounts on content prices or service membership fees, similar to the mileage programs of airline companies. How to convert virtual credit to rewards depends on the economic goals of each Dandelion deployment. Note that collusions among clients are still not effective. If a client wishes to boost another client's credit, it would need to request decryption keys for certain chunks. Since a paying client aims at downloading the complete content, a colluder would have to send multiple decryption key requests for a certain chunk. The server can detect and punish this behavior, deterring collusion among rational clients.

2.4 Discouraging unauthorized content distribution

In Dandelion, rational authorized clients are discouraged from serving content to unauthorized clients. This is because a server does not award them credit for illegitimate transactions. Clients are able to verify the legitimacy of requests for service (as described in Section 2.2), hence they can avoid wasting bandwidth to send encrypted chunks to unauthorized peers. Furthermore, due to this ability, clients can be held liable if they choose to send plaintext contents to unauthorized clients. These properties discourage users from using Dandelion for illegal content replication and make our solution appealing to distributors of copyright-protected digital goods.

Dandelion Server

Dandelion Operation	Size	Time (ms)
Decrypt decryption key	40 bytes	0.1
MAC decryption key request	138 bytes	0.02
Transmit decryption key response	92 bytes	$\sim$ 0.84
Query and update credit base (SQLite)	N/A	1.08
Transmit chunk	256 KB	$\sim$ 2330

Dandelion Client

Dandelion Operation	Size	Time (ms)
Encrypt/decrypt chunk	256 KB	4.1
Encrypt/decrypt chunk	16 KB	0.35
Commit to chunk (Hash and MAC)	256 KB	1.45

Table 1: Timings of per-chunk Dandelion operations.

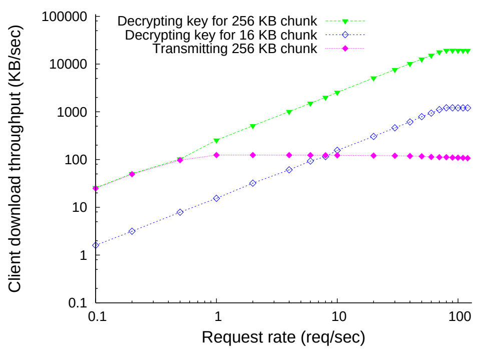


Figure 2: The *y* axis shows the achievable aggregate content download throughput of Dandelion clients when the server responds to: a) requests for keys used to decrypt 256 KB encrypted chunks; b) requests for keys used to decrypt 16 KB encrypted chunks; and c) requests for chunks. The *x* axis is the request rate received by the Dandelion server.

3 Preliminary Evaluation

We implemented a prototype of Dandelion in C under Linux, and conducted a preliminary evaluation of its performance on PlanetLab. This section describes our implementation and the results of our PlanetLab experiments.

3.1 Prototype Implementation

We implemented Dandelion's cryptographic operations using the *openssl* C library and the credit management system using the lightweight database engine in the *sqlite* library.

Our server implementation draws from the Flash [19] web server's Asymmetric Single Process Event Driven Architecture and the Staged Event Driven Architecture [10]. Both architectures assign thread pools to specific tasks.

When a disk read or a database operation is required by a request, Dandelion's main thread dispatches the request to a synchronized producer-consumer queue served by a pool of helper *disk access* or *database access* threads, respectively.

When a helper thread finishes its operations, it dispatches the request to another thread pool (next stage) for subsequent processing. To avoid multiple copies between kernel and user space during reading and sending a file chunk, we use the *sendfile()* system call. Owing to *sendfile()*, both reading a chunk from file and writing it to the network are executed by the same thread.

This design exploits parallelism and maintains good performance when both small and cached files or large disk-residing files are requested from the server itself. In addition, it does not bind the number of concurrent connections or pending requests to the number of processes/threads that the OS can efficiently accomodate simultaneously.

3.2 Experimental Results

We first evaluate the computational costs of a Dandelion server. In a flash crowd event, the main task of a Dandelion server is to process key decryption requests and send short responses to those requests. To process one decryption request, a server performs one HMAC operation and one block cipher decryption on small messages. Furthermore, it performs one query and two update operations on a credit database. Lastly, it transmits the decryption key.

In our experiments, we deployed a Dandelion system with one server and 100 clients. The server runs on Linux 2.6.14 on a 1.7 GHZ/2 MB Pentium M CPU and one GB RAM. To stress our design and emulate a typical resource-limited server with 1 Mbps access link, we rate limited our server's upload rate to 120 KB/s. We let the Dandelion clients send the following two types of requests to the server and benchmarked the client download throughput along with the processing costs. The first type was requests for decryption key, which emulate the load on the server during peer-serving. The second type was requests for file chunks directly from the server. Each client sent requests at a rate ranging from 0.001 to 1 requests/sec. As the client chunk request rate increased, the client would send a new request prior to downloading the previously requested chunk. For each rate, the experiment duration was 10 minutes and the results were averaged over 10 experiments.

Table 1 shows the cost for each operation. As can be seen, the cryptographic operations of Dandelion are highly efficient, as only symmetric cryptography is involved. A Dandelion client can encrypt and decrypt a 256 KB chunk much faster than download it or transmit it at 1 Mbps. This result suggests that the client's processing overhead does not affect its upload or download throughput.

Figure 2 compares the case in which Dandelion clients send decryption key requests to a server, as if they peer-serve each other, with the case in which clients request the file directly from the server, i.e., the HTTP/FTP-like downloading. The curves show that with a chunk size of

256 KB, the Dandelion clients' download throughput would be almost 200 times higher than the throughput obtained when the clients request the file directly from the server. A smaller chunk size reduces the performance gain, as a server must process more decryption key requests. We note that the performance of Dandelion can be further improved. Figure 2 was obtained using a version of the Dandelion server that was optimized for directly serving file chunks, but was poorly tuned for processing decryption key requests. In our latest version of the implementation we configured the system to commit the SQLite credit database to the disk much less frequently. This resulted to the credit query and update operation requiring 1 ms instead of 10 ms (Table 1). Our latest preliminary results indicate a five fold increase in the throughput of decryption key response. This means that a commodity server with a 1 Mbps access link could process up to $\sim$500 decryption key requests per second, effectively serving up to $\sim$2000 clients downloading 256 KB chunks at 64 KB/s from other clients.

The cost of a complaint is higher because it involves reading a chunk, encrypting it with the sender client's key and hashing the encrypted chunk. However, the server blacklists misbehavers, thus it does not repeatedly incur the cost of complaints sent by them.

4 Related Work

This section briefly compares our work with related work.

Swarm file downloading protocols. Dandelion is inspired by swarm downloading protocols such as Bittorrent [8] and Slurpie [24]. A key difference of our work is its robust incentive mechanism. Slurpie does not provide incentives for peer-serving. Although Bittorrent employs rate-based "tit-for-tat" incentives, these do not punish free riders [15] due to the specifics of its unchoking mechanism. In addition, a free rider can enhance its advantage by obtaining a larger than normal initial partial view of the BitTorrent network. In this way, a peer can discover many seeders and choose to connect to them only [17], increasing his download rate. It can also increase the frequency with which it gets optimistically unchoked by connecting to all leechers in its large view [25].

Furthermore, as there is no robust mechanism to motivate seeding in BitTorrent, the number of clients that seed for long periods of times is very small [21]. In contrast, credit in a Dandelion system provides robust incentives for clients to seed files, which we believe will improve file availability when the swarm size is small.

Lastly, Dandelion has the desirable feature that rational clients have no incentives to serve unauthorized peers, as in such case the server will not reward them. In BitTorrent, content access policies are enforced by requiring password-based authentication with the tracker. However, an unauthorized peer can join the network simply by finding a single colluding peer that is willing to share its swarm view with it. The unauthorized peers can then download content from authorized peers, which have the incentives to serve them as long as the unauthorized peer is tit-for-tat compliant. As a result, a single authorized but misbehaving peer can facilitate illegal content replication at a large scale. In an upcoming BitTorrent version, access policies are implemented by accelerating legitimate content transfers through the use of strategically placed caches, which can be accessed only by authorized clients [1, 4]. Our scheme does not require third party infrastructure.

Escrow services in peer to peer networks. Horne et al. [14] proposed an encryption-based fair-exchange scheme for peer-to-peer file exchanges. Dandelion shares similarities regarding motivation and the general approach with their work, but differs in specific protocol design. Their scheme divides and transmits a file in chunks to enable erasure-code-based techniques for detecting cheaters that upload invalid content, whereas we divide files to support efficient and incentivized peer-to-peer distribution. Their scheme detects cheating with probabilistic guarantees, whereas Dandelion deterministically detects and punishes cheaters. In addition, their scheme requires that all chunks for a given file come from a single peer, which renders the distribution pipeline inefficient.

Fair-exchange schemes. Among the proposed solutions for the classic cryptographic fair-exchange problem, our scheme bears the most similarity with the one by Zhou et al. [32]. Their scheme also encrypts the content to be exchanged and uses an online trusted third party (TTP) to relay the decryption key. A key difference is that Zhou et al.'s scheme uses public key cryptography for encryption and for commiting to messages, and both of the exchange parties need to communicate with the TTP for each transaction. In contrast, our scheme uses efficient symmetric key encryption, and only one client needs to communicate with the TTP per transaction. The technique they use to determine whether a message originates from a party is similar to the one used by our complaint mechanism, but our work also addresses the specifics of determining the validity of the message.

Pairwise credit-based incentives. Swift [29] introduces a pairwise credit-based trading mechanism (barter) for peer-to-peer file sharing networks and examines the available peer strategies. Scrivener [9] is also an architecture in which peers maintain pairwise credit balances to regulate content exchanges among each other. In contrast, a Dandelion server maintains a central credit bank for all clients.

Global credit-based incentives. Similar to Dandelion, Karma [30] employs a global credit bank, with which clients maintain accounts. It distributes the credit auditor set of a peer among the peer's *k* closest neighbors in a DHT overlay [22]. Karma uses certified-mail-based [23] fair exchange of content for reception proofs, which requires both peers to communicate with the mediating auditor set for each exchange. Unlike Dandelion, Karma requires public key cryptographic operations at the peer side. Karma provides probabilistic guarantees with respect to the integrity of the credit-base. In the presence of numerous malicious bank nodes or in a highly dynamic network, the credit-base becomes difficult to maintain reliably.

5 Conclusion and Future Work

This paper describes a cooperative content distribution system: Dandelion. Dandelion's primary function is to offload a server during a flash crowd event, effectively increasing availability without overprovisioning. A server delegates a client with available resources to serve other clients. We use a cryptographic fair-exchange technique to provide robust incentives for client cooperation. The server rewards a client that honestly serves other clients with credit. A client can redeem its credit for further service or monetary rewards. In addition, the design of Dandelion discourages unauthorized content exchange. Since a server mediates all fair exchanges, clients who serve unauthorized requests are not rewarded, therefore it is in their best interests not to waste their upload bandwidth to serve unauthorized clients. A preliminary evaluation shows that Dandelion has low processing and bandwidth costs on the server side. A commodity server with 1 Mbps access link may support up to two thousand simultaneous clients. We are in the process of evaluating and fine-tuning Dandelion's prototype implementation on PlanetLab.

6 Acknowledgements

We thank Nikitas Liogkas and anonymous reviewers for their useful feedback on early drafts of this paper. We also thank Jonathan Park and Rex Chen for their assistance with PlanetLab experiments.

References

[1] Bittorrent announces authorized rollout. `http://www.slyck.com/news.php?story=1090`, Feb. 2006.

[2] CBS to sell new survivor episodes on own site. `http://www.msnbc.msn.com/id/11134875/`, Feb. 2006.

[3] Disney does big business selling tv episodes online. `http://www.showbizdata.com/contacts/picknews.cfm/40484/DISNEY_DOES_BIG_%BUSINESS_SELLING_TV_EPISODES_ONLINE`, Jan. 2006.

[4] Ntl, bittorrent and cachelogic announce joint technology trial. `http://www.cachelogic.com/news/pr100206.php`, 2006.

[5] Quote from PACIFIC BELL: $18000 per month for an OC3 line. `http://shopforoc3.com/`, Mar. 2006.

[6] Universal announces a new download-to-own service. `http://edition.cnn.com/2006/TECH/03/23/movie.download/index.html`, Mar. 2006.

[7] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman. Planetlab: an overlay testbed for broad-coverage services. In *SIGCOMM CCR*, pages 3–12, 2003.

[8] B. Cohen. Incentives build robustness in bittorrent. In *P2P Econ*, 2003.

[9] P. Druschel, A. Nandi, T.-W. J. Ngan, A. Singh, and D. Wallach. Scrivener: Providing incentives in cooperative content distribution systems. In *Middleware*, 2005.

[10] M. W. et al. Seda: An architecture for well-conditioned, scalable internet services. In SOSP, 2001.

[11] M. J. Freedman, E. Freudenthal, and D. Mazires. Democratizing content publication with coral. In *NSDI*, March 2004.

[12] C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *INFOCOM*, 2005.

[13] J. Gray. Distributed computing economics. Technical report, Microsoft Research, 2003. MSR-TR-2003-24.

[14] B. Horne, B. Pinkas, and T. Sander. Escrow services and incentives in peer-to-peer networks. In *EC*, pages 85–94, 2001.

[15] S. Jun and M. Ahamad. Incentives in bittorrent induce free riding. In *P2P Econ*, pages 116–121, 2005.

[16] K. Kong and D. Ghosal. Pseudo-serving: a user-responsible paradigm for internet access. In *WWW*, pages 1053–1064, 1997.

[17] N. Liogkas, R. Nelson, E. Kohler, and L. Zhang. Exploiting bittorrent for fun (but not profi t). In *IPTPS*, 2006.

[18] S. Morris. iTunes outsells CD stores as digital revolution gathers pace. `http://arts.guardian.co.uk/netmusic/story/0,13368,1649421,00.html`, Nov. 2005.

[19] V. S. Pai, P. Druschel, and W. Zwaenepoel. Flash: An effi cient and portable Web server. In *USENIX*, 1999.

[20] K. Park and V. S. Pai. Scale and performance in the coblitz large-fi le distribution service. In *NSDI*, 2006.

[21] J. A. Pouwelse, P. Garbacki, D. H. J. Epema, and H. J. Sips. The bittorrent p2p fi le-sharing system: Measurements and analysis. In *IPTPS*, pages 205–216, 2005.

[22] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Middleware*, pages 329–350, 2001.

[23] B. Schneier. Applied Cryptography, 2nd edition, 1995.

[24] R. Sherwood, R. Braud, and B. Bhattacharjee. Slurpie: A cooperative bulk data transfer protocol. In *INFOCOM*, 2004.

[25] J. Shneidman, D. Parkes, and L. Massoulie. Faithfulness in internet algorithms. In *PINS*, 2004.

[26] Y. J. Song, V. Ramasubramanian, and E. G. Sirer. Cobweb: a proactive analysis-driven approach to content distribution. In *SOSP*, pages 1–3, 2005.

[27] T. Stading, P. Maniatis, and M. Baker. Peer-to-peer caching schemes to address fash crowds. In *IPTPS*, 2002.

[28] A. Stavrou, D. Rubenstein, and S. Sahu. A lightweight, robust p2p system to handle fash crowds. In *ICNP*, pages 226–235, 2002.

[29] K. Tamilmani, V. Pai, and A. Mohr. Swift: A system with incentives for trading. In *P2P Econ*, 2004.

[30] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. Karma: A secure economic framework for p2p resource sharing. In *P2P Econ*, 2003.

[31] L. Wang, K. Park, R. Pang, V. S. Pai, and L. L. Peterson. Reliability and security in the codeen content distribution network. In *USENIX*, pages 171–184, 2004.

[32] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *IEEE Symposium on Research in Security and Privacy*, pages 55–61, 1996.

A Appendix

A.1 Detailed Protocol Description

This section provides a detailed description of the Dandelion peer-serving protocol.

A.1.1 Setting and Assumptions

We assume that the server S keeps a table matching any file F with a pool of available clients currently downloading or seeding the file. A client A gets a temporaty shared key K_{SA} with S. K_{SA} can be efficiently computed as $K_{SA} = H(K_S, \langle A\rangle, \langle i\rangle)$. The notation $\langle X\rangle$ denotes a client X's Dandelion ID, K_S is S's master secret key, H is a cryptographic hash function such as SHA-1, and $\langle i\rangle$ refers to a time period. Our protocol enables S to tolerate some lag in the $\langle i\rangle$ assumed by a client. The temporary shared keys are delivered from the server to the client over a secure channel. For every client, the server S maintains database entries of that client's credit, virtual money which can be used to purchase more services.

A.1.2 Client-Serving Protocol

The protocol starts with the client B sending a request for file $\langle F\rangle$ to S.

1) $B \longrightarrow S$: [server file request] $\langle F\rangle$

If B has access to F, S chooses a random short list of clients $\langle A\rangle_{\text{list}}$, which are currently downloading or seeding the file. Each list entry, besides the Dandelion ID of the client, also contains the clients inbound *internet address*. Also for every client in $\langle A\rangle_{\text{list}}$, S sends a ticket $T_{SA} = MAC_{K_{SA}}[\langle A\rangle, \langle B\rangle, \langle F\rangle, \text{ts}]$ to B. *MAC* is a message authentication code (e.g. an HMAC), ts is a timestamp and $\langle A\rangle$ is a client in $\langle A\rangle_{\text{list}}$. The tickets T_{SA} are only valid for a certain amount of time T (considering clock skew between A,S) and allow B to request chunks of file F from client's A. When T_{SA} expires and B still wishes to download from A it requests a new T_{SA} from S. As commonly done to maintain file integrity, S also sends the *SHA-1* hash $h_{\langle ch\rangle} = H(ch)$ for all chunks ch of the file F S may charge B for the issuance of tickets T_{SA} to prevent misbehaving clients from wasting server resources.

2) $S \longrightarrow B$: [server file response] $T_{SA\ \text{list}}, \langle A\rangle_{\text{list}}, h_{\langle ch\rangle \text{list}}, \langle F\rangle, \text{ts}, \langle i\rangle_S$

The client B forwards this request to each $A \in \langle A\rangle_{\text{list}}$

3) $B \longrightarrow A$: [client file request] $T_{SA}, \langle F\rangle, \text{ts}, \langle i\rangle_S$

If $current - time \leq ts + T$ and T_{SA} is not in A's cache, A verifies if $T_{SA} = MAC_{K_{SA}}[\langle A\rangle, \langle B\rangle, \langle F\rangle, \text{ts}]$.[1] As long as B remains connected to A, it periodically renews its T_{SA} tickets. If the verification fails, A drops this request. Also, if $\langle i\rangle_S$ is greater than A's current epoch $\langle i\rangle_A$, A learns that it should renew its key with S soon. Otherwise, A caches T_{SA} and it starts running a protocol with B for file chunk selection. A reports periodically to B what chunks it has for as long as the timestamp is fresh. Also, B reports its available chunks to A and A can request them from B, after he retrieves T_{SB} from S. B determines which chunks it wishes to download and from which clients according to a chunk selection algorithm. For presentation purposes, each of the following messages involves one chunk, whereas in practice information for multiple chunks may be bundled in a message.

4) $B \longrightarrow A$: [client chunk request] $T_{SA}, \langle F\rangle, \langle ch\rangle, \text{ts}, \langle i\rangle_S$

B's requests are served as long as the timestamp is fresh and T_{SA} is cached or verifies. For each requested chunk, A retrieves and encrypts it using a symmetric-key encryption Enc, as $C = Enc^{\text{iv}_{\langle ch\rangle}}_{k_{\langle ch\rangle}}(ch)$, where $k_{\langle ch\rangle}$ is a randomly chosen key distinct for each chunk, and $\text{iv}_{\langle ch\rangle}$ is the encryption Initial Vector (IV). A encrypts the random key with the one it shares with the server, as $e = Enc^{\text{iv}_{\text{SA}}}_{K_{SA}}(k_{\langle ch\rangle}, \text{iv}_{\langle ch\rangle})$. Finally, A hashes the ciphertext C as $hc = H(C)$ and computes a MAC value $T_{AS} = MAC_{K_{SA}}[\langle A\rangle, \langle B\rangle, \langle F\rangle, \langle ch\rangle, e, hc, \text{ts}]$. Note that A can pre-compute several values $(k_{\langle ch\rangle}, e, C, hc)$, so the on-line cost of A can be reduced to one MAC computation.

5) $A \longrightarrow B$: [client chunk response]$T_{AS}, \langle F\rangle, \langle ch\rangle, e, C, \text{ts}, \langle i\rangle_A$

B retrieves C, computes its own hash $hc' = H(C)$ and forwards the following to S.

6) $B \longrightarrow S$: [decryption key request]$\langle A\rangle, \langle F\rangle, \langle ch\rangle, e, hc', \text{ts}, T_{AS}, \langle i\rangle_A$

If timestamp ts is fresh enough, ticket T_{AS} is not in S's cache, and $\langle i\rangle_A$ is not too much off, S checks if $T_{AS} = MAC_{K_{SA}}[\langle A\rangle, \langle B\rangle, \langle F\rangle, \langle ch\rangle, e, hc', \text{ts}]$, where key K_{SA} is computed using K_S, $\langle A\rangle$, and $\langle i\rangle_A$. The verification may fail either because hc' is invalid due to transmission error in step (5) or because either A or B are misbehaving. Since S is unable to determine which one is the case, it does not punish either clients. Yet it notifies B, which is expected to remove A from its client list in case A repeatedly sents invalid messages. If B keeps sending invalid decryption key requests, S penalizes him. If the verification succeeds, S caches T_{AS}, and checks whether B has sufficient credit. It also checks again whether B has access to the file F. If B is approved, it charges B and reward A. S also decrypts $(k'_{\langle ch\rangle}, \text{iv}'_{\langle ch\rangle}) = Dec_{K_{SA}}(e)$, and sends them to B.

7) $S \longrightarrow B$: [decryption key response]$\langle A\rangle, \langle F\rangle, \langle ch\rangle, (k'_{\langle ch\rangle}, \text{iv}'_{\langle ch\rangle}$

[1]The purpose of this check is to provide a simple mechanism for protecting A from DoS attacks from unauthorized clients and to allow clients to fi lter request for unauthorized fi le uploading.

B uses $(k'_{\langle ch\rangle}, \mathrm{iv}'_{\langle ch\rangle})$ to decrypt the chunk as $ch' = Dec^{\mathrm{iv}'_{\langle ch\rangle}}_{k'_{\langle ch\rangle}}(C)$. If decryption fails or if $H(ch') \neq h_{\langle ch\rangle}$ (see item (2)), then *B* complains to *S* by sending the following message.

8) $B \longrightarrow S$: [complaint], $\langle A\rangle, \langle F\rangle, \langle ch\rangle, T_{AS}, e, hc', \mathrm{ts}, \langle i\rangle_A$

S ignores this message if timestamp ts is not fresh enough (using much more liberal time interval then before) or if this complaint is already cached. If $T_{AS} \neq MAC_{K_{SA}}[\langle A\rangle, \langle B\rangle, \langle F\rangle, \langle ch\rangle, e, hc', \mathrm{ts}]$ *S* punishes *B*, since *B* had already been notified in step (6) that T_{AS} is invalid. If T_{AS} verifies, *S* caches this complaint, recomputes K_{SA} as before, decrypts $(k'_{\langle ch\rangle}, \mathrm{iv}'_{\langle ch\rangle}) = Dec_{K_{SA}}(e)$ once again, retrieves *ch* from its storage, and encrypts *ch* himself using the above key and IV vector, $C' = Enc^{\mathrm{iv}'_{\langle ch\rangle}}_{k'_{\langle ch\rangle}}(ch)$. If the hash of the ciphertext $H(C')$ is equal to the value hc' that *B* sent to *S*, then *S* decides that *A* has acted correctly, *B*'s complaint is unjustified, *S* drops this complaint request and blacklists *B* or charges *B* a large amount. Otherwise, *S* decides that *B* was cheated by *A*, removes *A* from its pool of active clients, blacklists or charges it, and issues an update that cancels the corresponding update on *A*'s and *B*'s credit. Finally, *S* serves *B* itself or it repeats the protocol from step (2).

A.2 Security Analysis

We claim the following security properties of our protocol:

1. If an honest client *B* gets charged (his credit decreases) by *S*, then *B* must have received correct chunk *ch*, even if the transaction involved a malicious client *A*. This is because *B* gets charged only if the data *S* gets in steps (6) and (8) verifies and if $hc' = H(C')$. Since hc' is a hash that *B* computes itself on *C* received from *A*, $C = C'$. Furthermore, since the same k, iv pair is used by *S* to encrypt *ch* into C' and by *B* to decrypt *C* into ch', then $C = C'$ implies that $ch' = ch$.

2. If an honest client *A* always encrypts chunk *ch* anew when servicing a request, then even if client *B* is malicious, if *B* gets *ch* in this protocol instance then *A* also gets credit from *S*. This is because if *A* encrypts *ch* using one-time key $k_{\langle ch\rangle}, \mathrm{iv}_{\langle ch\rangle}$, then *B* sees $k_{\langle ch\rangle}, \mathrm{iv}_{\langle ch\rangle}$ only in the encrypted form *e*. The only way for *B* to get it (short of stealing key K_{SA} from *A* or *S*) is to get it decrypted by *S*, in which case *S* will log a charge against *B*. The only way *B* can possibly avoid this charge is by sending (8) which includes T_{AS} and hc' s.t. $T_{AS} = MAC_{K_{SA}}[\langle A\rangle, \langle B\rangle, \langle ch\rangle, e, h', \mathrm{ts}]$, but such that $hc' \neq H(C')$ where C' is computed by *S* in that step. However, since we consider this attack only against an honest *A*, the T_{AS} MAC value will verify only if all the values it includes are the ones that *A* sent to *B*, and if hash hc' is correctly computed on ciphertext *C* included in that transfer. But if that's the case then *S* will decrypt *e* to the same $k_{\langle ch\rangle}, \mathrm{iv}_{\langle ch\rangle}$ pair that *A* used, hence *S*'s encryption C' will be the same as the *C* that *A* computed. Consequently, hc' will be equal to $H(C')$, hence *B* is not able to reverse its charge.

3. If *A* pre-computes only one encryption of some chunk *ch* and services requests for that file always using the same ciphertext (C, e), then *A* runs some risk that colluding *B*'s can attempt to use *A* to download *ch* with only one of the *B*'s charged for it. Namely, the colluding clients *B*'s have some chance of getting tickets to the same client *A* from *S*, so each of them would receive the same encryption *C* of *ch* from *A*. Then one *B* can incur a charge to retrieve key $k_{\langle ch\rangle}$, but it can share this key with the remaining colluders. The chance of success in such attack decreases if the list of the clients returned by *S* is short and if *A* pre-computes many ciphertext tuples $(k_{\langle ch\rangle}, e, C, h)$ for the same *ch*, and services a request by choosing one of them at random. Note that *A* can individually adjust how much to pre-compute, or even to always encrypt *ch* on-line.

4. A malicious client *B* can always abandon any instance of the protocol or intentionally send invalid messages to *S* (e.g $hc' \neq H(C)$). In such case, *A* does not receive any credit even though *B* consumed *A*'s resources (but also *B* does not receive the file in that instance, as we argue above). This is a denial of service attack against *A*, and we mitigate it by having *S* issue a short-lived MAC'ed ticket T_{SA} only to authorized clients. Therefore *B* can stage this attack against *A* only for as long as the ticket is valid. If *B* is identified as misbehaver client, he will not be issued new tickets. In addition, *S* may charge *B* for the issuance of tickets T_{SA} effectively preventing *B* from maliciously expending both *A*'s and *S*'s resources.

5. Two clients *B* and *A* could agree to share a file that *B* is not entitled to receive based on a file access policy and pretend that *A* is uploading a file to which *B* has access. In that way, *A* would get paid, therefore it has incentives to provide the whole file and violate the policy. However, this is problematic because the complaint mechanism can only rule in favor of *B*, therefore *A* cannot trust that *B* will allow *A* to be rewarded for his service.

Rational Secret Sharing, Revisited

[Extended Abstract]

S. Dov Gordon
gordon@cs.umd.edu

Jonathan Katz[*]
jkatz@cs.umd.edu

Dept. of Computer Science
University of Maryland
College Park, MD 20742

ABSTRACT

We consider the problem of secret sharing among n rational players. This problem was introduced by Halpern and Teague (STOC 2004), who claim that a solution is *impossible* for $n = 2$ but show a solution for the case $n \geq 3$. Counter to their claim, we show a simple protocol for the case of $n = 2$ players. Our protocol extends to the case $n \geq 3$, where it is both simpler than the Halpern-Teague solution and also offers a number of other advantages. We also show how to avoid the continual involvement of the dealer, in either our own protocol or that of Halpern-Teague.

Our techniques extend to the case of rational players trying to securely compute an arbitrary function, under certain assumptions on the utilities of the players.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Systems**]: General—*security and protection*; E.3 [**Data**]: Data Encryption

General Terms

Security

Keywords

Game theory, secret sharing, secure computation

1. INTRODUCTION

The classical problem of *t-out-of-n secret sharing* [10, 1] involves a "dealer" D who wishes to entrust a secret s to a group of n players $P_1, \ldots, P_n$ so that (1) any group of t or more players can reconstruct the secret without further intervention of the dealer, yet (2) any group of fewer than t players has no information about the secret. As an example, consider the scheme due to Shamir [10]: say the secret s lies in a field $\mathbb{F}$, with $|\mathbb{F}| > n$. The dealer chooses a random polynomial $f(x)$ of degree at most $t-1$ subject to the constraint $f(0) = s$, and gives the "share" $f(i)$ to player P_i (for $i = 1, \ldots, n$). Any set of t players can recover $f(x)$ (and hence s) by interpolation; furthermore, no set of fewer than t players has any information about s.

The implicit assumption above is that at least t players are willing to cooperate and pool their shares[1] when it is time to recover the secret; equivalently, at least t players are *honest* and hence at most $n-t$ players are *malicious*. Halpern and Teague [4] consider a scenario in which no one is (completely) honest, but instead all that is guaranteed is that at least t players are *rational* (as before, up to $n-t$ players may refuse to cooperate altogether). Shamir's protocol may no longer succeed in this scenario [4]. Specifically, if all players prefer to learn the secret above all else, and otherwise prefer that fewer parties learn the secret, then no player has any incentive to reveal their share. Consider P_1: if strictly fewer or greater than $t-1$ other players reveal their shares, nothing changes whether P_1 reveals his share or not. On the other hand, if *exactly* $t-1$ other players reveal their shares then P_1 learns the secret regardless (using his share), while P_1 can prevent other players from learning the secret by not publicly revealing his share. Thus, although for $t < n$ having all players reveal their shares is a Nash equilibrium, it is a weakly dominating strategy for each player *not* to reveal its share (and this is the equilibrium likely to be reached). Note that for $t = n$, having all players reveal their shares is not even a Nash equilibrium.

Does there exist *any* protocol for reconstructing the secret in which it is in players' best interests to follow the protocol? Generalizing the above, Halpern and Teague rule out any protocol terminating in a *fixed* number of rounds. This leaves open the possibility of *probabilistic* protocols without a fixed upper bound on their round complexity, and indeed Halpern and Teague show such a protocol for $n \geq 3$ parties. In contrast, they claim that a solution is *impossible* for $n = 2$ even if probabilistic protocols are allowed.

Our results: We revisit the question of rational secret sharing, in the model of [4]. As perhaps our most surprising

[*]Research supported by NSF Trusted Computing grants #0310499 and #0310751; NSF CAREER award #0447075; and US-Israel Binational Science Foundation grant #2004240.

[1]We assume adversarial behavior is limited to refusal to cooperate. Reporting an incorrect share is easily prevented by having the dealer sign the shares.

result, we show a simple, probabilistic protocol for $n = 2$ parties to reconstruct a shared secret, thus disproving the claim of Halpern and Teague. Interestingly, their *proof* appears to be correct; the problem is that their *assumptions* about the types of protocols that might be used are too restrictive. By relaxing their assumptions in a reasonable way, we are able to circumvent their impossibility result.

Our protocol generalizes in a straightforward way to the case of $n \geq 3$ and arbitrary t. Although Halpern and Teague also claim a general solution of this sort, our solution is much simpler and has a number of other advantages.

In the Halpern-Teague solution, the dealer is involved periodically throughout the entire lifetime of the protocol. We also show how to remove this involvement of the dealer, in either our own protocol or that of Halpern and Teague.

Our results extend to the case of rational players computing an *arbitrary* function, under certain assumptions regarding their utilities. See the full version [3] for more details.

Related work: There has been much recent interest in bridging cryptography and game theory [6, 5, 7]. While prior work [6, 5] offers solutions to the problem considered here, we focus on simplicity and efficiency rather than generality. Our work also makes weaker physical assumptions than that of [6, 5]: specifically, we assume *simultaneous broadcast* (equivalently, we do not allow *rushing*) rather than "*secure envelopes*." See [3] for further discussion.

Recent and independent work [8] shows how to use essentially the same ideas shown here to obtain a stronger and more general result.

2. MODEL

We review the model of Halpern and Teague, filling in some details they omit. We have a dealer D holding a secret s, and n players. A protocol proceeds in a sequence of *iterations*, where each iteration consists of multiple *rounds*. At the beginning of each iteration, D distributes some information (privately) to the n players; at this point, any set of fewer than t players should have no information about s. The dealer is not involved during an iteration. Instead, some set of at least t players run the protocol amongst themselves by simultaneously broadcasting messages in a series of rounds. (Halpern-Teague additionally allow private communication between the players but we do not need this.) At the end of an iteration, the protocol either terminates or proceeds to the next iteration. We assume the dealer follows the protocol as specified. To rule out trivial protocols, we require that if at least t players follow the protocol in each iteration, the secret is eventually reconstructed.

Let $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ denote a vector of (possibly randomized) strategies used by the players. A *protocol* corresponds to the above *game* along with a prescribed strategy vector $\vec{\sigma}$. As in [4], we are interested in strategy vectors corresponding to a Nash equilibrium that survives iterated deletion of weakly-dominated strategies. See [9, 4] for definitions.

Let $\mu_i(\vec{\sigma})$ denote the utility of P_i for the strategy vector $\vec{\sigma}$. For a particular outcome o of the protocol, we let $\delta_i(o)$ be a bit denoting whether or not P_i learns the secret, and let $\mathsf{num}(o) = \sum_i \delta_i(o)$. We assume that for all i:

- $\delta_i(o) > \delta_i(o') \Rightarrow \mu_i(o) > \mu_i(o')$.
- If $\delta_i(o) = \delta_i(o')$,
  $\mathsf{num}(o) < \mathsf{num}(o') \Rightarrow \mu_i(o) > \mu_i(o')$.

That is, players first prefer outcomes in which they learn the secret; if this is held fixed, players prefer outcomes in which the fewest other players learn the secret. Let $U_i(\vec{\sigma})$ denote the expected value of the utility of P_i under strategy vector $\vec{\sigma}$. We assume rational players wish to maximize this value.

3. PROTOCOLS

We provide a high-level overview of the Halpern-Teague solution for 3-out-of-3 secret sharing. (Details of their proposed generalization for $n > 3, t \geq 3$ are in [4, 3].) At the beginning of each iteration, the dealer runs a fresh invocation of the Shamir scheme and sends the appropriate share to each player. During an iteration, each P_i flips a biased coin c_i with $\Pr[c_i = 1] = \alpha$. Players then securely compute $p = \bigoplus c_i$ such that it is impossible to cheat or to learn information about the $\{c_i\}$ values of the other parties. If $p = c_i = 1$, player P_i broadcasts his share. If all shares are revealed, the secret is reconstructed and the protocol ends. If $p = 1$ and no shares are revealed, all players terminate the protocol. In any other case, players proceed to the next iteration. Assuming players act honestly, the expected number of iterations until the protocol terminates is α^{-3}.

For a quick sketch as to why this works, assume P_1, P_2 follow the protocol and consider whether P_3 should deviate. One can show that there is no incentive for P_3 to change the distribution of c_3. If $p = 0$ or $c_3 = 0$, there is clearly no incentive for P_3 to deviate. When $p = c_3 = 1$, player P_3 does not know whether $c_1 = c_2 = 1$ (which occurs with probability $\frac{\alpha^2}{\alpha^2 + (1-\alpha)^2}$) or $c_1 = c_2 = 0$ (which occurs with the remaining probability). If P_3 does not broadcast its share it runs the risk of having the protocol terminate without learning the secret. Setting α appropriately based on P_3's utility function, it is not in P_3's best interest to deviate.

3.1 Our Solution

Halpern and Teague implicitly assume that the dealer is restricted to sending valid Shamir shares to the players, and their impossibility proof for $n = 2$ therefore focuses only on what happens *during* an iteration. Removing this restriction circumvents the impossibility result for $n = 2$ (and also drastically simplifies things for the case of general n [3]).

The idea is as follows: with some probability the dealer shares the *actual* secret, and with the remaining probability the dealer shares a "bogus" secret. No player can tell which is the case given the share he receives. Then, players simply pool their shares and reconstruct the shared value. If this is the "actual" secret, the protocol terminates; otherwise, players continue to the next iteration. (We have to provide the players with a way to detect whether a reconstructed value is the actual secret or not; this is quite easy to do.)

We focus on the case $n = 2$ but it is easy to see that our idea generalizes to arbitrary t, n. Say the dealer holds a secret s that lies in a *strict subset* S of a field $\mathbb{F}$ (if s lies in a field $\mathbb{F}'$, this is achieves by taking a larger field $\mathbb{F}$ containing $\mathbb{F}'$ as a subfield). At the beginning of each iteration, with probability β the dealer generates a random sharing of s, and with probability $1-\beta$ the dealer generates a random sharing of an arbitrary element $\hat{s} \in \mathbb{F} \setminus S$. In a given iteration, the players simply broadcast their shares. If in any iteration some player does not broadcast their share, both players immediately terminate the protocol. Otherwise, both shares were broadcast and the players either reconstruct the secret

$s \in S$ and terminate the protocol successfully, or reconstruct a value $\hat{s} \in \mathbb{F} \setminus S$ and proceed to the next iteration.

To see why this works, note first that a player cannot tell from its share whether the dealer has shared the "real" secret s or the "bogus" secret $\hat{s}$. Assume P_1 acts honestly and consider whether P_2 has any incentive to deviate. The only possible deviation is for P_2 to refuse to broadcast his share. In this case, it learns the secret (while P_1 does not) with probability β, but with probability $1 - \beta$ it will never learn the secret. Setting β appropriately depending on P_2's utility, it is not in P_2's best interest to deviate.

The above shows that following the protocol is a Nash equilibrium. It is possible to additionally prove that it survives iterated deletion of weakly dominated strategies [3].

4. DISCUSSION AND EXTENSIONS

We view the main import of our result as a demonstration that rational secret sharing is, in fact, possible when $n = 2$; this serves as an illustration of the sensitivity of an impossibility result to the precise model under consideration. Our approach also has various other advantages as compared to the Halpern-Teague solution; chief among these may be its *simplicity*. See [3] for additional points of comparison.

In the full version [3], we show how to remove the need for the dealer to be involved at the beginning of each iteration so that, as in standard secret sharing, the dealer need only be involved *once*, at the beginning of the protocol. We also show how our results may be extended to the case of secure computation of arbitrary functions (à la [2]) by parties assumed only to be *rational* (i.e., without making the assumption that any parties are completely honest [2]).

5. REFERENCES

[1] G.R. Blakley. Safeguarding Cryptographic Keys. *National Computer Conference*, AFIPS Press, 1979.

[2] O. Goldreich, S. Micali, and A. Wigderson. How to Play any Mental Game or A Completeness Theorem for Protocols with Honest Majority. STOC '87.

[3] S.D. Gordon and J. Katz. Rational Secret Sharing, Revisited. Available at http://eprint.iacr.org/2006/142.

[4] J. Halpern and V. Teague. Rational Secret Sharing and Multiparty Computation. STOC 2004.

[5] S. Izmalkov, S. Micali, and M. Lepinski. Rational Secure Function Evaluation and Ideal Mechanism Design. FOCS 2005.

[6] M. Lepinski, S. Micali, C, Peikert, and A. Shelat. Completely Fair SFE and Coalition-Safe Cheap Talk. PODC 2004.

[7] M. Lepinski, S. Micali, and A. Shelat. Collusion-Free Protocols. STOC 2005.

[8] A. Lysyanskaya and N. Triandopoulos. Rationality and Adversarial Behavior in Multi-Party Computation. Crypto 2006, to appear.

[9] M.J. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 1994.

[10] A. Shamir. How to share a secret. *Comm. ACM*, 22(11): 612–613 (1979).

Path Auction Games When an Agent Can Own Multiple Edges [*]

Ye Du [†] Rahul Sami[§]
Yaoyun Shi [†]
[†] Department of Electrical Engineering and Computer Science, University of Michigan
2260 Hayward Ave, Ann Arbor, MI 48109-2121, USA
Email: duye|shiyy@umich.edu
[§]School of Information, University of Michigan, Ann Arbor, MI, 48109, USA
Email: rsami@umich.edu

Abstract

We study path auction games in which multiple edges may be owned by the same agent in this paper. The edge costs and the set of edges owned by the same agent are privately known to the owner of the edge. We show that in this setting, given the assumption the losing agent always has 0 payoff, there is no *individual rational* strategyproof mechanism in which only edge costs are reported. If the agents are asked to report costs as well as ownership, we show that there is no efficient mechanism that is *false-name proof.* We then study a first-price path auction in this model. We show that, in the special case of parallel-path graphs, there is always a pure-strategy ϵ-Nash equilibrium in bids. We show that this result does not extend to general graphs: we construct a graph in which there is no such ϵ-Nash equilibrium.

1 Introduction and Motivation

In the *path auction game*, there is a network $G = (V, E)$, in which each edge $e \in E$ is owned by an agent. The true cost of e is private information and known only to the owner. Given two vertices, source s and destination t, the customer's task is to buy a path from s to t. This path auction can be used to model problems in supply chain management, transportation management, QoS routing and other domains. Recently, path auctions have been extensively studied [11, 9, 2, 8, 4]; much of this literature has focused on the Vickrey-Clarke-Groves (VCG) mechanism [12, 3, 6]. In the VCG mechanism, the customer pays each agent on the winning path an amount equal to the highest bid with which the agent would still be on the winning path.This mechanism is attractive because it is efficient and*strategyproof*,*i.e.,* the dominant strategy for each agent is to report its true cost.

In the traditional path auction model, each agent only owns one edge in the graph, and there is no cooperation between agents. Here, we study a variant of the path auction game in which *each agent may own multiple edges.* In this extended model, if the ownership information is publicly available (i.e. the customer knows which agent owns which edge), the VCG mechanism design approach yields a strategyproof mechanism.

In practice, however, the ownership information is more likely to be private – it could be costly for the customer to find out the true ownership information, or the agent may have an incentive to hide its true ownership information in order to get better payoff. For example, in Figure 1, there are two agents: a and b. Agent a owns edges (s, i) and (i, t) with true cost 1 each; agent b owns edges (s, j) and (j, t) with true cost 2 each. If agents a and b reveal the true ownership information to the customer, the most natural VCG mechanism will choose path (s, i), (i, t) as the winning path and pay agent a an amount equal to 2. However, if agent a hides its ownership information, the mechanism will treat edges (s, i) and (i, t) as owned by different agents. When the agents bid their true costs, the winning path stays the same, but the payment to agent a would be $2 \times 3 = 6$. Moreover, when the ownership information is not available to the customer, agent

[*][†]Supported in part by NSF grant 0347078.

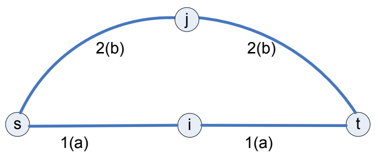


Figure 1: VCG mechanism is not strategyproof for this game

a can increase its payoff by bidding lower than its true cost. For example, it can bid 0.5 for both edges (s,i) and (i,t). This does not change the winning path, but the payoff to agent a would increase to $2 \times 3.5 = 7$. Hence, the straightforward VCG mechanism, which assumes that each edge is owned by an individual agent, is not strategyproof. In this paper, we model situations in which each agent can own multiple edges at the same time, but the ownership information is private. Thus the traditional path auction model is a special case of our extended model. One real-life example of our model is an online auction system in which each seller/buyer can have multiple accounts in the system. Now, if a buyer wants some combination of goods that can be expressed in path auction form, it is hard for her to find the true identity of each seller account, and so she is faced with the unknown-ownership scenario.

In this paper, we analyze path auctions under two solution concepts: dominant strategies and Nash equilibrium in bids. We begin by studying *truthful* dominant strategy mechanisms, *i.e.* strategyproof mechanisms. We show that if the agents only submit bid prices in the auction for each edge, there is no strategyproof mechanism that satisfies individual rationality under assumption that the losing agent always has 0 payoff. The natural extension is to consider mechanisms in which agents are invited to reveal their entire private information, the ownership of edges as well as the costs. An important strategic property in this setting is that the mechanism is *false-name proof* [13], *i.e.*, an agent cannot gain by dividing her owned edges among two or more pseodonyms. We show that earlier results on false-name proof mechanisms [13] imply that there is no Pareto-efficient false-name proof mechanism in the extended auction format.

We next turn to a first-price auction bidding game, and study ϵ-Nash equilibria of this game. For the class of parallel-path graphs, we constructively prove that at least one ϵ-Nash equilibrium exists, and we prove a lower bound on the total payments in any such equilibrium. However, we find a non-parallel-path graph which can be proven not to have a pure strategy ϵ-Nash equilibrium.

Please note that all proofs have been deferred to the appendix.

1.1 Related work

Path auction games have been extensively studied in recent years. Nisan and Ronen introduced the shortest-path game in their paper on algorithmic mechanism design [11], and showed that the VCG mechanism for this problem is computationally tractable. Hershberger and Suri [7] described an improved algorithm for this problem. However, several authors have noted that the VCG mechanism may pay much higher than the true cost of the winning path. This has led to the study of the frugality [2] of VCG mechanism. Archer and Tardos [2], and Elkind *et. al* [4] studied the frugal path auction mechanism, and showed that the payments can be arbitrarily high. Karlin and Kempe [9] extended the path model to a more general set system model and introduced a new frugality ratio definition; they designed a mechanism that performs better than VCG in path auction. The problem of agents owning multiple edges was mentioned as future work in [9]. Immorlica *et. al.* [8] studied first-price path auctions in the traditional single-ownership setting. They showed the existence of a strong ϵ-Nash equilibrium in bids, and bounded the payments in equilibrium. Yokoo *et. al.* [13] introduced the concept of false-name proof mechanisms, and showed that in combinatorial auctions, there is no false-name proof mechanism that satisfies Pareto efficiency.

2 Definitions and Problem Statement

First, we introduce the formal definition of path auction based on the definition of set system in [9].

The simple model of path auction: Given a graph $G = (V, E)$, each edge $e \in E$ is owned by an agent and has a cost c_e, the true cost it incurs if it is selected. This value is private, i.e. known only to the agent which owns e. We define the feasible set $F = \bigcup_i P^i(s,t)$, where $P^i(s,t)$ is the ith path from s to t. Given two specific vertices s(the source) and t(the destination), the task of the customer is to buy a path from s to t by auction. It consists of the following two steps:

1. Each agent submits a sealed bid b_e to the customer. The bidding vector b is $(b_{e_1}, b_{e_2}, ..., b_{e_m})$, where b_{e_i} is the bidding price for edge $e_i \in E$. Moreover, let B denote the bidding space that is the set of all possible bidding vectors.

2. Given the bidding vector b, the customer selects a path P^i from the feasible set F as the winning path, and computes a payment $p_e \geq b_e$ for each edge $e \in P^i$. We say that if an agent owns an edge e on the winning path P^i, it *wins*, and all other agents *lose*.

In order to implement the auction, we need to design a mechanism $(f, p_1, p_2, ..., p_m)$, where $f : B \rightarrow F$ selects one element in the feasible set as the winning path and $p_i : B \rightarrow R$ computes the payment to agent i. Moreover, we assume that:

- (G, F) is common knowledge to the customer and all agents.
- the game is *monopoly free*, which means no edge is in all feasible sets, i.e. $\bigcap_{P^i(s,t)\in F} P^i(s,t) = \varnothing$.
- the agent is rational and has quasilinear utility, i.e., the agent want to maximize its utility: $u_e = p_e(b) - c_e$ if e is on the winning path; or else $u_e = 0$.

Definition 1. A mechanism is *strategyproof* if, for any agent i that owns edge e, any $b_{-i} \in B_{-i}$ and $b_i^{'}$, $p_i(c_e, b_{-i}) - c_e \geq p_i(b_i^{'}, b_{-i}) - c_e$, where b_{-i} is the bidding vector of all agents except i.

The VCG mechanism is strategyproof in the simple path auction game, *i.e.* the dominant strategy of each agent is to bid its true cost in VCG mechanism.

In the simple path auction model, each agent only owns one edge in the graph. We extend the model in the following way:

The extended model of path auction: Now assume that each agent can own multiple edges. We can partition the edge set E as: $E = \bigcup_i E_i$, where E_i is the set of edges owned by agent i. We also assume that if agent i owns k edges, *i.e.* $|E_i| = k$, it has k identities $ID_i = \{ID_{i1}, ID_{i2} ..., ID_{ik}\}$, one for each edge to use in the auction. In the extended model, a game is monopoly free if for any agent i, there is at least one path between s and t in graph $(V, E \setminus E_i)$. A mechanism is strategyproof if for any agent, the dominant strategy is to bid the true cost for each edge it owns. Moreover let p_i denote the payment to agent i. Then p_i is equal to $\sum\limits_{e_j \in E_i} p_{e_j}$. According to the type of bidding space, we can define two types of auctions:

Path Auction of Type I: In this type of auction, the agent is only asked to submit the bidding price for each edge it owns. The mechanism will select the winning path and compute the payment to each edge.

Path Auction of Type II: In this type of auction, the agent is asked to submit the ownership information about which set of edges it owns and the bidding price for each edge it claims to own. Let $o = (o_{e_1}, o_{e_2}, ..., o_{e_m})$ be the claimed ownership information vector, where if edge e_j is owned by agent i (*i.e.* $e_j \in E_i$), $o_{e_j} \in ID_i$. We assume that no more than one agent claims to own the same edge and each edge is claimed to be owned by some agent. Since the agent has one identity for each edge it owns, it can choose arbitrary strategy to report the ownership information for edges owned by itself.

We will not only study the strategyproof mechanism in the above two types of auctions, but also a weaker solution concept: ϵ-Nash equilibrium.

Definition 2. An ϵ-Nash equilibrium for a game is a set of strategies, one for each player, such that no player can unilaterally deviate in a way that improves its payoff by at least ϵ.

3 The Nonexistence of strategyproof Mechanism

In the extended model of path auction, the question to answer is: Is it possible to design a mechanism such that it is in every agent's best interest to bid her true cost? We focus on the auction of type I in subsection 3.1 and the auction of type II in subsection 3.2.

3.1 No individual rational strategyproof mechanism in auction of type I

In auction of type I, we can construct a trivial strategyproof mechanism, which always selects a fixed path as the winning path and pays a fixed amount of money to the edges on the path. We call such a mechanism the *dictator mechanism*. It is not hard to verify that the dictator mechanism is strategyproof, but it might not be *individual rational*. The definition of individual rational is:

Definition 3. A mechanism is *individual rational* if, for any agent i, the payment to itself is at least the true incurred cost when it is selected by the mechanism, *i.e.* $p_i \geq c_i$.

Based on the definition of individual rationality, we have the following theorem:

Theorem 1. *Given the assumption that the losing agent always has 0 payoff, there is no strategyproof mechanism for auction of type I that satisfies individual rationality.*

We believe that if we remove the assumption that the losing agent always has 0 payoff, the theorem still holds. It would be interesting to find a simple proof for such extension of theorem 1.

3.2 No false-name proof mechanism in auction of type II that satisfies Pareto efficiency

As shown in previous subsection, if the agent only submits the bidding price information, it is almost impossible to enforce the agent to bid its true cost. In order to make the agent bid truthfully, the customer may ask the agents to reveal more information, such as the ownership information, besides the bidding price information. Therefore we consider auction of type II. First we give the definition of false-name proof mechanism [13] in the context of path auction game.

Definition 4. A mechanism is false-name proof if for any fixed bidding vector b_{-i} and the claimed ownership vector o_{-i} by all agents other than i, it is agent i's best interest to bid the true cost of each edge it owns, *i.e.* $b_i = (c_{e_{i1}}, c_{e_{i2}}, ..., c_{e_{ik}})$ where $E_i = \{e_{i1}, e_{i2}, ..., e_{ik}\}$, and to claim the real ownership information $o_i = \underbrace{(ID_{ij}, ID_{ij}, ..., ID_{ij})}_{k}$ where $1 \leq j \leq k$.

For situations in which the true ownership cannot be determined, a false-name false-name proof mechanism [13] is desirable. The next natural question is: Is it possible to design a false-name proof mechanism in the extended model of path auction game? Yokoo *et. al.* [13] showed the following impossibility result for combinatorial auctions:

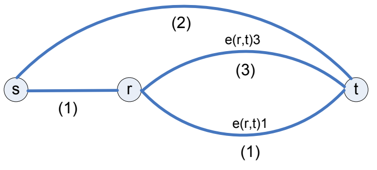


Figure 2: There is no false-name proof mechanism which satisfies Pareto efficiency in this game.

Proposition 1. [13] In combinatorial auctions, there is no false-name proof auction protocol that satisfies Pareto efficiency. □

The definition of Pareto efficiency is:

Definition 5. A winning path selection mechanism is *Pareto efficient* if given the winning path $P^i(s,t)$, $\forall k$, $\sum_{e \in P^i(s,t)} c_e \leq \sum_{e \in P^k(s,t)} c_e$, which means that the mechanism always selects the path from s to t with minimum cost.

The above proposition is proved by constructing a generic counter example. Since path auction is only an instance of more general class of combinatorial auctions, it might be possible to design a false-name proof mechanism for path auction even the impossibility result holds for combinatorial auctions. However, the generic counter example constructed in [13] can be easily transformed to a path auction game and show the impossibility result in auction of type II.

Proposition 2. In the extended model of path auction game, there is no false-name proof mechanism for auction of type II that satisfies Pareto efficiency.

4 Existence of ϵ-Nash Equilibrium

Since strategyproof mechanism is not widely achievable in the extended model of path auction game, we need to extend the solution concept from dominant strategies to non dominant strategies. The concept of ϵ-Nash equilibrium is an important candidate. In this section, we study the existence of ϵ-Nash equilibrium under the VCG mechanism and the *first-price auction* mechanism [8], which elicits the bids from the agents, chooses the cheapest path respect to the bidding vector as the winning path and pays each winning agent exactly the bidding price.

Since VCG is not strategyproof in the extended model, a natural question to ask is: If we apply VCG mechanism, is there an equilibrium in the resulting game? For the game in Figure 1, suppose b is the bidding vector that reaches an ϵ-Nash equilibrium. As the straightforward VCG mechanism assumes each edge is owned by an individual agent, whatever the winning path is in Figure 1, the winning agent can increase its payoff by decreasing its bidding vector until its bidding prices reach 0. This implies that the winning agents have the incentive to bid as low as they can if all other agents bid truthfully. We will exclude such equilibrium from discussion.

Now, we would like to study first price auction mechanism in the extended model. In practice, a rational agent is not willing to bid below the true cost for each edge in first price auction because such strategy may incur negative payoff to the agent. Therefore, we assume that the bidding price of each edge is at least its true cost, *i.e.* $\forall e, b_e \geq c_e$, when we discuss ϵ-Nash equilibrium in the following. In the next, we would like to show the existence of ϵ-Nash equilibrium in the *parallel-path graph* [5], which can be defined inductively as:

Definition 6. A parallel-path graph(PPG) is a network (V, E, s, t), such that one of the following conditions is satisfied:

Base Case: A path from s to t is a PPG;

Parallel: Suppose $G_1 = (V_1, E_1, s, t)$ and $G_2 = (V_2, E_2, s, t)$ are PPG such that $V_1 \bigcap V_2 = \varnothing$ and $E_1 \bigcap E_2 = \varnothing$. Set $V = V_1 \bigcup V_2$ and $E = E_1 \bigcup E_2$, then (V, E, s, t) is a PPG.

Given the definition of parallel-path graph, we can prove the following theorem:

Theorem 2. *If the underlying network is a parallel-path graph, the first-price path auction has an ϵ-Nash equilibrium.*

Since the underlying network (V, E, s, t) is a parallel-path graph, we can represent it as $\bigcup\limits_k P^k(s,t)$, where $P^k(s,t)$ is the kth path from s to t and $\forall i \neq j$, $P^j(s,t) \bigcap P^j(s,t) = \varnothing$. Moreover, let $C(P^k(s,t)) = \sum\limits_{e \in P^k(s,t)} c_e$ denote the cost of path $P^k(s,t)$ with respect to true cost vector c. We sort the paths from low to high according to their costs, i.e. the path with lower cost has smaller index. If agent A_i owns at least one edge on the cheapest path $P^1(s,t)$, let $LPI(A_i)$ be the smallest path index such that path $P^{LPI(A_i)}(s,t)$ does not have an edge owned by agent A_i but for any path that has smaller path index than $LPI(A_i)$, it must have at least one edge owned by agent A_i. We compute $LPI(A_i)$ for each agent A_i that owns at least one edge on $P^1(s,t)$. Suppose agent A_k has the highest value: $LPI(A_k)$ (break ties arbitrarily), we can bound the payment of any ϵ-Nash equilibrium in the following corollary, which is derived directly from the proof of theorem 2.

Corollary 1. The total payment in any ϵ-Nash equilibrium is at least: $C(P^{LPI(A_k)}(s,t))$.

The lower bound given in corollary 1 is tight. In order to study the frugality of first price auction mechanism in our model, an interesting question is to find out the upper bound of the payment in any ϵ-Nash equilibrium for parallel-path graph. A small value upper bound will imply that first-price auction mechanism is frugal in our model.

Although, there exists an ϵ-Nash equilibrium for parallel-path graph, we can find a non-parallel-path graph that does not have a pure-strategy ϵ-Nash equilibrium. We show this counter example in Figure 3 and the following proposition proves this result. Please note that in Figure 3, the integer number in the bracket denotes the identity of the agent who owns that edge.

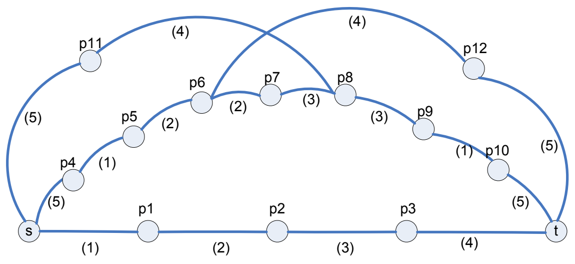


Figure 3: There is no pure-strategy ϵ-Nash equilibrium in this first-price path auction game.

Proposition 3. Given the assumption that each edge's bidding price is at least its true cost, i.e. $\forall e \in E, b_e \geq c_e$, the graph showed in Figure 3 can not have a pure-strategy ϵ-Nash equilibrium in first-price path auction.

5 Conclusion And Future Work

In this paper, we studied the path auction games in which an agent can own multiple edges. Our model is more general than the simple path auction model.

However, our results show that strategyproofness is not widely achievable in the extended model; moreover, general graphs may not have a pure-strategy ϵ-Nash equilibrium in first-price path auction mechanism. Therefore, our model leaves a few challenges.

In this paper, although we have found an ϵ-Nash equilibrium for parallel-path graph, we do not have a mechanism such that when the agents play the game under the mechanism, they can reach the ϵ-Nash equilibrium. So a natural open problem is to design such a mechanism. Moreover, we believe that there exists an ϵ-Nash equilibrium for series parallel-graph [5]. It would be interesting to extend the result of theorem 2 to more general class of graphs.

For the non-parallel-path graphs, we found a counter example which does not have a pure-strategy ϵ-Nash equilibrium in first price path auction mechanism. An interesting question is to analyze the mixed strategy Nash equilibrium or Bayes-Nash equilibrium given some distribution on the edge costs.

Acknowledgements

We would like to thank Tilman Börgers for helpful discussions. We also thank Mike Wellman for pointing out the previous work on false-name proof bidding in combinatorial auctions.

References

[1] Aaron Archer and Eva Tardos. Truthful mechanisms for one-parameter agents. *In Proceeding of Symposium on Foundations of Computer Science*, pages 482–491, 2001.

[2] Aaron Archer and Eva Tardos. Frugal path mechanism. *In Proceedings of the 2002 Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 991–999, 2002.

[3] E. Clarke. Multipart pricing of public goods. *Public Choice*, 11:17–33, 1971.

[4] E. Elkind, A. Sahai, and K. Steiglitz. Frugality in path auctions. *In Proceeding of 15th ACM Symposium on Discrete Algorithm*, 2004.

[5] Edith Elkind. True costs of cheap labor are hard to measure: Edge deletion and vcg payments in graphs. *In Proceeding of 7th ACM conference on Electronic Commerce*, 2005.

[6] Theodore Groves. Incentives in teams. *Econometrica*, 41:617–663, 1973.

[7] John Hershberger and Subhash Suri. Vickrey pricing in network routing: Fast payment computation. *In Proc. of the 42nd IEEE Symposium on Foundations of Computer Science*, 2001.

[8] Nicole Immorlica, David Karger, Evdokia Nikolova, and Rahul Sami. First-price path auctions. *In Proceeding of 7th ACM conference on Electronic Commerce*, 2005.

[9] Anna R. Karlin and David Kempe. Beyond vcg: Frugality of truthful mechanisms. *In Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, 2005.

[10] V. Krishna. Auction theory. *Academic Press*, 2002.

[11] N. Nisan and A. Ronen. Algorithmic mechanism design. *In Proceeding of 31st Annual ACM Symposium on Theory of Computation*, pages 129–140, 1999.

[12] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.

[13] M. Yokoo, Y. Sakuri, and S. Matsubara. The effect of false-name bids in combinatorial auctions: New fraud in internet auctions. *Games and Economics Behavior*, 46:174–188, 2004.

APPENDIX

Theorem 3. *[9, 1, 10] A strategyproof mechanism has the following two properties.*

1. *A mechanism is strategyproof only if the selection rule is monotone: No losing agent can become a winner by rasing his bid, given fixed bids by all other agents.*
2. *Given a monotone selection rule, there is a unique strategyproof mechanism with this selection rule. This mechanism pays each agent his threshold bid, i.e. the highest value he could have bid and won.*

Lemma 1. *In the extended path auction model, given the assumption that the losing agent always has 0 payoff, for any individual rational strategyproof mechanism $(f, p_{e_1}, ..., p_{e_m})$ and a given* *strictly positive bidding vector $b^1 = (b^1_{e_1}, ..., b^1_{e_m})$, we can construct another bidding vector $b^{'} = (b^{'}_{e_1}, ..., b^{'}_{e_m})$ such that when the agents bid according to $b^{'}$, all the edges on the winning path have positive payoffs. Moreover, $\forall j, |b^1_{e_j} - b^{'}_{e_j}| \leq \epsilon$, where ϵ is a small positive number.*

Proof: Suppose initially when the agents bid according to b^1, the winning path is P^1, *i.e.* $f(b^1) = P^1$. Moreover, we assume that all the losing agents have payoff zero.

In the first round, for an edge $e_1 \in P^1$, we decrease the bidding price of it from $b^1_{e_1}$ to $b^1_{e_1} - \epsilon$ and keep the biding prices of all other edges unchanged. Let $\mathcal{TWP} = \{e_1\}$. Thus we get a new bidding vector b^2 and $f(b^2) = P^2$. According to theorem 3, e_1 must be on the new winning path P^2. Moreover, the payment to edge e_1 should not change, *i.e.* $p_{e_1}(b^1) = p_{e_1}(b^2)$. Or else, if $p_{e_1}(b^1) > p_{e_1}(b^2)$, when the true cost of edge e_1 is $b^1_{e_1} - \epsilon$, edge e_1 can improve its payoff by increasing its bidding price to $b^1_{e_1}$. If $p_{e_1}(b^1) < p_{e_1}(b^2)$, when the true cost of edge e_1 is $b^1_{e_1}$, edge e_1 can improve its payoff by decreasing its bidding price to $b^1_{e_1} - \epsilon$. Both cases contradict to the strategyproofness. Since $p_{e_1}(b^1) = p_{e_1}(b^2)$, when edge e_1 decreases its bidding price from $b^1_{e_1}$ to $b^1_{e_1} - \epsilon$, its payoff will increase by ϵ.

In the kth round where $k \geq 2$, for an edge $e_k \in P^k$ but $e_k \notin \mathcal{TWP}$, we decrease the bidding price of it from $b^k_{e_k}$ to $b^k_{e_k} - \frac{\epsilon}{2^{k-1}}$ and keep the biding prices of all other edges unchanged. Thus we get a new bidding vector b^{k+1} and $f(b^{k+1}) = P^{k+1}$. Let $\mathcal{TWP} = \mathcal{TWP} \bigcup \{e_k\}$. Similar to the above arguments, edge e_k must be on the new winning path P^{k+1} and its payoff is increased by $\frac{\epsilon}{2^{k-1}}$ because the payment to it does not change. Moreover, any edge $e_j \in \mathcal{TWP}$ is still on the new winning path P^{k+1} and its payoff cannot decrease by more than $\frac{\epsilon}{2^{k-1}}$, *i.e.* $p_{e_j}(b^{k+1}) - p_{e_j}(b^k) \geq -\frac{\epsilon}{2^{k-1}}$. If $p_{e_j}(b^{k+1}) - p_{e_j}(b^k) < -\frac{\epsilon}{2^{k-1}}$, when an agent i owns both edges e_j and e_k, and the true cost of edge e_k is $b^k_{e_k} - \frac{\epsilon}{2^{k-1}}$, agent i can increase its payoff by increasing the bidding price of edge e_k to $b^k_{e_k}$. This contradicts to the strategyproofness. Since the payoff of edge e_j cannot decrease by more than $\frac{\epsilon}{2^{k-1}}$ and for any k and a finite number $N > k$, $\frac{\epsilon}{2^k} > \sum_{i=k+1}^{N} \frac{\epsilon}{2^i}$, it implies that edge e_j is still on the new winning path P^{k+1} and it has positive payoff.

Finally, when the process is terminated, the winning path $P = \mathcal{TWP}$ and the final bidding vector is $b^{'}$. According to the argument above, all the edges on the winning path must have positive payoffs when the agents bid according to $b^{'}$. □

Theorem 1. *Given the assumption that the losing agent always has 0 payoff, there is no strategyproof mechanism for auction of type I that satisfies individual rationality.*

Proof: According to lemma 1, for any individual rational strategyproof mechanism $(f, p_{e_1}, ..., p_{e_m})$, we can construct a sequence of bidding vectors $b(r) = (r \times b_{e_1} - \epsilon(e_1, r), ..., r \times b_{e_m} - \epsilon(e_m, r))$ such that the winning agents always have positive payoffs, where $r \in \mathcal{N}$ and $\forall j, \epsilon(e_j, r)$ is a small positive number. Let $\mathcal{PB} = \{b(r) | r \in \mathcal{N}\}$ denote the set of all such bidding vectors. For each $b(r) \in \mathcal{PB}$, $f(b(r))$ will select a winning path. Since there are infinite number of $b(r)$s, but only finite number of possible winning paths, there must be an infinite subsequence $\mathcal{SPB} = \{b(r_1), b(r_2)......\}$ such that $\forall b(r_i) \in \mathcal{SPB}$, $f(b(r_i)) = P$ always selects P as the winning path. According to the assumption of individual rationality, and given that the payment to each edge is finite, we can find two bidding vectors $b(r_p), b(r_q) \in \mathcal{SPB}$ such that for any edge e_j on the winning path P, $b(r_p)_{e_j} \leq p_{e_j}(b(r_p)) < b(r_q)_{e_j} \leq p_{e_j}(b(r_q))$.

Given a bidding vector b such that the losing agent has payoff 0 while the winning agent has positive payoff. If for an edge e_j not on P, increasing b_{e_j} to $b^{'}_{e_j}$ can change the winning path from P to $P^{'}$ *i.e.* there exists an edge $e \in P$ but $e \notin P^{'}$, we can get some contradiction. According to theorem 3, e_j cannot be on $P^{'}$. Thus its payoff is still 0. Assume an agent i owns both e_j and e. Since e has positive payoff when it is on winning path P, if the true cost of e_j is $b^{'}_{e_j}$, agent i can increase its payoff by understating e_j's true cost as b_{e_j}. This contradicts to the strategyproofness. W.O.L.G. we assume that the winning path is a simple path, thus $P = P^{'}$. Moreover, increasing b_{e_j} does not change the payment to any edge. For any edge $e \notin P$, the payment to it is always 0. Suppose increasing b_{e_j} to $b^{'}_{e_j}$ can increase the payment to edge $e \in P$ from p_e to $p^{'}_e$, *i.e.* $p_e < p^{'}_e$. When an agent i owns both e_j and e, and the true cost of edge e_j is b_{e_j}, agent i can increase its payoff by overstating e_j's cost as $b^{'}_{e_j}$. This contradicts to the strategyproofness. Similarly, we can get the contradiction when increasing b_{e_j} to $b^{'}_{e_j}$ can decrease the payment to edge $e \in P$.

Similarly, we can prove that for an edge e_j on the winning path P, decreasing b_{e_j} cannot change either the winning path or the payment to any edge in the graph.

When we have bidding vectors $b(r_p)$ and $b(r_q)$, we can construct a new bidding vector b^* such that $b^*_{e_j} = \min\{b(r_p)_{e_j}, b(r_q)_{e_j}\}$ if e_j is on the wining path P while $b^*_{e_j} = \max\{b(r_p)_{e_j}, b(r_q)_{e_j}\}$ if e_j is not on the wining path. According to the construction and the above arguments, we can get $\forall e_j$, $p_{e_j}(b(r_p)) = p_{e_j}(b^*) = p_{e_j}(b(r_q))$. This contradicts to the fact that any edge e_j on the winning path P, $p_{e_j}(b(r_p)) < b(r_q)_{e_j} \le p_{e_j}(b(r_q))$.

Therefore, given the assumption that the losing agent always has 0 payoff, there is no strategyproof mechanism for auction of type I that satisfies individual rationality. □

Proposition 2. *In the extended model of path auction game, there is no false-name proof mechanism for auction of type II that satisfies Pareto efficiency.*

Proof Sketch: We are going to prove this proposition by presenting a generic counter example as [13] assuming there is an efficient false-name-proof mechanism. The generic counter example is given in Figure 2. In this example, edges $e(s,r), e(r,t)_1$ are owned by agent 1, edge $e(s,t)$ is owned by agent 2 while edge $e(r,t)_3$ is owned by agent 3. Since agent 1 owns two edges, when bidding in auction of type II, the ownership of edge $e(r,t)_1$ can be claimed as agent 1 or some artificial non existent agent 4. Given this example, the proof of this proposition is almost the same as the proof of Proposition 1 in [13]. □

Theorem 2. *If the underlying network is a parallel-path graph, the first-price path auction has an ϵ-Nash equilibrium.*

Proof Sketch: The ϵ-Nash equilibrium bidding vector is constructed as follows. Suppose the parallel-path graph is $(V,E,s,t) = \bigcup_k P^k(s,t)$, where $P^k(s,t)$ is the kth path from s to t and $\forall i \neq j$, $P^j(s,t) \bigcap P^j(s,t) = \varnothing$. Initially, each agent bids its true cost *i.e.* $b = c$. Let $W_b(P^k(s,t)) = \sum_{e \in P^k(s,t)} b_e$ denote the cost of path $P^k(s,t)$ with respect to the bidding vector b. Moreover, we assume that agent A_k has the highest value of $LPI(A_k)$ which is defined before. In order to find the ϵ-Nash equilibrium bidding vector, first we would pick one edge in $E_{A_k} \bigcap P^1(s,t)$ and increase its bidding price until $W_{b'}(P^1(s,t)) = W_b(P^{LPI(A_k)}(s,t)) - \epsilon$, where b' is the new bidding vector. For any path $j \in [2 ... LPI(A_k)-1]$, we pick one edge in $E_{A_k} \bigcap P^j(s,t)$ and increase its bidding price until $W_{b'}(P^j(s,t)) = W_b(P^{LPI(A_k)}(s,t))$. We call the final bidding vector b^f.

It is not hard to verify that b^f is an ϵ-Nash equilibrium bidding vector. □

Proposition 3. *Given the assumption that each edge's bidding price is at least its true cost, i.e. $\forall e \in E, b_e \ge c_e$, the graph showed in Figure 3 can not have a pure-strategy ϵ-Nash equilibrium in first-price auction.*

Proof: In Figure 3, the numbers in the brackets represent the identities of agents that owns the edges. There are 5 agents in this game and 5 paths from s to t:

Path 1: (s, p_1, p_2, p_3, t)

Path 2: $(s, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, t)$

Path 3: $(s, p_4, p_5, p_6, p_{12}, t)$

Path 4: $(s, p_{11}, p_8, p_9, p_{10}, t)$

Path 5: $(s, p_{11}, p_8, p_6, p_7, p_6, p_{12}, t)$

Let b be the ϵ-Nash equilibrium bidding vector. We claim that the cost of each path respect to b can differ at most by ϵ. We prove this by contradiction. Suppose P^k is the winning path and $\exists j, W_b(P^j) > W_b(P^k) + \epsilon$. For any agent $i \in [1...5]$ in Figure 3, there is only one path that does not have edges owned by i. We can assume that path P^j does not have edges owned by the agent i, but for all other 4 paths, agent i owns edges on all of them. Thus agent i can increase the bidding prices of its edges(but still keep P^k as the winning path) such that its payoff can increase by at least ϵ. Then contradiction occurs. Therefore, if b is an ϵ-Nash equilibrium bidding vector, the cost of each path respect to b can differ at most by ϵ. Based on this fact, we can get the following two equations:

$|(b_{p_8,p_7} + b_{p_7,p_6} + b_{p_6,p_{12}} + b_{p_{12},t}) - (b_{p_8,p_9} + b_{p_9,p_{10}} + b_{p_{10},t})| \le \epsilon ...(1)$

$|(b_{p_6,p_7} + b_{p_7,p_8} + b_{p_8,p_9} + b_{p_9,p_{10}} + b_{p_{10},t}) - (b_{p_6,p_{12}} + b_{p_{12},t})| \le \epsilon ...(2)$

According to equations (1) and (2), we can get:

$(b_{p_7,p_8} + b_{p_6,p_7}) + b_{p_8,p_9} + b_{p_9,p_{10}} + b_{p_{10},t} - \epsilon$

$\le -(b_{p_7,p_6} + b_{p_8,p_7}) + b_{p_8,p_9} + b_{p_9,p_{10}} + b_{p_{10},t} + \epsilon$

Therefore, the following inequality holds:

$b_{p_7,p_6} + b_{p_8,p_7} + b_{p_6,p_7} + b_{p_7,p_8} \le 2\epsilon$

Moreover, according to our assumption $\forall e, c_e \le b_e$, the following inequality holds

$c_{p_7,p_6} + c_{p_8,p_7} + c_{p_6,p_7} + c_{p_7,p_8} \le 2\epsilon$

When ϵ is small enough, but the true cost of each edge is large enough, contradiction occurs. Therefore, there is no ϵ-Nash equilibrium of first-price auction in Figure 3. □

Bootstrapping the Long Tail in Peer to Peer Systems

Bernardo A. Huberman and Fang Wu
HP Labs, Palo Alto, CA 94304

May 4, 2006

Abstract

We describe an efficient incentive mechanism for P2P systems that generates a wide diversity of content offerings while responding adaptively to customer demand. Files are served and paid for through a parimutuel market similar to that used for betting in horse races and in lotteries. An analysis of the performance of such a system shows that there exists an equilibrium with a long tail in the distribution of content offerings, which guarantees the real time provision of any content regardless of its popularity.

1 Introduction

The provision of digitized content on-demand to millions of users presents a formidable challenge. With an ever increasing number of fixed and mobile devices with video capabilities, and a growing consumer base with different preferences, there is a need for a scalable and adaptive way of delivering a diverse set of files in real time to a worldwide consumer base.

Providing such varied content presents two problems. First, files should be accessible in such a way that the constraints posed by bandwidth and the diversity of demand is met without having to resort to client server architectures and specialized network protocols. Second, as new content is created, the system ought to be able to swiftly respond to new demand on specific content, regardless of its popularity. This is a hard constraint on any distributed system, since providers with a finite amount of memory and bandwidth will tend to offer the most popular content, as is the case today with many peer-to-peer systems.

The first problem is naturally solved by peer to peer networks, where each peer can be both a consumer and provider of the service. Peer to peer networks, unlike client server architectures, automatically scale in size as demand fluctuates, as well as being able to adapt to system failures. Examples of such systems are Bittorrent [4] and Kazaa, who account for a sizable percentage of all the use of the Internet. Furthermore, new services like the BBC IMP, (`http://www.bbc.co.uk/imp/`) show that it is possible to make media content available through a peer-to-peer system while respecting digital rights.

It is the second problem, that of an adaptable and efficient system capable of delivering any file, regardless of its popularity, that we now solve. We do so by creating an implementable incentive mechanism that ensures the existence of a diverse set of offerings which is in equilibrium with the available supply and demand, regardless of content and size. Moreover, the mechanism is such that it automatically generates the long tail of offerings which has been shown to be responsible for the success of a number of online businesses such as Amazon or eBay [2]. In other words, while the system delivers favorite mainstream content, it can also provide files that constitute small niche markets which only in the aggregate can generate large revenues.

In what follows we describe an efficient incentive mechanism for P2P systems that generates a wide diversity of content offerings while responding adaptively to customer demand. Files are served and paid for through a parimutuel market similar to that commonly used for betting in horse races. An analysis of the performance of such a system shows that there exists an equilibrium with a long tail in the distribution of content offerings, which guarantees the real time provision of any content regardless of its popularity. In our case, the bandwidth fraction of a given file offered by a server plays the role of the odds, the bandwidth consumed corresponds to bettors, the files to horses, and the requests are analogous to races.

An interesting consequence of this mechanism is that it solves in complete fashion the free riding problem that originally plagued P2P systems like Gnutella [1] and that in milder forms still appears in other such systems. The reason being that it transforms the provision of content from a public good into a private one.

We then analyze the performance of such a system by making a set of assumptions that are first restrictive and are then relaxed so as to make them correspond to a realistic crowd of users. We show that in all these cases there exists an equilibrium in which the demand for any file can be fulfilled by the system. Moreover this equilibrium exhibits a robust empirical anomaly which is responsible for generating a very long tail in the distribution of content offerings. We finally discuss the scenario where most of the servers are bounded rational and show that it is still possible to achieve an optimum equilibrium. We conclude by summarizing our results and discussing the feasibility of its implementation.

2 The system and its incentive mechanism

Consider a network-based file exchange system consisting of three types of traders: content provider, server, and downloader or user. A content provider supplies—at a fixed price per file—a repertoire of files to a number of people acting as peers or servers. Servers then selectively serve a subset of those files to downloaders for a given price. In a peer-to-peer system a downloader can also, and often does, act as a server.

If the files are typically large in size, a server can only afford to store and serve a relatively small subset of files. It then faces the natural problem of choosing an optimal (from the point of view of maximizing his utility) subset of files to store so as to sell them to downloaders.

Suppose that the system charges each downloader a flat fee for downloading any one file (as in Apple's iTunes music store), which we normalize to one. Since many servers can help distribute a single file, this unit of income has to be allocated to the servers in ways that will incentivize them to always respond to a changing demand.

In order to do so, consider the case where there are m servers and n files. Let b_{ij} be the effective bandwidth of server i serving file j, normalized to $\sum_{i,j} b_{ij} = 1$. Also, denote the bandwidth fraction of file j by $\pi_j = \sum_k b_{kj}$.

Suppose that when a downloader connects to the system, it starts downloading different parts of the file simultaneously from all available servers that have it. When it finishes downloading, it will have received a fraction of the file j

$$q_{ij} = \frac{b_{ij}}{\sum_k b_{kj}} = \frac{b_{ij}}{\pi_j} \qquad (1)$$

from server i. Our mechanism prescribes that *the system should should pay an amount q_{ij} to server i as its reward for serving file j.*

Now consider the case when server i's reserves an amount of bandwidth b_{ij} as his "bid" on file j. Because we have normalized the total bandwidth and the total reward for serving one request both to one, the proportional share allocation scheme described by Eq. (1) can be interpreted as redistributing the total bid to the "winners", in proportion to their bids. Thus our payoff structure is similar to that of a pari-mutuel horse race betting market, where the π_j can be regarded as the odds, the bandwidth corresponds to bettors, the files to horses, and the requests are analogous to races. It is worth pointing out however, that in a real horse race all players who have placed a bet on the winning horse receive a share of the total prize, whereas in our system only those players that kept the "winning" file and also had a chance to serve it get paid. In spite of this difference it is easy to show that when rewritten in terms of expected payoffs, the two mechanisms behave in similar fashion.

3 The solution

3.1 Rational servers with static strategies and known download rates

In this section we make three simplifying assumptions. While not realistic they serve to set the framework that we will utilize later on to deal with more realistic scenarios. First, every server is rational in the sense that he chooses the optimal bandwidth allocation that maximizes his utility, whose explicit form will be given below. Second, every server's allocation strategy is static, i.e. the b_{ij}'s are independent of time. Third, we assume that each file j is requested randomly at a rate $\lambda_j > 0$ that does not change with time, and these rates are known to every server.

Consider a server i with the following standard additive form of utility:

$$U = \mathbb{E}\left[\int_0^\infty e^{-\delta t} u(t) dt\right], \tag{2}$$

where $u(t)$ is his income density at time t, and $\delta > 0$ is his future discount factor. Let X_{j1} be the (random) time that file j is requested for the first time, let X_{j2} be the time elapsed between the first request and the second request, and so on. According to our parimutuel reward scheme, server i receives a lump-sum reward b_{ij}/π_j from every such request, at times $X_{j1}, X_{j1} + X_{j2}$, etc. Thus, the server i's total utility is given by

$$U = \sum_j \frac{b_{ij}}{\pi_j} \sum_{l=1}^{\infty} \mathbb{E}[e^{-\delta \sum_{k=1}^{l} X_{jk}}] \equiv \sum_j \frac{b_{ij}}{\pi_j} u_j. \tag{3}$$

which amounts to each server receiving a utility proportional to the fraction of the file that he serves. Notice that the sum of expectations in Eq. (3) (denoted by u_j) can be calculated explicitly. Because the X_{jk}'s are i.i.d. random variables with density $\lambda_j^{-1} \exp(\lambda_j x)$, it can be calculated that $u_j = \lambda_j/\delta$. If we let $\lambda = \sum_j \lambda_j$ be the total request rate and $p_j = \lambda_j/\lambda$ be the probability that the next request asks for file j, then we can also write $u_j = \lambda p_j/\delta$. Plugging this back into Eq. (3), we obtain

$$U = \frac{\lambda}{\delta} \sum_j \frac{p_j b_{ij}}{\pi_j}. \tag{4}$$

Since we assume that server i is rational, he will allocate b_{ij} in a way that it solves the following optimization problem:

$$\max_{(b_{ij})_{j=1}^n \in \mathbb{R}_+^n} \sum_j \frac{p_j b_{ij}}{\sum_k b_{kj}} \quad \text{subject to} \quad \sum_j b_{ij} \le b_i. \tag{5}$$

where b_i is the total upload bandwidth of user i. Thus we see that the servers are playing a *finite budget resource allocation game*. This type of game has been studied intensively, and a Nash equilibrium has been shown to exist under mild assumptions [6, 9]. In such an equilibrium, the players' utility functions are strongly competitive and in spite of a possibly large utility gap, the players behave in almost envy-free fashion, i.e. each player believes that no other player has received more than they have.

3.2 Rational servers with static strategies and unknown request rates

We now relax some of the assumptions made above so as to deal with a more realistic case.

It is usually hard to find out the accurate request rate for a given file, especially at the early stages when there is no historical data available. Thus it makes more sense to assume that every server i holds a *subjective belief* about those request rates. Let p_{ij} be server i's subjective probability that the next request is for file j. Then server i believes that file j will be requested at a rate $\lambda_{ij} = \lambda p_{ij}$. Eq. (5) then becomes

$$\max_{(b_{ij})_{j=1}^n \in \mathbb{R}_+^n} \sum_j \frac{p_{ij} b_{ij}}{\sum_k b_{kj}} \quad \text{subject to} \quad \sum_j b_{ij} \le b_i. \tag{6}$$

which is still a finite budget resource allocation game as considered in the previous section.

It is interesting to note that when m is large, b_{ij} is small compared to $\pi_j = \sum_k b_{kj}$, so that π_j can be treated as a constant. In this case, the optimization problem can be well approximated by

$$\max_{(b_{ij})_{j=1}^n \in \mathbb{R}_+^n} \sum_j \frac{p_{ij} b_{ij}}{\pi_j} \quad \text{subject to} \sum_j b_{ij} \le b_i. \quad (7)$$

Thus, user i should use all his bandwidth to serve those files j with the largest ratio p_{ij}/π_j.

This scenario (7) corresponds to the so-called *parimutuel consensus* problem, which has been studied in detail. In this problem a certain probability space is observed by a number of individuals, each of which endows it with their own subjective probability distributions. The issue then is how to aggregate those subjective probabilities in such a way that they represent a good consensus of the individual ones. The parimutuel consensus scheme is similar to that of betting on horses at a race, the final odds on a given horse being proportional to the amount bet on the horse. As shown by Eisenberg and Gale [5], an equilibrium then exists such that the bettors as a group maximize the weighted sum of logarithms of subjective expectations, with the weights being the total bet on each horse.

Moreover a number of empirical studies of parimutuel markets [7] have shown that they do indeed exhibit a high correlation between the subjective probabilities of the bettors and the objective probabilities generated by the racetracks. Equally interesting for our purposes is the existence of a robust empirical anomaly called *the favorite-longshot bias* [7]. The anomaly shows that favorites win less frequently than the subjectives probabilities imply, and longshots more often. This anomaly enhances the long tail, which is populated by those files which while not singly popular, in aggregate are responsible for a large amount of the traffic in the system.

3.3 Rational servers with a dynamic strategy

We now consider the case where the rate at which files are requested can change with time. Because of this, each server has to actively adjust its bandwidth allocation to adapt to such changes. As we have seen in the last section, user i has an incentive to serve those files with large values of p_{ij}/π_j. Recall that $\pi_j(t)$ is just the fraction of total bandwidth spent to serve file j at time t, which in principle can be estimated from the system's statistics. Thus it would be useful to have the system frequently broadcast the real-time π_j to all servers so as to help them decide on how to adjust their own allocations of bandwidth.

From Eq. (1) we see that, by serving file j, user i's expected per bandwidth earning from the next request is $p_j q_{ij}/b_{ij} = p_j/\pi_j$. Hence a user will benefit most by serving those files with the largest "p/π ratio". However, as soon as a given user starts serving file j, the corresponding p/π ratio decreases. As a consequence, the system self-adapts to the limit of uniform p/π ratios. If the system is perfectly efficient, we would expect that p_j/π_j = constant. Because p_j and π_j both sum up to one, this implies that $\pi_j = p_j$, or $\sum_k b_{kj} = \lambda_j/\lambda \propto \lambda_j$. In other words, the total bandwidth used to serve a file is proportional to the file's request rate.

This result has interesting implications when considering the social utility of the downloaders. Recently, Tewari and Kleinrock [8] have shown that in a homogeneous network the average download time is minimized when $\sum_k b_{kj} \propto \lambda_j$. This implies that in the perfectly efficient limit, our mechanism maximizes the downloaders' social utility, which is measured by their average download times.

Since in reality a market is never perfectly efficient, the above analysis only makes sense if the characteristic time it takes for the system to relax back to uniformity from any disturbance is short. As a concrete example, consider a new file j released at time 0, being shared by only one server. Suppose that every downloader starts sharing her piece of the file immediately after downloading it. Because there are few servers serving the file but many downloaders requesting the file, for very short times afterwards the upload bandwidth will be fully utilized. That is, during time dt, an amount $\pi_j(t)dt$ of data is downloaded and added to the total upload bandwidth immediately. Hence we have

$$d\pi_j(t) = \pi_j(t)dt. \quad (8)$$

which implies that $\pi_j(t)$ grows exponentially until

$\pi_j(T) \sim p_j$. Solving for T, we find

$$T \sim \log\left(\frac{p_j}{\pi_j(0)}\right). \tag{9}$$

Thus the system reaches uniformity in logarithmic time, a signature of its high efficiency.

3.4 Servers with bounded rationality

So far we have assumed that all servers are rational, so that they will actively seek those files that are most under-supplied so as to serve them to downloaders. In reality however, while some servers do behave rationally, a lot of others do not. This is because even a perfectly rational server sometimes can make wrong decisions as to which files to store because his subjective probability estimate of what is in demand can be inaccurate. Also, such a bounded-rational server can at times be too lazy to adjust his bandwidth allocation, so that he will keep serving whatever he has, and at other times he might simply imitate other servers' behavior by choosing to serve the popular files. In all these cases we need to consider whether or not the lack of full rationality will lead to equilibrium on the part of the system.

As a simple example, assume there are only two files, A and B. Let $p = \lambda_A/\lambda$ be file A's real request probability, and let $1-p$ be file B's real request probability. Suppose the servers are divided into two classes, with α fraction rational and $1-\alpha$ fraction irrational, arriving one by one in a random order. Each rational server's subjective probability in general can be described by an identically distributed random variable $P_t \in [0,1]$ with mean p. Then with probability $\mathbb{P}[P_t > \pi(t)]$ he will serve file A, and with probability $\mathbb{P}[P_t < \pi(t)]$ he will serve file B. In order to carry out some explicit calculation below, we consider the simplest choice of P_t, namely a Bernoulli variable

$$\mathbb{P}[P_t = 1] = p, \quad \mathbb{P}[P_t = 0] = 1-p. \tag{10}$$

(Clearly $\mathbb{E}[P_t] = p$, so the subjective probabilities are accurate on average.) It is easy to check that under this choice a rational server chooses A with probability p and B with probability $1-p$.

On the other hand, consider the situation where an irrational server chooses an existing server at random and copies that server's bandwidth allocation. That is, with probability $\pi(t)$ an irrational server will choose file A.[1]

From these two assumptions we see that

$$\mathbb{P}[\text{server } t \text{ serves A}] = \alpha p + (1-\alpha)\pi(t), \tag{11}$$

and

$$\mathbb{P}[\text{server } t \text{ serves B}] = \alpha(1-p) + (1-\alpha)(1-\pi(t)). \tag{12}$$

The stochastic process described by the above two equations has been recently studied in the context of choices among technologies for which evidence of their value is equivocal, inconclusive, or even nonexistent [3]. As was shown there, the dynamics generated by such equations leads to outcomes that appear to be deterministic in spite of being governed by a stochastic process. In the context of our problem this means that when the objective evidence for the choice of a particular file is very weak, any sample path of this process quickly settles down to a fraction of files downloaded that is not predetermined by the initial conditions: ex ante, every outcome is just as (un)likely as every other. Thus one cannot ensure an equilibrium that is both optimum and repeatable.

In the opposite case, when the objective evidence is strong, the process settles down to a value that is determined by the quality of the evidence. In both cases the proportion of files downloaded never settles into either zero or one.

In the general case that we have been considering, there are always a number of servers that will behave in bounded rational fashion and a few that are perfectly rational. Specifically, when $\alpha > 0$, which corresponds to the case where a small number of servers are rational, the $\pi(t)$ will converge to p in the long time limit. That is, a small fraction of rational servers is enough for the system to reach an optimum equilibrium. However, it is worth pointing out that since

[1] This assumption can also be interpreted as follows. Suppose a downloader starts serving his files immediately after downloading, but never initiates to serve a file. (This is the way a non-seed peer behaves within Bittorrent.) Then the probability that he will serve file j is exactly the probability that he just downloaded file j, which is $\pi_j(t)$.

the characteristic convergence time diverges exponentially in $1/\alpha$, the smaller the value of alpha α, the longer it will take for the system to reach such an optimum state.

4 Conclusion

In this paper we described a peer-to-peer system with an incentive mechanism that generates diversity of offerings, efficiency and adaptability to customer demand. This was accomplished by having a pricing structure for serving files that has the structure of a parimutuel market, similar to those commonly used in horse races, where the the bandwidth fraction of a given file offered by a server plays the role of the odds, the bandwidth corresponds to bettors, the files to horses, and the requests are analogous to races. Notice that this mechanism completely solves the free riding problem that originally plagued P2P systems like Gnutella and that in milder forms still appears in other such systems.

We then analyzed the performance of such a system by making a set of assumptions that are first restrictive but are then relaxed so as to make the system respond to a realistic crowd. We showed that in all these cases there exists an equilibrium in which the demand for any file can be fulfilled by the system. Moreover this equilibrium is known to exhibit a robust empirical anomaly, that of the *favorite-longshot bias*, which in our case generates a very long tail in the distribution of offerings. We finally discussed the scenario where most of the servers are bounded rational and showed that it is still possible to achieve an optimum equilibrium if a few servers can act rationally.

The implementation of mechanism is feasible with present technologies. The implementation of a prototype will also help study the behavior of both providers and users within the context of this parimutuel market. Given its feasibility, and with the addition of DRM and a payment system, it offers an interesting opportunity for the provision of legal content with a simple pricing structure that ensures that unusual content will always be available along with the more traditional fare.

Acknowledgements. We benefited from discussions with Eytan Adar, Tad Hogg and Li Zhang.

References

[1] Eytan Adar and Bernardo A. Huberman. Free Riding on Gnutella. First Monday October (2000).

[2] Chris Anderson. The long tail. `http://longtail.typepad.com/the_long_tail/` (2005).

[3] Jonathon Bendor, Bernardo A. Huberman and Fang Wu. Management fads, pedagogies and soft technologies. `http://www.hpl.hp.com/research/idl/papers/fads/fads.pdf` (2005).

[4] Bram Cohen. Incentives build robustness in Bittorrent. Working Paper, Workshop on the Economics of P2P Systems (2003).

[5] Edmund Eisenberg and David Gale. Consensus of subjective probabilities: The parimutuel method. Annals of mathematics statistics (1958).

[6] L. Shapley and M. Shubik. Trade using one commodity as a means of payment. Journal of Political Economy, Vol. 85:5, 937–968 (1977).

[7] Richard H. Thaler and William T. Ziemba, Parimutuel betting markets: racetracks and lotteries, Journal of Economic Perspectives, Vol. 2, No. 2, pp. 161–174 (1988).

[8] Saurabh Tewari and Leonard Kleinrock. On fairness, optimal download performance and proportional replication in peer-to-peer networks. Proceedings of IFIP Networking 2005, Waterloo, Canada (2005).

[9] Li Zhang. The efficiency and fairness of a fixed budget resource allocation game. ICALP (2005).

Incentive Based Ranking Mechanisms

Rajat Bhattacharjee [*]
Stanford University

Ashish Goel [†]
Stanford University

Position Paper

Abstract

We consider ranking and recommendation systems based on user feedback. We make a case for sharing the revenue generated by such systems with users as incentive to provide useful feedback. Our main contribution are mechanisms for ranking/recommendation which gives incentive for the users to provide useful feedback and is resistant to selfish/malicious behavior (click spam). The mechanisms are designed to give higher incentives for discovering high quality entities rather than for merely providing additional positive feedback for already established entities. A page whose rating/ranking is at variance with its real quality represents an arbitrage opportunity. The mechanisms are simple enough to be used with existing technology in ranking and recommendation systems, requiring little or no extra effort by the users.

1 Introduction

Before the advent of the Internet, content generation was channeled through a limited number of publishers, such as book publishers, movie production companies, music companies, newspapers, and magazines. In order to regulate and also advertise the quality of content, a system of content evaluation had evolved. Evaluation in traditional publishing is done primarily by professional reviewers and editors who are paid for their opinions. In contrast to self-publishing, the editors decide which content gets published in accordance with the quality of the content.

Content generation is no longer channeled through a limited number of publishers. Individuals self-publish their views, or articles, or creative pieces using websites, blogs, photograph hosting services, podcasts, etc. The scale and decentralization of the content in the Internet makes the old centralized mode of content evaluation impractical. At the same time this decentralization and the corresponding lack of editorial control at the source makes content evaluation all the more important. This need has played a strong role in the success of search engines like Google [14], Yahoo [16] and many others, which not only search but also rank content, thus playing the role of reviewers. In addition, recommendation systems use similarities in the feedback profiles of users and entities to recommend new items [1].

PageRank [22] uses the link structure of the Internet to rank webpages. The philosophy of this approach is that the quality of a webpage is indicated by the quality of the webpages pointing to it. However, interested parties have used it to promote the ranking of their own webpages, for example, by creating dummy webpages pointing to their own. As heuristics have been proposed and implemented to detect these malicious webpages, the techniques used by the search engine optimizers have also gotten better [10] [11]. Detecting these PageRank amplifying structures is equivalent to the sparsest cut problem [25], which is NP-hard [18].

An alternative to link-based methods such as PageRank [22] and Hits [17] is to use the feedback from users (e.g. clicks). This approach is already used in many recommendation systems [1]. The difficulty of using feedback/clicks stems from detecting whether the feedback/clicks are coming from genuine users who found the webpage useful or are coming from a single source, a phenomenon known as click spam. Various solutions have been

[*]Department of Computer Science. Email: rajatb@stanford.edu.

[†]Department of Management Science and Engineering and (by courtesy) Computer Science. Email: ashishg@stanford.edu.

proposed for this problem. However, these in turn have resulted in smarter techniques being used by the spammers [2][19][24].

The main contribution of this paper is two fold: (1) we make a case for sharing with users the revenue generated by such systems as incentive to provide useful feedback. (2) we present a preliminary design of mechanisms for ranking/recommendation systems which give incentive to the users to provide useful feedback. The mechanisms are designed to provide a higher incentive for discovering high quality entities rather than for providing more positive feedback for already established entities. In section 3, we motivate and list desirable properties of a ranking system. The proposed ranking mechanisms (section 4) are shown to possess these properties, in particular they are resistant to click spam. The mechanisms are simple enough to be used with existing technology in ranking and recommendation systems. We begin by giving a generic model for ranking and recommendation systems.

2 A Generic Model for Ranking Systems

In the introduction, we mention the problems of using PageRank for ranking. Here we consider ranking systems which are based on user feedback. A typical ranking system has the following features (similar ideas apply to recommendation systems based on user feedback as well):

1. **Entities.** The set of entities which we wish to rank is denoted by $\mathcal{E}$. We denote the ith entity by e_i. Each entity has an inherent quality denoted by q_i which is *not* known. However, if two entities with qualities q_i and q_j are presented to users with $q_i > q_j$, then more users would find entity i better than entity j.

2. **Users.** The set of users in the system is denoted by $\mathcal{U}$. These users provide feedback on the entities. We denote the ith user in the system by u_i. Note that we implicitly assume that these users are registered with the system. The users are further classified as (this classification is inspired by the well known difficulty of eliciting useful feedback from users [8][23]):

   (a) **Sheep.** The label sheep corresponds to the user who leaves feedback for an entity when the entity is shown to him/her (recommended or ranked highly). A high quality entity which is not shown to a sheep would not get any feedback from that sheep.

   (b) **Connoisseur.** A connoisseur is a user who would find a good quality entity even when it is not shown to him/her. We assume that the ratio of connoisseurs to sheep in the system is ϵ. Typically, we expect ϵ to be small. In the context of web search, a connoisseur would be a user who wouldn't merely depend upon search engine ranking and would use more specific keywords or otherwise targeted searches to find the information he is looking for. In the context of news articles, a connoisseur would be a user who is really interested in a particular topic and would look out for any interesting news article on this topic. Note that the same user can be a connoisseur for a certain topic and a sheep for another.

3. **Feedback.** The notion of feedback is captured by tokens. When a user gives positive feedback for an entity i, the number of tokens placed on the entity, denoted by τ_i, is incremented by 1. We represent the relative number of tokens an entity attracts, $\frac{\tau_i}{\sum_j \tau_j}$, by r_i.

4. **Revenue/Utility.** We identify the revenue/utility generated by an entity i with the rate at which the sheep leave positive feedback for that entity. The rate at which revenue is generated by entity i is given by the revenue function, $f(r_i, q_i)$. In general, the revenue function is *not* known but we do know the revenue generation event for each entity. The function is assumed to be non-decreasing in r_i and increasing in q_i. In other words, if two entities have the same share of tokens but the quality of first entity is better than the second, then the first entity generates more revenue than the second. Similarly, if two entities have the same quality but one has greater share of tokens, then the revenue generated by the first is more than the second. Implicit in these assumptions is the fact that the revenue function is a good indicator of the utility that an entity generates for the system. For example, this might not be the case in the pay-per-click model for ad-funded ranking systems, however, it would be a good indicator in the pay-per-acquisition

model. The revenue/utility can be of three kinds: (1) recommendations can be directly related to the purchase of goods, e.g. in the case of e-merchants like Amazon.com [13], thus better recommendations would lead to better revenues, (2) rankings and recommendations would lead to increased consumer satisfaction, thus attracting more users and the number of users in the system is directly related to the revenue generates, e.g. in the case of service providers who charge users for membership such as Netflix [15], and (3) the relationship with actual revenue generation can be more abstract like in the case of ad-funded search engines such as Google [14], where consumer satisfaction increases the number of users, which is then translated into revenue through ads.

In most cases we have some knowledge of function f. For example, an interesting case is when $f(r,q) = rq$, which arises when the probability that a webpage gets viewed is given by r and the conditional probability that the page gets clicked is given by q (this function also arises in related settings [21][4]). We define a general class of functions which includes the above function. A function f is said to be a *separable function* if $f(r,q) = qr^{\alpha}$, for some $\alpha > 0$.

3 Desiderata

In this section, we motivate and list properties that a ranking system should have. In section 4.1.3, we formalize these properties.

1. A ranking system should result in a ranking which is in accordance with the quality of the entities. More precisely, if for two entities i and j, $q_i > q_j$, then the ranking of the entities should be such that entity i is positioned before entity j. We call this property *ranking by quality.*

2. Groups associated with a particular entity might have interest in promoting its rank irrespective of its quality. For example, the owner of a hotel in Bali would like the webpage of his hotel to be one of the first few webpages which show up when a user searches for Bali. Also some groups might be interested in demoting the rank of a certain entity. For example, the owner of a rival hotel might try to lower the ranking of the webpage of the other hotel. Thus, a ranking system should be resistant to such selfish/malicious behavior. We call this property *resistance to gaming.*

3. Imagine two entities of similar quality (one can think of two news providing webpages) with huge resources. The number of users these two entities attract would then depend upon their relative ranking. An item which is slightly lower in ranking might succeed in improving its ranking by using the knowledge of how the ranking system works. For example, in case of PageRank, the entity would try to make sure that more webpages point to it. In case the ranking system uses click through analysis, the entity might try to fraudulently generate more clicks. In response the rival entity might indulge in similar practices to restore the relative ranking. This cycle can repeat endlessly making the ranking system unstable. A good ranking system should not foster such behavior. We call this property *resistance to racing.*

3.1 Case for Incentives

We believe that incentives are necessary for the proper functioning of a ranking system based on user feedback. There are three main reasons for our position. (1) The difficulty of eliciting useful feedback from users is well known [8][23]. In a similar vein, it has been shown that search engine results influence the popularity of webpages [5][6]. (2) The feedback profile of an entity plays an important role in attracting future users. This gives a strong incentive for groups associated with the entity to leave fraudulent positive feedback for it. In the context of reputation systems, this phenomenon is known as ballot stuffing and bad mouthing [3] [7]. In the context of webpage ranking, this phenomenon has been studied in the literature under the name of click spam [2][24]. We believe that solutions proposed to solve this problem would lead to a heuristic race in the lines of PageRank. (3) The problem of new users in a system has been studied in the reputation systems [9]. Similar phenomenon may occur in ranking systems as well. New entities can be added (new webpages are created all the time). Or, there might be a sudden change in the relevance of an entity. For example, articles on a certain individual might suddenly become very relevant when he/she is nominated for some important post, or articles on a certain stock might suddenly be sought after a surprise declaration of strong earnings. Even if technology could be developed for combating click spam, it can be shown that for small values of ϵ (the fraction of connoisseurs in the system), an entity would take an impractically large amount of time to attain a position in the ranking which is in accordance with its quality. Please see Appendix for a more detailed analysis.

4 Incentive Based Mechanisms for Ranking

We first present the mechanisms for ranking systems based on incentives. We then show that the ranking system has the properties we listed in section 3 (and which we also formalize in section 4.1.3), in particular, it is resistant to click spam.

4.1 Ranking Mechanisms

Users are allowed to place positive and negative tokens on various entities subject to some constraints. The ranking of entities is updated based on the knowledge of the number of tokens placed on the entities. The ranking results in revenue generation events. At each such event, a part of the revenue is shared with the users. Central to the ranking mechanism are the notions of tokens and incentives. We first formalize the notion of tokens and then describe the incentives. Finally, we describe the mechanisms for placing tokens.

4.1.1 Tokens

A token T_i is a five tuple

$$\{p_i, u_i, e_i, w_i, t_i\}.$$

The value $p_i \in \{+1, -1\}$ specifies whether the token is a positive/negative token. A value of $+1$ indicates that T_i is a positive token and a value of -1 indicates that the token is a negative one. The user who placed the token is determined by $u_i \in \mathcal{U}$ and $e_i \in \mathcal{E}$ determines the entity on which the token is placed. The weight of a token (chosen by the user while placing the token) is given by $w_i \in \mathcal{R}^+$ and the time at which the token is placed is given by $t_i \in \mathcal{R}^+$. The order of arrival of tokens is given by the subscript i. We assume that no two tokens arrive at the same time. The only constraint is that at any given time the net positive tokens of a user is bounded by γ which is a system parameter. Note that a user can obtain more positive tokens (for potental future placement) by placing negative tokens.

4.1.2 Incentives

In this section, we describe how revenue is shared among the users. Suppose a revenue generation event occurs for an entity e at time t, and results in R amount of revenue being generated for the system. The mechanism has two parameters pertaining to incentives, β and s. The fraction of revenue to be distributed as incentive among the users is determined by $\beta \leq 1$. The parameter $s > 1$ controls the relative importance of tokens placed on an item depending on the order in which they were placed.

Let $\mathcal{T}$ be the set of all the tokens in the system. For a given token T_i, such that $e_i = e$, and a time period t we define $a_i(t)$ and $b_i(t)$ as follows (informally $a_i(t)$ is the weight of tokens on entity e_i which were placed before T_i and $b_i(t) - a_i(t)$ is the weight of token T_i at time t, note that $a_0(t) = 1$):

$$a_i(t) = \sum_{T_j \in \mathcal{T} : j < i, e_j = e} p_j w_j + 1,$$

$$b_i(t) = p_i w_i + a_i(t). \tag{1}$$

In case the above values fall below 1 for an entity, it is removed from the system for some pre-defined time. The revenue share of user u_i during time period t due to token T_i is given by

$$s\beta R \int_{a_i(t)}^{b_i(t)} \frac{1}{\tau^s} d\tau. \tag{2}$$

Note that the above quantity is positive or negative depending on p_i. We emphasize the following two properties: (1) the relative importance of the tokens placed earlier (discoveries of high quality entities) can be controlled by s, (2) the tokens placed after token T_i have no bearing on the incentives generated by T_i (contrast it with the case where s is allowed to be equal to 1). We note that the proposed mechanisms can be implemented in the existing systems where there are ways of giving explicit or implicit positive and negative feedback.

4.1.3 Properties of the System

In the following, we assume users behave rationally. In particular, we assume that if users see an arbitrage opportunity then the opportunity will be availed. Under this assumption, in our setting the desired properties listed in section 3 can be formalized as follows.

1. **Ranking by quality.** For every pair of entities (i, j) such that $q_i < q_j$ and $\tau_i > \tau_j$, there should exist a profitable arbitrage opportunity in the form of removing a token from entity i and placing it on entity j. We will now demonstrate that our mechanism satisfies the properties listed in the deseridata, when the revenue function $f(r, q)$ is a separable function (see section 2 for definition). Recall that this class contains the important function, $f(r, q) = rq$.

   Suppose there exists a pair of entities (i, j) such that $q_i < q_j$ and $\tau_i > \tau_j$, where τ_i and τ_j are the respective number of tokens taking into account the weights in equation 1. Let $f(r, q) = qr^{\alpha}$. Since $f(r, q)$ is an increasing function of q, $f(r_i, q_i)/\tau_i^s < f(r_i, q_j)/\tau_i^s$. We set the parameter s to an arbitrary value greater than α. Now $f(r_i, q_j)/\tau_i^s = q_j r_i^{\alpha}/\tau_i^s = \frac{q_j \tau_i^{\alpha}}{\tau_i^s (\sum_k \tau_k)^{\alpha}} < \frac{q_j \tau_j^{\alpha}}{\tau_j^s (\sum_k \tau_k)^{\alpha}} = f(r_j, q_j)/\tau_j^s$. The last inequality follows from the fact that $\tau_i > \tau_j$ and $s > \alpha$. Note that if $f(r_i, q_i)/\tau_i^s < f(r_j, q_j)/\tau_j^s$, then users can perform arbitrage by placing a negative token on entity i and a positive token on entity j. Hence the system ranks entities according to quality.

2. **Resistance to gaming.** In the setting of our mechanisms, the definition of resistance to gaming is identical to the definition of ranking by quality (the above mentioned inconsistency in the ranking can be a result of malicious behavior).

3. **Resistance to racing.** The system is said to be resistant to racing if two users A and B cannot indefinitely repeat actions a_A and a_B, respectively, where a_B undoes the effects of a_A and vice versa. Let acc_i be the current account of user i. This value acc_i is the amount that the user i has generated as incentives from the past. The user can cash all or part of this amount at any point (acc_i gets reduced by the amount cashed). However, the user cannot pay the system to get a larger acc_i. In case the value of the account of a user goes negative, the feedback of the user is not taken into consideration for a predefined time (the older tokens placed by the user are removed by setting the w_i's of corresponding tokens to 0). Note because of the bound on the number of positive tokens, two users cannot keep adding positive tokens to their chosen entities ad infinitum. Also they cannot continuously keep placing positive token on their chosen entity and negative token on their rival's entity, as one of these actions would have a net negative value and theireventually one of them would get bankrupt[1].

The system allows for addition of other features, for example, the tokens can be made to decay at a rate $d(t)$. The decay function ensures that a misstep of a user (that is, placing an erroneous negative token) is not recurrently penalized. Also, in many ranking and recommendation systems, we have greater leverage in controlling f. Suppose there are n entities that need to be recommended and we knew the number of eyeballs that an entity in the ith position would attract. Then the scheme we use to convert the r_i's to a (possibly probabilistic) ranking would decide the revenue function f. Since the choice of the scheme is in the hands of the designer, so is the revenue function f. In general, an appropriate model for users' response to a ranking system would help one to design better ranking schemes for that system.

4.2 Comparison to Information Markets

An alternative way of thinking about the problem is to characterize it as an information aggregation problem. Information markets have been successfully used for this purpose. So can we use information markets for webpage ranking? An information market approach can be implemented by floating shares of an entity and allowing users to trade them. (Note that this would require a separate system for trading and explicit participation of the users.) It is reasonable to assume that the part of revenue that one would share with the information market would only be a fraction of the actual revenue generated by the webpage. Hence, the owner of the webpage would

[1]A more explicit approach to avoid "racing" is to multiply any negative revenue (i.e. when p_i is negative in equation 1, resulting in a negative share from equation 2) by $(1 + \delta)$ where δ is an arbitrarily small positive number. Now even if two players are "racing" on pages which have the same quality and the same number of tokens, one of them will go bankrupt quite quickly. And the property of resistance to gaming will be affected only marginally.

have an incentive to buy all the shares, thus creating a thin market in which the owner by the act of hoarding the shares succeeds in taking away the arbitrage opportunities of other users, thus artificially increasing the price of the webpage. Market scoring rules [12] solve the thin market problem. Our scheme might be seen as an adaptation of market scoring rules. However, there are important features in our setting which makes the direct use of market scoring rules infeasible. Unlike other information aggregation problems, the outcome of a ranking system is not divorced from the machinations of the market. In fact here the market is not merely a predictor of an event but plays an indispensable role in content distribution. Miller et al. [20] counter the lack of objective outcomes by comparing a user's reviews to that of its peers. Their scheme gives users an incentive to provide honest feedback. However, their approach doesn't address malicious users and the discovery of good entities which haven't attracted much feedback yet. Also, in their model, the impact of reviews on the outcome (for example, on the revenue generated by the system) is not explicit.

5 Future Directions

As pointed out in section 4.1.3, an appropriate model for users' response to a ranking system would help in designing better systems. In our view, modeling users' response to ranking/recommendation systems in specific domains is an important direction for future work. Another direction is designing ranking schemes with r_i's as input. While the appropriateness of the models would have a strong bearing on the design and success of these schemes, it is also possible that there are ranking algorithms and revenue sharing schemes which can be shown to work for a generic class of models of user behavior. One example would be the class of models where the number of eyeballs that a position attracts is fixed but unknown and the probability that an eyeball is converted to a useful event is a function of the quality of the entity present there. The separable revenue functions studied in this paper are a step in that direction.

References

[1] G. Adomavicius, A. Tuzhilin. Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No. 6, June 2005.*

[2] V. Anupam, A. Mayer, K. Nissin, B. Pinkas, M. Reiter. On the security of pay-per-click and other web advertising schemes. In *Proceedings of the 8th International Conference on World Wide Web, 1091-1100, 1999.*

[3] R. Bhattacharjee, A. Goel. Avoiding ballot stuffing in eBay-like reputation systems. *Third workshop on economics of peer-to-peer systems, 2005.*

[4] G. Bianconi, A-L. Barbasi. Competition and multiscaling in evolving networks. Europhysics letters, 54(4), 436-442, 2001

[5] J. Cho, S. Roy. Impact of Web search engines on page popularity. In *Proceedings of the Thirteenth International WWW Conference, 2004.*

[6] J. Cho, S. Roy, R. E. Adams. Page quality: In search of an unbiased web ranking. *SIGMOD, 2005.*

[7] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *Proceedings of the second ACM Conference on Electronic Commerce, October 2000.*

[8] C. Dellarocas. The digitization of word-of-mouth: promise and challenges of online reputation systems. *Management Science, Oct 2003.*

[9] E. Friedman, P. Resnick. The social cost of cheap pseudonyms. *Journal of Economics and Management Strategy, 10(2):173-199. 2001.*

[10] Z. Gyongyi, H. Garcia-Molina. Link Spam Alliances. *31st International Conference on Very Large Data Bases (VLDB).*

[11] Z. Gyongyi, H. Garcia-Molina. Spam: It's not just for inboxes anymore. *IEEE Computer Magazine, 38:10, 28-34.*

[12] R. Hanson Combinatorial Information Market Design. *Information Systems Frontiers, 5:1, 107-119, 2003.*

[13] http://www.amazon.com

[14] http://www.google.com

[15] http://www.netflix.com

[16] http://www.yahoo.com

[17] J. M. Kleinberg. Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM, 46(5):604-632, 1999.*

[18] F. T. Leighton, S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM, 46(6):787-832, 1999.*

[19] A. Metwally, D. Agrawal, A. El Abbadi. Duplicate detection in click streams. In *Proceeding of the 14th International Conference on World Wide Web, 12-21, 2005.*

[20] N. Miller, P. Resnick, R. Zeckhauser. Eliciting informative feedback: the peer-prediction method. *Management Science 51(9), 2005.*

[21] R. Motwani, Y. Xu. Evolution of page popularity under random web graph models. *Principles of Databases Systems, 2006*

[22] L. Page, S. Brin, R. Motwani, T. Winograd. The PageRank Citation Ranking: Bringing Order to the Web. *Stanford Digital Library Technologies Project, 1998.*

[23] P. Resnick, R. Zeckhauser, E. Friedman, K. Kuwabara. Reputation systems. *Communications of the ACM, 43(12):45-48, December 2000.*

[24] D. Vise. Clicking to steal. *Washington Post Magazine, F01, April 17 2005.*

[25] H. Zhang, A. Goel, R. Govindan, K. Mason, B. Van Roy. Making eigenvector-based reputation systems robust to collusion. *Workshop on Algorithms and Models for the Web Graph (WAW) 2004.*

Appendix A

In this appendix, we demonstrate in a simple setting, the need of incentives for a ranking/recommendation system to work properly. We emphasize that the problems we point to are not due to fraudulent clicks and hence cannot be fixed by better technology for detecting click spam. We show that an initial imbalance in the feedback would take exponential time to be corrected. Suppose there are two entities, e_1 and e_2, of the same quality q with e_1 having 1 token and e_2 having $\gamma > 1$ tokens. This difference can be due to various reasons. It can be a result of some targeted feedback by an interested party. Or, there might be a sudden change in the relevance of an entity. For example, articles on a certain individual might suddenly become very relevant when he/she is nominated for some important post, or articles on a certain stock might suddenly be sought after a surprise declaration of strong earnings. Also, the difference may be a result of the fact that the first entity is a new one in the system.

Since the quality of the two entities are the same, we can ignore the dependence of f on q_i. For the purpose of exposition, we assume that f depends linearly on r_i, that is, $f(r_i) = r_i = \frac{\tau_i}{\sum_j \tau_j}$. Similar results can be shown for other functions. Let the number of users in the system be $(1+\epsilon)n$ where there are n sheep and ϵn connoisseurs. Let τ_1 be the number of tokens on e_1 and τ_2 be the number of tokens on e_2. We normalize the number of users in the system and assume the ratio of sheep to connoisseur is $1 : \epsilon$. Now the rate at which sheep would put tokens on e_1 is given by $\frac{\tau_1}{\tau_1+\tau_2}$. Similarly the rate at which tokens are put on e_2 by sheep is given by $\frac{\tau_2}{\tau_1+\tau_2}$. Initially, $\tau_1 = 1$ and $\tau_2 = \gamma$. Since we are interested in proving a negative result, it doesn't harm us to assume that all the connoisseur weight ϵ is assigned to e_1. Hence,

$$\begin{aligned}
\frac{d\tau_1}{dt} &= \epsilon + \frac{\tau_1}{\tau_1+\tau_2}, \frac{d\tau_2}{dt} = \frac{\tau_2}{\tau_1+\tau_2} \\
\frac{d\tau_1}{dt} + \frac{d\tau_2}{dt} &= 1+\epsilon \qquad \text{[summing]} \\
\tau_1+\tau_2 &= 1+\gamma+(1+\epsilon)t \qquad \text{[integrating]} \\
\frac{d\tau_2}{dt} &= \frac{\tau_2}{(1+\gamma)+(1+\epsilon)t} \\
\log\frac{\tau_2}{\gamma} &= \frac{1}{1+\epsilon}\log\frac{(1+\gamma)+(1+\epsilon)t}{1+\gamma} \\
\tau_2 &= \gamma\left(1+\frac{1+\epsilon}{1+\gamma}t\right)^{\frac{1}{1+\epsilon}}
\end{aligned}$$

We are interested in the time when $\tau_1 = \tau_2$. Let the time when this equality is reached be T.

$$\begin{aligned}
2\gamma\left(1+\frac{1+\epsilon}{1+\gamma}T\right)^{\frac{1}{1+\epsilon}} &= 1+\gamma+(1+\epsilon)T \\
T &= \left[\left(\frac{2\gamma}{1+\gamma}\right)^{1+1/\epsilon} - 1\right]\left(\frac{1+\gamma}{1+\epsilon}\right)
\end{aligned}$$

For a large value of γ and $\epsilon = .01$, the time taken would be of the order of 2^{100} which is impractical. Note that even if we assume that we don't normalize and at every step the total change is of the order of n, this number would still be large for appropriate values of γ. It is easy to see that merely allowing negative feedback would make no qualitative difference in the above analysis.

Havelaar: A Robust and Efficient Reputation System for Active Peer-to-Peer Systems

Dominik Grolimund, Luzius Meisser, Stefan Schmid, Roger Wattenhofer
{grolimund@inf., meisserl@, schmiste@tik.ee., wattenhofer@tik.ee.}ethz.ch
Computer Engineering and Networks Laboratory (TIK), ETH Zurich, CH-8092 Zurich

Abstract— **Peer-to-peer (p2p) systems have the potential to harness huge amounts of resources. Unfortunately, however, it has been shown that most of today's p2p networks suffer from a large fraction of free-riders, who consume resources without contributing much to the system themselves. This results in an overall performance degradation, and hence proper incentives are needed to encourage contributions. One interesting resource is bandwidth. Thereby, a service differentiation approach seems appropriate, where peers contributing higher upload bandwidths are rewarded with higher download bandwidths in return. Keeping track of the contribution of each peer in an open, decentralized environment, however, is a difficult task; many proposed systems are susceptible to false reports. Besides being prone to attacks, some solutions have a large communication and computation overhead, which can even be linear in the number of transactions—an unacceptable burden in practical and active systems. In this paper, we propose a reputation system which is robust to false reports and overcomes this scaling problem. Our results are promising, indicating that the mechanism is accurate and efficient especially when applied in systems where there are lots of transactions.**

I. Introduction

The power of *peer-to-peer (p2p) computing* is based on the resource contribution of the network's constituent parts, the peers. Therefore, the success of a system in practice crucially depends on its ability to cope with selfish peers which aim at consuming more than they contribute.

When faced with the task of implementing a fairness scheme for upload bandwidth for our distributed p2p storage system *Kangoo*[1], we could not find a solution which entirely fits our needs. In Kangoo, *erasure codes* are employed to achieve high data availability with moderate redundancy. Files are divided into blocks, which are encoded into many small fragments. All these fragments are then stored on a different peer. Consequently, in Kangoo, there is a large number of transactions. Our goal was to develop a scheme for this environment, where peers contributing higher upload bandwidths for longer time periods are rewarded with a higher download bandwidth in return.

This paper presents the reputation system *Havelaar* which we have implemented for Kangoo. Unlike many existing solutions, Havelaar does not rely on transitivity of trust, and achieves a high robustness to attacks by design. This is accomplished by a novel aggregation technique in which a peer u always reports directly observed or aggregated contributions to the same set of peers. These *successor* peers are determined by a hash function $h(u)$ on the identifier (e.g., the IP-address) of u. This scheme allows a successor to detect and defend against egoistic cheating (e.g., reporting too large values, or reporting too often). Hence, our solution is different from *distributed hash table (DHT)-based* approaches where a peer u benefitting from a peer v reports v's contribution value to peers determined by a hash function $h(v)$ (i.e., depending on v rather than u).

Our results are promising: Havelaar is not only robust to attacks, but also efficient and—unlike many other solutions—scales well in the number of transactions. Therefore, we believe that Havelaar is well-suited for other active p2p systems with many transactions.

The rest of this paper is organized as follows. In Section II, related work is reviewed. Section III gives background information on Kangoo and on the intended environment for Havelaar. Section IV quickly outlines how peers could be rewarded given their contribution values. The Havelaar reputation system is then presented in detail in Section V. In Section VI, the accuracy of Havelaar's approximation of the real contributions is analyzed. We report on our results concerning communication costs in Section VII. Section VIII shows how Havelaar copes with various attacks. Finally, Section IX presents some simulation results, before Section X concludes the paper.

II. Related Work

It is not hard to find evidence of selfish behavior in existing p2p systems [2], [14], and the field has already spurred a large body of research [8], [12], [18], [20], [28].

Perhaps the simplest fairness mechanism is to directly incorporate contribution monitoring into the *client software*. For instance, in the popular file-sharing system *Kazaa*, the client records the contribution of its user. However, such a solution can simply be bypassed by implementing a different client which hard-wires the contribution level of the user to the maximum, as it was the case with *Kazaa Lite*.

In systems such as *BitTorrent* [6], where peers upload to the same set of peers from which they also download, a simple *tit-for-tat mechanism* [3] may be fine. When interactions between the same pairs of peers are less frequent, however, such *barter systems* [32] fail.

Inspired by real economies, some researchers have also proposed the introduction of some form of virtual money which is used for the transactions. However, these *monetary or credit based approaches* have a substantial overhead in terms of communication costs and infrastructure, and are inefficient [11], [33]. Often these systems also require market regulating mechanisms [31] to cope with inflation or deflation—a complex issue.

[1]To be released (http://www.caleido.com/kangoo).

Additionally, monetary based systems may deter users from participating [21].

If a peer makes too few direct observations to judge a peer's contribution, it has to take into account indirect observations from other peers [17]. Such systems are generally called *reputation* (or *reciprocity based*) systems, and are well-known from auctioning applications such as eBay. However, second-hand observations introduce the problem of *false reports* [4], [17]. Many proposals seek to mitigate these effects [1], [5], [9], [15], most of them relying on *transitivity of trust*, where observations are weighted by the reputation of the reporter. Additionally to the problem of false reports, an infrastructure to exchange the second-hand observations is needed [17]. In most reputation-based systems, second-hand observations are either requested before a transaction from other peers [1], [4], or they are simply flooded throughout the system. Alternatively, the contribution values can be stored in a *distributed hash table* (DHT) [25], [27], [30] ("DHT-based approach"). For systems with lots of transactions where contribution values are updated constantly, however, this results in an unacceptable communication overhead: updating and checking a peer's reputation entails costly (wide-area) DHT lookups [22], [23].

In contrast, in the Havelaar reputation system, a peer is able to compute the reputation of a requesting peer *locally*. By aggregating the contribution values of a large number of peers, our system tackles also the problem of transitivity of trust. Finally, a peer is able to defend against cheating by checking the *credibility of reports*, thus preventing many reputation attacks by design.

III. System Model

Havelaar was designed with a special application in mind: *Kangoo*, a large-scale distributed storage system for the Internet. Kangoo divides files into blocks, which are encrypted and then encoded into redundant fragments using erasure codes. These encrypted fragments are stored on different peers in a DHT. However, in Kangoo, peers fall into different categories, and fragments are only stored on so-called *storage peers* which fulfill some minimal requirements such as having long uptimes (e.g., more than 6 hours a day). How storage peers are selected among the set of all available peers, and why rational peers also have incentives to become storage peers is beyond the scope of this paper. However, in the following, we will assume that all peers store data, and that these peers generally have long uptimes.

Because of the used encoding scheme, storing and retrieving files in Kangoo results in lots of transactions with many different peers in the system (e.g., 500 transactions with different peers to store and retrieve *one* file). While this is necessary for high availability and for fast parallel downloads, it is crucial that the fairness mechanism scales well in the number of transactions—an important objective of Havelaar.

There are a number of other properties of Kangoo which influence Havelaar. Since fragments are stored in the Kangoo DHT depending on hash functions on the fragment itself, an attacker cannot determine the destination of a transaction. Furthermore, peer identifiers are securely assigned by Kangoo—i.e., externally to Havelaar—, and therefore Havelaar can rely on randomly assigned node identifiers and does not tackle Sybil attacks [7] or white-washing [9], [10] itself. However, Havelaar also minimizes incentives to create new identifiers by assigning new peers a low initial reputation.

Finally, in Kangoo, *churn* [16] is not a major concern, as (storage) peers are required to stay online on almost a daily basis and for several hours, and are also expected to remain in the system for longer time periods (months or years).[2]

IV. Rewarding Mechanism

Havelaar is mainly a *reputation system* (cf Section V) and therefore independent of a concrete rewarding mechanism; that is, given the contribution values of Havelaar, many strategies can be applied to allocate bandwidth to the peers.

However, to complete the picture, we briefly sketch the approach we have chosen for our system. We make use of a mechanism similar to the one described in [19]. But as performance is crucial in Kangoo, we only apply fairness mechanisms in situations of *contention*, i.e., when several peers want to download from the same node *concurrently*. Assuming that bandwidth is free—and lost when not used—, the maximum possible bandwidth will always be allocated to a requesting peer; no artificial limits are used. For our system, this is the desired behavior because—unlike monetary-based systems—we do not want to provide any disincentives to participate and download in the network. We only limit the resource allocation for excessive downloaders, as will be described in Section V.

V. Reputation System

In this section, we describe the main ideas behind Havelaar. The goal of Havelaar is to track the contribution of each peer in the system. Since it would be very expensive to inform each peer about *all* transactions happening in the system, we seek to provide the peers with a good *approximation* of the real contribution values. Thereby, our solution must be efficient and also resilient to cheating. Basically, Havelaar has three goals: (1) accurate estimation of the real (global) contribution values of other peers, (2) robustness against selfish peers, and (3) efficiency, i.e., scalability in the number of transactions.

In our system, the reputation of a peer u should be reflected by the peer's contribution value C_u, which in turn depends on the bandwidth b_u the peer provides, *and* the size s of the corresponding fragments. Hence, the total contribution value of peer u is given by $C_u = \sum_{\text{transactions } t} (b_{u,t} \cdot s_{u,t})$. Note that the contribution value will only be increased *after a complete upload* in order to reward proper transactions only.

So how does Havelaar track these contributions? Each peer u maintains a vector $\vec{o}$ (observations) of size n (number of peers in the network) to store the contributions of other peers which u has directly experienced itself. That is, after each download, u updates its vector $\vec{o}$ accordingly.

Even in active systems with lots of transactions, a peer only gets in touch with a subset of all peers. Therefore, peers need to share their private observations by sending them to other peers once in a while. To achieve this, Havelaar employs a *round-based aggregation technique*. Thereby, once in a round, each

[2]Note that in Kangoo, peers represent long-term customers.

peer u sends its observation vector $\vec{o}$ to a small number k (e.g., 7) of other peers in the system, called u's *successors* (similarly, we will refer to u as a *predecessor* of such a successor peer). The successor peers are determined by a set of k hash functions $\{h_1(u), ..., h_k(u)\}$ on u's identifier.[3] Moreover, we will assume that a *round* is roughly one week. Note that it does not matter when exactly in this time interval a peer sends its report to the successors, i.e., when a peer u cannot contact a peer v (e.g., because v is offline), it can try again later.

A peer u always informs the same set of peers, *independently* of which peers contributed resources to u. Upon receiving a vector, a peer can check whether it has been sent by a correct predecessor by verifying the hash function[4]—otherwise, it can simply drop the vector. The "observed" contribution value of the predecessor itself is not taken into account, in order to render the most attractive attack impossible by design. What is more, each peer can also ignore the vector of a peer that sends too frequently (more than once per round). Therefore, a peer can only attack the system with a false praise or accusation of another node. However, such an attack can either be detected or it will be averaged out, as we will see in Section VIII.

Unfortunately, sending direct observations to the k successors is still not sufficient in order to accurately estimate the contributions of all peers in large networks. Therefore, we extend the mechanism as follows: Upon receiving the observation vectors from its k predecessors, each peer aggregates them with its own observations, and sends the new vector to its k successors. Thus, one vector can summarize a large number of observations. However, we have to make sure that the values in the vectors do not contain too many observations from the past which do not reflect the current behavior of each peer. What is needed is a scheme where old observations can be truncated in the observation vector, but where there are still enough observations to update the contribution vector after each round.

This is accomplished as follows. In every round, a peer puts its own observations into a vector $\vec{o}_0$ (so far denoted by $\vec{o}$). After each round, it sends a message to its k successors containing its own (direct) observations from this round ($\vec{o}_0$), the aggregated observations of its k predecessors from the last round ($\vec{o}_1$), the aggregated observations of the k predecessors of its own k predecessors from the round before the last round ($\vec{o}_2$), and so forth. The message thus contains a matrix $O := [\vec{o}_0, \ldots, \vec{o}_{r-1}]$. Upon receiving the matrix $O_i := [\vec{o}_1, \ldots, \vec{o}_r]$ from predecessor $i \in [1, k]$ (note that when sending, the index runs from $0 \ldots r-1$, and when receiving, it is renamed to $1 \ldots r$), a peer aggregates all observations and updates its contribution vector $\vec{c}$ accordingly. Thus, the vectors from previous rounds aggregate an exponentially growing number of observations. The oldest observations lie r rounds in the past.

A simplified description of the Havelaar reputation system is given in Algorithm 1. Here, the algorithm is generalized by an *aging factor* $\gamma \leq 1$. However, since the aggregation vectors already include many observations from the past, using $\gamma = 0$ is fine for our purpose, but can be increased in order to account for longer absences from the system.

Algorithm 1 Simplified Havelaar Reputation System

```
1: observe $\vec{o}_0$;
2: receive $O_1 := [\vec{o}_1, \ldots, \vec{o}_r], \ldots, O_k$ from predecessors;
3: for $j \in [1, r]$: $\vec{o}_j = \sum_{i=1}^{k} O_{ij}$;
4: $\vec{c} = \gamma\vec{c} + (1-\gamma)(\sum_{i=0}^{r} \vec{o}_i)$;
5: send $O := [\vec{o}_0, \ldots, \vec{o}_{r-1}]$ to $k$ successors;
```

In Havelaar, a peer u increases the contribution values only after downloading fragments from other peers, but it *never decreases* any contribution values if it has to provide upload bandwidth to some peer. This has the drawback that if two peers have contributed to the system equally, they will be assigned the same amount of download bandwidth, *independently* of their downloading behavior—it is questionable whether this is fair. However, as mentioned, we do not want to provide any disincentives for downloading in our network.

But the behavior of *excessive downloads* should be discouraged. Such downloads could trigger a vicious circle: because of excessive downloads, the network is congested, which in turn encourages other users to download in advance, resulting in an even more congested network. Therefore, in Havelaar, each peer u additionally maintains a second vector $\vec{d}$ (downloads). After another peer downloads from u, u will increase the download value of that peer. As opposed to the observation vector, the download vector will not be sent to other peers. Before allocating resources among competing peers, the download values will be subtracted from the respective contribution values of $\vec{c}$, and only then used to allocate the bandwidth. Since excessive downloaders are more likely to be involved in repeated interactions, they are eventually slowed down.

Finally, note that downloads could of course also be treated differently. For instance, downloads could be punished with a mechanism similar to the one used to reward uploads.

VI. Analysis

A. Overview

Assume that two peers u and v compete for the same upload bandwidth of a given peer w. In order to achieve the desired fairness, peer w should allocate the bandwidth to u and v according to their real, i.e., global, contribution values C_u^g and C_v^g, i.e., with respect to all transactions to which they have contributed. However, in Havelaar, peer w does not have precise information about C_u^g and C_v^g, but only knows the local approximations C_u^l and C_v^l (values from its contribution vector $\vec{c}$). Hence, we want to achieve a good approximation $C_u^g/C_v^g \approx C_u^l/C_v^l$ such that the peers indeed receive the corresponding share of the bandwidth.

In this section, we will analyze how many observations x are needed such that the ratios of the values in the local vectors $\vec{c}$ are an acceptable approximation of the ratios of the real contributions. Consequently, we can compute the number of rounds r that are necessary in Havelaar to get the required number of (aggregated) observations. Henceforth, let C_u and C_v denote C_u^l and C_v^l, respectively.

[3]Due to the hash function, some peers will have slightly more, others slightly less predecessors.

[4]To verify the predecessor identifier, public/private-key pairs among peers are presumed.

Our network consists of n peers, not all of which are always online. We simplify the analysis by assuming that at any time exactly $m < n$ peers are online. Furthermore, we assume that each peer downloads t fragments from other peers; the transactions are assumed to be distributed uniformly at random among the peers and over time. Hence, the probability of downloading a fragment from a given peer is $p = 1/m$.

B. Analysis

Whenever a peer downloads from peer u, it increases the contribution value C_u of peer u. Let us first assume that bandwidth and fragment sizes are equal to 1, that is, C_u is increased by 1 after each download from u. As explained before, peer u is chosen with probability p for every download. Therefore, C_u is a random variable. What is the probability distribution of C_u after x downloads?

This situation corresponds to a *balls-into-bins problem* [24], where x balls are tossed into m bins, and where $p = 1/m$ is the probability that a tossed ball lands in any given bin. For a given bin u, the ball tossing process is a sequence of x random, independent *Bernoulli trials*, each with a probability p of success. Therefore, the random variable C_u follows a *binomial distribution* $C_u \sim Bin(x,p)$, where $\mu_{C_u} = E(C_u) = x \cdot p$ and $\sigma^2_{C_u} = Var(C_u) = x \cdot p \cdot (1-p)$.

This assumes that peers u and v are online all the time and can thus be chosen for all x transactions. In reality, however, some peer u might be online much longer than some other peer v. Therefore, u is likely to be involved in more transactions and will hence also contribute more to the network. This should be reflected in the local approximations C_u and C_v.

Let us assume that peer u is online with a fixed probability p_u, and peer v with probability p_v. Based on the assumption that the transactions are distributed uniformly over time, peer u can only be chosen for $x_u = p_u x$ transactions on average, and peer v for $x_v = p_v x$ transactions.

Furthermore, since the ultimate goal of Havelaar is to encourage high upload bandwidth, we need to include the provided upload bandwidth into the model. Let us assume that peer u uploads fragments at a fixed bandwidth of b_u, and peer v at a bandwidth of b_v. Instead of adding 1 to the contribution value of each peer, the corresponding *bandwidth* is added.

Putting everything together, the mean of C_u is given by $\mu_{C_u} = E(C_u) = b_u \cdot p_u \cdot x \cdot p$, and the variance is $\sigma^2_{C_u} = Var(C_u) = b_u^2 \cdot p_u \cdot x \cdot p \cdot (1-p)$. Note that the bandwidth b_u is multiplied twice in the variance (b_u^2): The variance in the contribution C_u does not increase linearly in the bandwidth, but quadratically.

Of course, our model can be extended in several ways, for example by incorporating variable fragment sizes or issues of contention. However, this would be overly exact (cf technical report [13]); in order to keep things simple, we restrict ourselves to the main factors, omitting this generalization in our analysis.

For small x, the *coefficient of variation* σ_{C_u}/μ_{C_u} (or, similarly, also the *variance-to-mean ratio* $\sigma^2_{C_u}/\mu_{C_u}$) is large. However, for $x \to \infty$, it converges to 0. This indicates that with lots of observations, relative estimates become accurate enough. We are interested in the ratio of the contribution values of two peers competing for resources at the same time. Therefore, let us introduce a random variable Z reflecting this ratio: $Z := C_u/C_v$.

What is the mean and the variance of Z? Since C_u and C_v are independent, the following approximations are reasonable [26]:

$$\mu_Z = E(Z) \approx \frac{\mu_{C_u}}{\mu_{C_v}} + \sigma^2_{C_v}\frac{\mu_{C_u}}{\mu^3_{C_v}},$$

and

$$\sigma^2_Z = Var(Z) \approx \sigma^2_{C_v}\frac{\mu^2_{C_u}}{\mu^4_{C_v}} + \frac{\sigma^2_{C_u}}{\mu^2_{C_v}}.$$

As can be seen from these formulas, for $x \to \infty$, the variance decreases quickly, and hence, for lots of observations x, Z is a good approximation of the ratio of the real contributions of peers u and v. In the following subsection, we will make use of another helpful approximation of the coefficient of variation [29]:

$$\left(\frac{\sigma_Z}{\mu_Z}\right)^2 \approx \left(\frac{\sigma_{C_u}}{\mu_{C_u}}\right)^2 + \left(\frac{\sigma_{C_v}}{\mu_{C_v}}\right)^2 \qquad (1).$$

C. Results

We can now estimate the number of observations necessary for a good approximation, that is, for small coefficients of variation of Z. Plugging μ_{C_u}, σ_{C_u}, μ_{C_v}, and σ_{C_v} into (1) yields

$$\begin{aligned}\left(\frac{\sigma_Z}{\mu_Z}\right)^2 &= \left(\frac{\sqrt{b_u^2 p_u x(p-p^2)}}{b_u p_u x p}\right)^2 + \left(\frac{\sqrt{b_v^2 p_v x(p-p^2)}}{b_v p_v x p}\right)^2 \\ &= \frac{1-p}{p_u x p} + \frac{1-p}{p_v x p}.\end{aligned}$$

Solving this for x gives the following fact.

Fact 6.1: The number of observations needed in order to achieve a desired approximation (as expressed by the coefficient of variation) of the real contribution values is

$$x \approx -\frac{p_v(p-1) + p_u(p-1)}{\left(\frac{\sigma_Z}{\mu_Z}\right)^2 p_u p_v p}.$$

As an example, for a network with $n = 100,000$ peers, where at any time $m = \frac{1}{4}n = 25,000$ peers are online, and if we assume that $p_u = p_v = \frac{1}{4}$ (in Kangoo, storage peers are online for more than six hours per day), $x \approx 10^7$ observations are required for an acceptable coefficient of variation of 0.15.

Having computed the number of observations x which are approximately needed for an acceptable accuracy, we can now determine the number of aggregation rounds. Assuming that every peer makes t transactions (= observations), the number of rounds r is $r = \lceil \log_k \frac{x}{t} \rceil$. For the above example of $x = 10^7$ observations, assuming that each peer makes $t = 5,000$ transactions and sends its observations to $k = 7$ peers, $r \approx 4$ rounds are already enough!

VII. Communication Costs

The Achilles' heel of Havelaar are the communication costs: Every peer has to send—for example, once a week—the aggregated observations to its successors. However, we believe that in many practical systems, the burden is tolerable. Moreover, as described in the technical report [13], various compression techniques—e.g., due to sparseness—can be employed to reduce the size of the messages further.

Concretely, we have computed Havelaar's estimated message size [13] depending on the number of rounds r and the number of transactions t in the system, and assuming that the contribution values can be encoded with 8 bits each (in a simulation, the entropy was only ≈ 7 bits). We have then compared Havelaar to solutions where the contribution value is recorded in a DHT. In this approach, the contribution value of peer u is stored at a peer which is chosen based on a hash function $h(u)$. Since the contribution values in Havelaar are only approximations and since the local vector is only updated once per round, we have compared the communication costs to a DHT-based approach with the same level of approximation and the same amount of updates. That is, instead of updating the contribution value after each download and retrieving the contribution value before each upload, peers only update the contribution value in the DHT probabilistically (with a probability resulting in the same level of approximation as Havelaar), and they only update their local vector by a look-up operation once per round.

As an example, in a network with $n = 100,000$, $t = 5,000$, $k = 7$, and $r = 4$, the communication costs are ≈ 2.2 MB per peer and week. In comparison, an approximate DHT-based approach would require ≈ 3.7 MB. Thereby, the costs for the DHT-based approach are only a lower bound: More communication would be necessary in order to store the contribution values persistently. Havelaar scales much better in the number of transactions. For the same example, but with $t = 20,000$ transactions, the communication costs of Havelaar are ≈ 2.5 MB, compared to ≈ 13.8 MB of the DHT-based approach. Finally, for a network with $n = 1,000,000$ and $t = 40,000$, the communication costs in Havelaar are ≈ 23.3 MB and the DHT-based approach ≈ 87.7 MB. We refer the reader to the technical report [13] for a more detailed comparison.

In conclusion, although the costs of Havelaar can be relatively high for a small number of transactions (e.g., up to ≈ 1.73 MB for $t < 2000$ in a network of $n = 100,000$, compared to ≈ 1.51 MB for the DHT-based approach), our system scales much better than various forms of DHT-based solutions. Moreover, we believe that the communication costs are acceptable in many practical environments.

VIII. Attacks

Havelaar is designed to cope with peers aiming at selfishly consuming larger shares of resources than other peers. The fact that every peer can send its observations only to its k successors facilitates *local defenses*.

A peer uses several defense mechanisms. A receiver checks whether a sender is one of its predecessors, and otherwise ignores the report. Moreover, it can make sure that a peer does not report contributions too often.

A peer does not take into account observations about the reporting predecessor itself. Thus, the most attractive attack is made hard *by design*: It is only possible to falsely “praise” or “accuse” *another* peer.

In addition to limiting the range of possible values, other measures are taken in our system to detect and ignore false reports. Before updating the local contribution vector $\vec{c}$ on the basis of the observation matrices, for each value the average and variance is calculated. If one value is extremely large, it is considered an outlier with respect to the other $k-1$ values, and is dropped. Then, the average of the other $k-1$ values is taken as the input to the update function.

Clearly, one can think of several further local defense mechanisms. For instance, statistical measures could be included to detect a possible false report by studying the distribution (histogram) of the observation vector, e.g., by checking whether the histogram is spiked. In any of the above cases where the successor is suspicious of an attack, it could reduce the trust value associated with each predecessor. The trust value can be used to either drop observations by misbehaving peers, or to weigh their observation values accordingly. However, it has not been necessary to make use of these techniques so far.

Besides the advantages of local defense, the robustness of Havelaar comes from the *extensive aggregation*: A single wrong value hardly influences the overall outcome. In this sense, also the damage which can be done by a small fraction of colluding peers is limited. In particular, collusion is also made difficult by the fact that successor peers are determined by hash functions, and hence becoming a predecessor of a specific peer is hard.

In summary, Havelaar’s design is based on local defense and extensive accumulation, and unlike many other approaches does not rely on transitivity of trust. This renders attacking the system a difficult endeavour. Finally, recall that several other attacks such as Sybil attacks and whitewashing are tackled by mechanisms which are part of Kangoo and hence are external to Havelaar.

IX. Simulation

We have performed several simulations of Havelaar which fortify our results. In this section, we present the most interesting findings.

In Figure 1, the real ratio of the contribution values of two peers is compared to the ratio from the local approximation in a network of size $n = 100,000$, with $k = 7$ successors and $r = 4$ rounds. In each round, the peers change their upload bandwidth. In the first round, for instance, peer u contributed exactly three times more than peer v. Note that the approximation is shifted to the right; this is due to the fact that contribution values are only updated once a round, based on observations from the past. Note also that the standard deviation of the approximation is generally low—being higher when peers change their behavior abruptly.

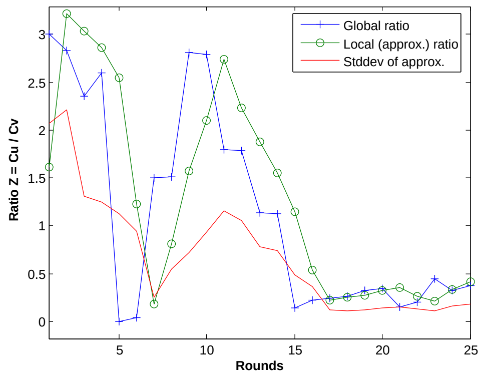


Fig. 1. Local approximation vs. global (real) contribution value of two peers in several rounds. Note that after a short bootstrapping phase in the beginning, the approximation becomes good quickly.

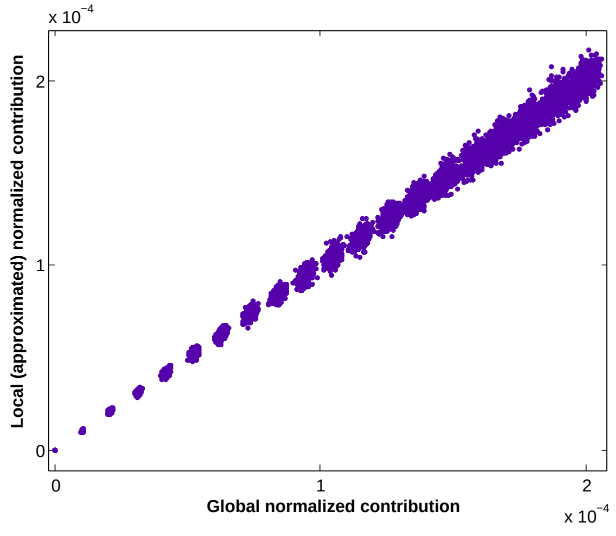


Fig. 2. Global normalized contribution value plotted against the locally approximated one.

The figure reveals that the bootstrapping process is quite fast, which implies that newly joining peers—or peers which return after a long period of absence—are up-to-date soon. Of course, this delay may still be unacceptably large for many systems where appropriate solutions which further speed up the process would be needed. However, this is not the case in Kangoo, as peers are expected to remain in the system for months or even years.

Figure 2 plots the local contribution vector for the same network against the global (real) contribution vector. Both vectors are normalized by dividing each entry by the sum of the whole vector. That is, each contribution value reflects the proportional contribution of the entire network. Again, the local approximation reflects the real contribution accurately (almost a straight line). For peers with higher contribution, however, the variance becomes larger. This is due to the fact that the variance is multiplied by the square of the bandwidth, as described in Section VI.[5]

X. Conclusions

The main goals of the Havelaar reputation system are (1) accurate estimation of the real contribution values of other peers, (2) robustness to selfish peers, and (3) efficiency, i.e., scalability in the number of transactions. This is achieved by a novel aggregation technique where peers always report the observed contributions values to the same set of peers. This allows for a local control of a peer's behavior. Encouraged by our results, we have integrated Havelaar in our distributed storage system Kangoo. We believe that Havelaar is a good choice for many active p2p systems requiring a fairness mechanism.

[5]Note, however, that the coefficient of variation is constant for every peer since it does not depend on the bandwidth.

References

[1] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proc. of the 10th Intl. Conf. on Information and Knowledge Management (CIKM)*, pages 310–317, 2001.
[2] E. Adar and B. Huberman. Free Riding on Gnutella. *First Monday*, 5(10), 2000.
[3] R. Axelrod. The Evolution of Cooperation. *Science*, 211(4489):1390-6, 1981.
[4] D. Banerjee, S. Saha, S. Sen, and P. Dasgupta. Reciprocal Resource Sharing in P2P Environments. In *Proc. 4th AAMAS*, 2005.
[5] S. Buchegger and J.-Y. L. Boudec. A Robust Reputation System for P2P and Mobile Ad-hoc Networks. In *Proc. 2nd Workshop on the Economics of Peer-to-Peer Systems*, 2004.
[6] B. Cohen. Incentives Build Robustness in BitTorrent. In *Proc. Workshop on Economics of Peer-to-Peer Systems*, 2003.
[7] J. R. Douceur. The Sybil Attack. In *Proc. 1st Int. Workshop on Peer-to-Peer Systems (IPTPS)*, pages 251–260. Lecture Notes in Computer Science (LNCS), Springer, 2002.
[8] M. Feldman and J. Chuang. Overcoming Free-Riding Behavior in Peer-to-Peer Systems. *ACM Sigecom Exchanges*, 6, 2005.
[9] M. Feldman, K. Lai, I. Stoica, and J. Chuang. Robust Incentive Techniques for Peer-to-Peer Networks. In *Proc. ACM Conf. on Electronic Commerce*, 2004.
[10] M. Feldman, C. Papadimitriou, J. Chuang, and I. Stoica. Free-riding and Whitewashing in Peer-to-Peer Systems. In *Proc. ACM SIGCOMM Workshop PINS*, 2004.
[11] F. D. Garcia and J.-H. Hoepman. Off-Line Karma: A Decentralized Currency for Peer-to-Peer and Grid Applications. In *Proc. 3rd Applied Cryptography and Network Security (ACNS)*.
[12] P. Golle, K. Leyton-Brown, and I. Mironov. Incentives in Peer-to-Peer File Sharing. In *Proc. 3rd ACM Conf. on Electronic Commerce (EC)*, 2001.
[13] D. Grolimund, L. Meisser, S. Schmid, and R. Wattenhofer. Havelaar: A Robust and Efficient Reputation System for Active Peer-to-Peer Systems. Technical report, TIK Report 246, available at http://www.tik.ee.ethz.ch/. ETH Zurich, Switzerland, 2006.
[14] D. Hughes, G. Coulson, and J. Walkerdine. Free Riding on Gnutella Revisited: The Bell Tolls? *IEEE Distributed Systems Online*, 6(6), 2005.
[15] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The Eigentrust Algorithm for Reputation Management in P2P Networks. In *Proc. WWW*, pages 640–651, 2003.
[16] F. Kuhn, S. Schmid, and R. Wattenhofer. A Self-Repairing Peer-to-Peer System Resilient to Dynamic Adversarial Churn. In *Proc. 4th Int. Workshop on Peer-To-Peer Systems (IPTPS), Ithaca, New York, USA*, February 2005.
[17] K. Lai, M. Feldman, J. Chuang, and I. Stoica. Incentives for Cooperation in Peer-to-Peer Networks. In *Proc. Workshop on Economics of Peer-to-Peer Systems (P2PEcon)*, 2003.
[18] Q. Lianz, Y. Pengx, M. Yangx, Z. Zhangy, Y. Daix, and X. Li. Robust Incentives via Multi-level Tit-for-Tat. In *Proc. 5th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2006.
[19] R. B. Ma, S. M. Lee, J. S. Lui, and D. Y. Yau. A Game Theoretic Approach to Provide Incentive and Service Differentiation in P2P Networks. In *SIGMETRICS*, 2004.
[20] S. J. Nielson, S. Crosby, and D. S. Wallach. A Taxonomy of Rational Attacks. In *Proc. 4th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, pages 36–46, 2005.
[21] A. M. Odlyzko. The Case Against Micropayments. In *Financial Cryptography*, pages 77–83, 2003.
[22] T. G. Papaioannou and G. D. Stamoulis. Effective Use of Reputation of Peer-to-Peer Environments. In *Proc. IEEE/ACM CCGRID 2004, GP2PC Workshop*, 2004.
[23] T. G. Papaioannou and G. D. Stamoulis. Reputation-based Policies that Provide the Right Incentives in Peer-to-Peer Environments. *Computer Networks*, 50(4):563–578, 2006.
[24] M. Raab and A. Steger. "Balls into Bins" - A Simple and Tight Analysis. In *Proc. 2nd Int. Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 159–170. Springer-Verlag, 1998.
[25] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content Addressable Network. In *Proc. of ACM SIGCOMM 2001*, 2001.
[26] J. A. Rice. *Mathematical Statistics and Data Analysis*. Duxbury Press, 1995.
[27] A. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proc. 18th IFIP/ACM Int. Conf. on Distributed Systems Platforms (Middleware)*, pages 329–350, 2001.
[28] J. Shneidman and D. C. Parkes. Rationality and Self-Interest in Peer to Peer Networks. In *Proc. 2nd Int. Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.
[29] W. Stahel. *Statistische Datenanalyse. Eine Einfuehrung fuer Naturwissenschaftler*. Vieweg, Braunschweig, 2000.
[30] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proc. ACM SIGCOMM Conference*, 2001.
[31] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. KARMA: A Secure Economic Framework for P2P Resource Sharing. In *Proc. P2PEcon*, 2003.
[32] W. Wang and B. Li. Trust Based Incentive in P2P Network. In *Proc. Int. IEEE Conf. on E-Commerce Technology for Dynamic E-Business (CEC-East)*, pages 302–305, 2004.
[33] W. Wang and B. Li. Market-driven Bandwidth Allocation in Selfish Overlay Networks. In *Proc. IEEE INFOCOM*, 2005.

Manipulability of PageRank under Sybil Strategies

Alice Cheng [*]   Eric Friedman [†]

Abstract

The sybil attack is one of the easiest and most common methods of manipulating reputation systems. In this paper, we quantify the increase in reputation due to creating sybils under the PageRank algorithm. We compute explicit bounds for the possible PageRank value increase, and we use these bounds to estimate the rank increase. Finally, we measure the effect of sybil creation on nodes in a web subgraph. We find that the resulting rank and value increases agree closely with the analytic values.

1 Introduction

Ranking systems are an important tool in a wide range of online settings, such as online shopping (Amazon, eBay), as a means of inferring reputation of sellers or goods; in the peer-to-peer setting, to weed out untrustworthy or freeloading users; and the area of online search, as a means of ranking webpages.

However, many ranking systems are vulnerable to manipulation, and users often have incentives to cheat. A higher ranking may offer an economic benefit - for example, a study of the eBay reputation system found that buyers are willing to pay a premium of 8% for buying from sellers with high reputation [11]. Another example is in online search, where websites, in order to gain web traffic, use the services of online companies which help sites improve their search engine rankings.

PageRank is currently one of the most widely used reputation systems. It is applied in peer-to-peer networks in the EigenTrust algorithm [7], and in web search, as the foundation for the Google search algorithm [9]. Although PageRank has proven to be a fairly effective ranking system, it is easily manipulable by a variety of strategies, such as collusion or the sybil attack [12, 5].

*Center for Applied Mathematics, Cornell University, alice@cam.cornell.edu

†School of Operations Research & Industrial Engineering and Center for Applied Mathematics, Cornell University, ejf27@cornell.edu

0Work supported by NSF. ITR-0325453

We focus primarily on the sybil attack, described by Douceur [4]. In this attack, a single user creates several fake users - called sybils - who are able to link to (or perform false transactions with) each other and the original user. For example, in the web, a user can create new webpages and manipulate the link structure between them. In many online settings, new identities are cheap to create, and it may be difficult to distinguish between sybils and real users. In the case of PageRank, users have already been observed performing sybil-like strategies, such as forming link farms [5].

It is easy to see that PageRank is vulnerable to sybil attacks. However, as we showed in earlier work, almost all practical reputation systems are vulnerable to sybil attacks [3]. It may be unrealistic to restrict one's attention only to sybilproof reputation systems, and reputation systems may vary widely in their exploitability. For example, the indegree reputation function (where a user's reputation value is his indegree) is easily exploitable - a user may increase his indegree to any desired value by creating sybils. On the other hand, a reputation function based on maximum flow is not sybilproof with respect to rank, but is more difficult to manipulate. Thus, it becomes important to gauge the degree of vulnerability of different reputation systems. In order to systematically compare PageRank to other reputation systems, we develop a method of estimating the potential PageRank rank and value improvement of a node in a web-like graph.

In this paper, we begin this research program with a formal and experimental analysis of the vulnerability of PageRank to sybil attacks. We provide analytic estimates of this vulnerability, which only depend on the overall PageRank distribution in the graph and then check the tightness of our analysis on empirical web graph data. We find a very close agreement and are led to believe that our estimates can be applied to estimate the vulnerability of PageRank on web-like graphs.

2 Related Work

Our work is related to [12] which considers the effect of collusion on PageRank. Collusion is a strategy where users mutually agree to alter their outlink structure in order to improve their reputations. Collusive strategies and sybil strategies differ in at least two critical ways. First, a sybil creator can gain reputation at the expense of his sybils, while colluders are unlikely to cooperate unless both can raise their reputations. Second, sybil strategies are likely to be less constrained in size - a user can often easily create a large sybil group, while it may be more difficult to find an equal number of users to form a colluding group.

Other related work includes Gyongyi and Garcia-Molina who give a fairly exhaustive list of strategies to falsely boost reputation on the web [5]. The PageRank algorithm itself has generated a lot of interest and study. Bianchini, Gori, and Scarselli consider the total PageRank within a community of nodes, and give methods for a community to boost its total reputation [2]. A survey paper by Langville and Meyer gives a general overview of the PageRank algorithm, and discusses many issues including PageRank stability and efficient computation [8].

3 Preliminaries

Given a set of users V, we represent the setting as a directed graph $\mathcal{G} = (V, E)$. The edges E represent direct trust between users. For example, in the web, an edge $(i, j) \in E$ may represent a hyperlink from site i to site j. Let $n = |V|$. Let $d(i)$ be the outdegree of the node $i \in V$. We require that every node has positive outdegree. Since this isn't always the case for real world graphs, we will insert a self-loop for all nodes with outdegree 0. We will assume that no other nodes have self-loops.

3.1 PageRank

The PageRank values on a network graph $\mathcal{G}$ are given by the stationary probabilities of the following random walk on $\mathcal{G}$: with probability $1 - \epsilon$, a walker at a node i walks along an outgoing edge of i, choosing the edge uniformly with probability $\frac{1}{d(i)}$, and with probability ϵ, jumps to a node chosen uniformly at random. Let v be the vector of stationary probabilities - v_i is the stationary probability of the node i. The resulting PageRank ranking is given by the order of the values of v, sorted from highest to lowest (note that a higher value v_i corresponds to a lower numbered rank). For convenience, we will typically not talk about the stationary vector of probabilities v, but will instead use $\pi = nv$. Clearly, π yields the same ranking as v. For a node i, we will refer to π_i as its *PageRank value* and its order on a highest to lowest list sorting the π_j's as its *rank*.

Given $\mathcal{G}$, we can construct the adjacency matrix of $\mathcal{G}$, A, $A_{ij} = 1$ if $(i, j) \in E$, and 0 otherwise. Let $M(\mathcal{G})$ be the matrix given by $M(\mathcal{G})_{ij} = \frac{A_{ji}}{d(j)}$.

Note that π is the principal eigenvector (with eigenvalue 1) of the matrix $(1 - \epsilon)M(\mathcal{G}) + \frac{\epsilon}{n}\overrightarrow{1}\,\overrightarrow{1}^T$, where $\overrightarrow{1}$ is the vector of all ones. That is, π satisfies the following matrix equation:

$$(1 - \epsilon)M(\mathcal{G})\pi + \epsilon\,\overrightarrow{1} = \pi$$

We may sometimes find it convenient to express the above as a scalar equation: for a node $i \in V$,

$$\pi_i = (1 - \epsilon)\sum_{j \to i} \frac{\pi_j}{d(j)} + \epsilon,$$

where $j \to i$ to denotes $(j, i) \in E$ (i.e. j points to i).

We can also consider the iterative version of the above equations, where $\pi_i^t \to \pi_i$ as $t \to \infty$ [8].

$$\pi_j^0 = 1, \forall j; \pi_i^t = (1 - \epsilon)\sum_{j \to i} \frac{\pi_j^{t-1}}{d(j)} + \epsilon$$

3.2 Sybil Strategies

In a sybil strategy, a node creates k sybils, and manipulates his own outlinks and those of his sybils. More formally,

Definition 1 *Given a graph $\mathcal{G} = (V, E)$ and a node $i \in V$, a* ***sybil strategy*** *for the node i, is a new graph $\mathcal{G}' = (V', E')$, such that $V' = V \cup S$, where $S = \{s_1, \ldots, s_k\}$ is a set of sybils (disjoint from the original node set) and E' is such that for all $j \in V, j \neq i$, for all $x \in V$, $(j, x) \in E \Leftrightarrow (j, x) \in E'$.*

A *sybil collective* is the node set $S \cup \{i\}$ (i and its sybils). Let r_i be the rank of i in $\mathcal{G}$, π_i be the PageRank value of i in $\mathcal{G}$. Let ρ_i be the new PageRank value for i in $\mathcal{G}'$ and r'_i be the new rank. Then a strategy is successful for i with respect to value if $\rho_i > \pi_i$. It is successful with respect to rank if $r'_i < r_i$.

We say that a reputation function is value (or rank) *sybilproof* if for all graphs $\mathcal{G}$, no node has a successful sybil strategy with respect to value (or rank).

It is straightforward to come up with an example where a node can increase its PageRank through creating sybils. In [3], we showed that no nontrivial symmetric reputation system (i.e. one that is invariant under a relabelling of the nodes) can be sybilproof. The version of PageRank that we described in the previous section is clearly symmetric, so there is a network where a node could benefit from creating sybils. Further, by this result, we know that adjusting some of the parameters of PageRank (such as the value of ϵ) in a nontrivial way while maintaining symmetry cannot yield a sybilproof mechanism. However, it is easy to show that even an asymmetric version of PageRank (such as the version used in EigenTrust) may be manipulated with sybils.

Note that a sybil creator may choose any configuration of edges within the sybil collective. However, for the purposes of this paper, we focus on one particular sybil strategy. In this strategy, a node i removes his outlinks, creates k sybils, and links to each of his sybils. The sybils link only to the sybil creator i. Figure 4 (in the appendix) depicts a node applying this strategy with 3 sybils.

Bianchini et. al. show that this configuration concentrates the maximum amount of reputation on the sybil creator [2]. Intuitively, any random walk inside the sybil collective must hit i on every other step. Further, removing any links from the collective to nodes outside of the collective improves their overall PageRank - a random walk which enters the collective must remain there until a random jump.

4 Analysis

Our main results are analytic bounds for the value increase upon creating sybils which are presented below. We then compare these bounds with empirical data.

4.1 Value Increase

We give the following upper and lower bounds for value increase:

Theorem 2 *Let π be the old PageRank value vector, and ρ be the new PageRank vector when node i creates k sybils by the above strategy, keeping all other nodes fixed. Then, if i has no self-loop in the original graph, we have the following bounds:*

$$\pi_i + k\frac{1-\epsilon}{2-\epsilon} \le \rho_i \le \frac{\pi_i + \epsilon(1-\epsilon)k}{\epsilon(2-\epsilon)}$$

Since we typically talk about the ratio between ρ and π, we give the corresponding bounds for the value inflation ratio ρ_i/π_i:

$$1 + \left(\frac{1-\epsilon}{2-\epsilon}\right)\frac{k}{\pi_i} \le \frac{\rho_i}{\pi_i} \le \frac{1}{\epsilon(2-\epsilon)} + \left(\frac{1-\epsilon}{2-\epsilon}\right)\frac{k}{\pi_i}$$

A proof of this theorem is included in the appendix.

These bounds allow us understand how the value increase changes as we increase the number of sybils or vary ϵ. Further, for given values of ϵ and π, we can estimate the number of sybils needed to increase a node's reputation by some given amount.

Increasing k increases both the upper and lower bounds, and appears to yield larger increases in the value inflation ratio when π_i is small. For example, for $\epsilon = 0.15$, we have $1 + 0.46(\frac{k}{\pi_i}) \le \frac{\rho_i}{\pi_i} \le 3.6 + 0.46(\frac{k}{\pi_i})$, meaning that for a node with value π_i equal to the mean value 1, doubling one's value requires between 1 and 3 sybils. For a node with the median value, which is ≈ 0.3 in our data sample, it requires only 1 sybil.

The above bounds are tight. The upper bound is attained for nodes i that are contained in no cycles. One can show (using similar techniques as in the proof of the theorem), that in this case, the reputation of i's recommenders (those nodes j with $j \to i$) are unchanged when i removes its outlinks. With a simple computation (or by following the proof of the above theorem), the equality follows.

The lower bound is attained for subgraphs in a "petal" configuration, where the node i points only to nodes who point only back at i (as in the sybil configuration). This configuration attains the lower bound because i's recommenders were previously "sybil-like", in that they attained most of their reputation from i and returned as much reputation as possible to i. Once i removes its outlinks, the value of the links (j, i) to i become very small.

However, most nodes may not lie in either of the extremes described above. Indeed, it is reasonable to expect (due to the observed high clustering in the web [1]) that some nodes lie on short cycles, leading to configurations similar to the "petal". At the same time, some of the edges out of i are likely not part of short cycles, suggesting configurations as in the upper bound.

4.1.1 Data for $k = 1$

In this section and the ones that follow we use a dataset from a webcrawl, available at [6]. The total number of nodes is $n = 281,903$. We preprocess the graph to insert self-loops for each node with outdegree 0, to guarantee that the matrix $M(G)$ of the graph (defined above) is indeed stochastic. In the first experiment, we select 10000 nodes uniformly at random from the graph, and for each node selected, we create a single sybil for that node under the above sybil strategy, keeping all other nodes fixed. We set

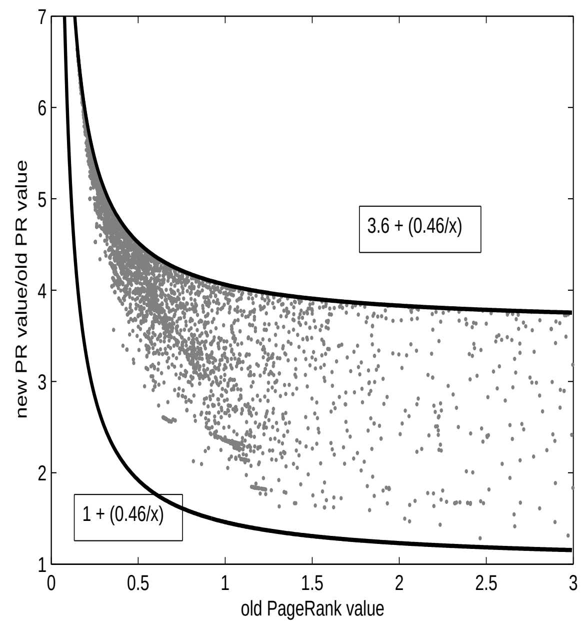


Figure 1: Old PageRank value (x axis) versus new value/old value ratio for the case of 1 sybil, jump parameter $\epsilon = 0.15$. The lines are given by the theoretical upper and lower bounds.

the jump parameter ϵ to 0.15.

In Figure 1, we plot the old PageRank value π_i versus the ratio $\frac{\rho_i}{\pi_i}$. For the sake of visual clarity, we cut off the graph at $\pi_i = 3$, which still includes the vast majority of the nodes ($\sim 97\%$). We observe that the nodes are able to achieve an average linear factor increase of 4.7. Further, the upper and lower bounds of the data appear to match closely with the computed bounds. For larger values of π_i, we found that both the upper and lower bounds appear to be tight, and the data points are roughly evenly distributed between the bounds. For smaller values, the upper bound appears tight, while the lower bound is not. One possible explanation for this discrepancy is that the lower bound (as discussed in the previous section) is attained when the original node is in a sybil-like structure, where the node is contained nearly exclusively in small cycles (i.e. many paths out of the node are small cycles). However, being in such a structure may also suggest a higher reputation value than a typical node, so nodes that nearly attain the lower bound may tend also to have higher reputation. In fact, the central node of a petal will have a PageRank value $\pi_i \geq 1$, and we note that the lower bound appears tight in this regime in our plot.

Further, we can note that aside from the deviation from the lower bound for lower reputation nodes, the nodes appear fairly evenly spread between the bounds, suggesting, as we stated earlier, that most nodes are widely distributed between the extremes of being in no cycles and being in many short cycles.

4.1.2 Data for $k = 1, 2, 5, 10$

For this experiment, we select a node uniformly at random from the graph. For each node selected, we set up a sybil strategy for that node with $k = 1, 2, 5, 10$. We set the jump parameter to $\epsilon = 0.15$. We repeat this 1000 times.

We plot the ratio of new PageRank to old PageRank in Figure 5, in the appendix. The data points appear roughly of the same shape as in the $k = 1$ case, and the boundaries of the data points agree with our computed bounds. Further, as in our bounds, we can observe that lower value nodes tend to gain larger increases with k and higher value nodes tend to have more modest increases.

4.2 Rank Increase

In many settings (such as web page ranking) one cares mainly about the ranking implied by the PageRank values and not the actual values themselves. In this section we evaluate the rank increases for a large class of graphs based on an analysis of a large web graph.

Given the value bounds from Theorem 2, if we assume that the PageRank values of most other nodes remain roughly fixed, we can estimate the rank increase using the PageRank distribution. Pandurangan et.al. estimate the probability density of PageRank in a large web subgraph, and find a density of $\approx \frac{c}{x^{2.1}}$, where c is a constant [10]. If we assume that the PageRank density is $F(x) = \frac{c}{x^{2.1}}$, then $Pr(\pi_i \geq x) = \frac{d}{x^{1.1}}$ for some constant d. For a node i, if its PageRank value is π_i, a rough estimate of its rank would be $nPr(\pi_i \geq x) = \frac{nd}{x^{1.1}}$. We found that our dataset matches the rough estimates above fairly closely - for nodes with rank < 40000, the value to rank function is $\approx c_1 v^{-1.1}$, and for nodes with rank > 50000, the value to rank function is $\approx c_2 v^{-0.86}$.

Let r_i be the old rank of i and r'_i be the new rank. Let $r(x) = cx^{-1.1}$ be the PageRank value to rank function (for some constant c). Then, the new rank to old rank ratio $\frac{r'_i}{r_i} \approx \frac{r(\rho_i)}{r(\pi_i)} = (\frac{\pi_i}{\rho_i})^{1.1}$, using the PageRank value ratio bounds, satisfies the bounds

$$\left(\frac{1}{\frac{1}{\epsilon(2-\epsilon)} + (\frac{1-\epsilon}{2-\epsilon})\frac{k}{\pi_i}}\right)^{1.1} \leq \left(\frac{\pi_i}{\rho_i}\right)^{1.1} \leq \left(\frac{1}{1 + (\frac{1-\epsilon}{2-\epsilon})\frac{k}{\pi_i}}\right)^{1.1}$$

.

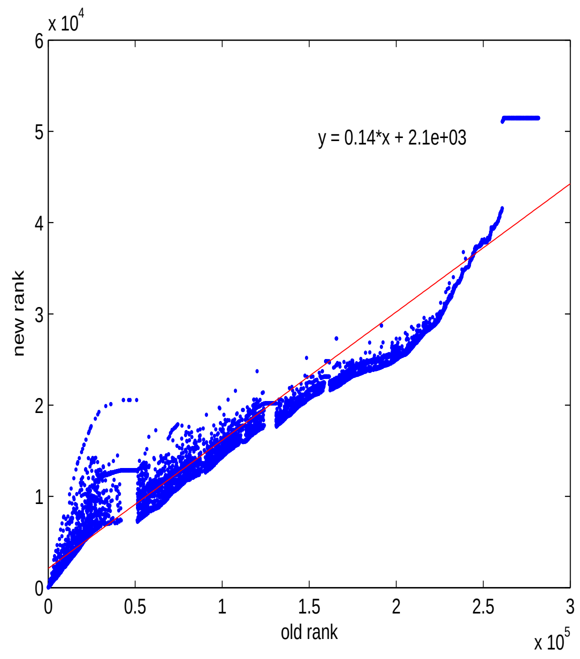


Figure 2: Old rank (x axis) versus new rank for the case $k = 1, \epsilon = 0.15$

For $\epsilon = 0.15$, and $k = 1$, the above bounds are $(\frac{\pi_i}{3.6\pi_i+0.46})^{1.1} \leq (\frac{\pi_i}{\rho_i})^{1.1} \leq (\frac{\pi_i}{\pi_i+0.46})^{1.1}$. For large π_i, we expect a lower bound in the rank increase of 0.28 and an upper bound of ≈ 1. For nodes with small π_i, say $\pi_i < 1$, which accounts for more than 80% of the nodes in our graph, we have a lower bound of approximately 0.13, and an upper bound of 0.66. From our analysis of the bounds for the value ratio in the previous section, we expect the lower bound to be much more accurate than the upper bound in the small π_i regime.

Given these tools, we can estimate the number of sybils needed for the median node (with rank $\frac{n}{2}$) to rise to the top k, for any k. Take the rank function $r(v)$ to be $r(v) = \frac{c}{v^{1.1}}$ for a constant c. Let $r_1 = \frac{n}{2}$ be the rank of the median node. $r_2 = k$. We can estimate the corresponding values for a graph of size n: $v_1 = r^{-1}(r_1) = (\frac{2c}{n})^{1/1.1}$, $v_2 = r^{-1}(r_2) = (\frac{c}{k})^{1/1.1}$. The value ratio $\frac{v_2}{v_1} = (\frac{n}{2k})^{.91}$. Plugging in the value ratio from the theorem inequalities (for $\epsilon = 0.15$), we have $k \sim \frac{\pi_i}{.46}(\frac{n}{2k})^{.91}$. Therefore, for a graph with ~ 300000 nodes, and median $\pi_i = 0.3$, a median node requires ~ 500 sybils to rise to the top 100. In a graph with median value $\pi_i < 1$, a median node would require less than ~ 76 sybils to rise to the top 1%.

4.2.1 Data for $k = 1$

The experimental setup is identical to the one described in section 3.1.1. We plot the old rank versus the new rank in Figure 3. We find that all but the very highest or very lowest ranked nodes are able to improve (or decrease) in rank by a factor of approximately 0.14 times - approximately a 6-fold improvement. This value agrees well with our computed lower bound (for small π) of 0.13. Further, we can observe that for nodes with original rank > 50000 (these nodes have $\pi_i > 1$), the improvement in rank is much more spread out, and less significant - which may be explained by the fact that the PageRank value ratios are more spread out, and attain the upper and lower bounds in the large π_i regime.

4.2.2 Data for $k = 1, 2, 5, 10$

The experimental setup here is identical to the one described in section 3.1.2. We plot the old rank versus the new rank in Figure 6 (in the appendix) for $k = 1, 2, 5, 10$.

We see a much more dramatic improvement in rank than value resulting from increasing the number of sybils. We find average ratios of old rank to new rank, $\frac{r_i}{r'_i}$ of 7.1 for $k = 1$, 16.4 for $k = 2$, 40 for $k = 5$, and 90.9 for $k = 10$. As expected, as in the value case and suggested by our bounds, sybil creation tends to be more effective for higher ranked (i.e., lower π_i) nodes.

4.3 Varying ϵ and sybil strategies

One way to vary the PageRank algorithm is to alter the parameter ϵ, which determines the probability of making a random jump at each step of the random walk. Our value bounds show that as ϵ increases, the potential increase in value declines. Intuitively, if ϵ is high, the effect of creating sybils may be reduced, since a random walk does not remain trapped in sybil collectives for a long time. By repeating the previous experiments for various values of ϵ, we found that the value increase does decline predictably as ϵ increases. However, nodes were still able to achieve significant rank improvements as we increased ϵ. In fact, higher values of ϵ yielded slightly higher average increases in rank for sybil-creating nodes. Figure 6 plots the average old rank to new rank ratio as ϵ varies. Though the value increase declines as ϵ increases, raising ϵ increases the likelihood of choosing a node at random in the PageRank random walk, making the overall PageRank distribution more uniform, compressing the set of typical pagerank values

We also considered two different sybil strategies. In one, users do not remove their outlinks to non-sybil nodes. In the other, users move their outlinks to a sybil node. In both of these cases, we observed an improvement in PageRank value and rank, though slightly less than in the original strategy.

5 Future Work

Our analysis shows that PageRank is extremely manipulable, even with simple strategies using a small number of sybils. We provided tight analytic approximations that can be used to estimate the manipulability of Pagerank in a variety of settings.

One issue that we haven't considered is the correlations between web pages on similar topics. For example, typically - and particularly in the web setting - a node is competing with a subset of nodes relating to the same topic (e.g. an electronics retailer probably doesn't care about ranking above a political weblog). Therefore, one potential further area of study is an analysis of how much the improvements observed above allow a typical node to beat its most likely competitors. Further the subset of competitors may look very different from a uniformly random subset of the web. For example, a subset of nodes all relating to the same topic may be more clustered than a random subset of the web. Is sybil creation more effective or less in this setting?

In this paper, we focus entirely on the PageRank algorithm, and find that it is easily manipulable. However, there are many other potential reputation systems, and we do not expect all of them to be as easily manipulable with sybils. Similar studies on the manipulability of other reputation systems may allow direct comparison of the manipulability of various reputation systems.

In particular, one would expect that there would be a trade off between the quality of the ranking system its manipulability. For example, as shown in [3], the "shortest path" ranking system is immune to sybil attacks; however, it is most likely less effective at ranking than PageRank. The development of robust and efficient ranking mechanisms is an important open problem.

References

[1] Lada Adamic. The small world web. In S. Abiteboul and A.-M. Vercoustre, editors, *Research and Advanced Technology for Digital Libraries, Lecture Notes in Comp. Sci.,1696*, pages 443–452. 1999.

[2] Monica Bianchini, Marco Gori, and Franco Scarselli. Inside pagerank. *ACM Transactions on Internet Technology*, 5(1), February 2005.

[3] Alice Cheng and Eric Friedman. Sybilproof reputation mechanisms. In *Third Workshop on the Economics of Peer-to-Peer Systems*, 2005.

[4] J. Douceur. The sybil attack. In *Proceedings of the IPTPS02 Workshop*, 2002.

[5] Z. Gyongyi and H. Garcia-Molina. Web spam taxonomy. In *In First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb)*, 2005.

[6] Sepandar Kamvar. stanford.edu web crawl, 2002, http://nlp.stanford.edu/ sdkamvar/data/stanford-web.tar.gz.

[7] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the Twelfth International World Wide Web Conference (WWW)*, 2003.

[8] Amy Langville and Carl Meyer. Deeper inside pagerank. *Internet Mathematics*, 1(3), 2004.

[9] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. In *Proceedings of the 7th International World Wide Web Conference*, pages 161–172, 1998.

[10] Gopal Pandurangan, Prabhakar Raghavan, and Eli Upfal. Using pagerank to characterize web structure. In *8th Annual International Computing and Combinatorics Conference (COCOON)*, 2002.

[11] Paul Resnick, Richard Zeckhauser, John Swanson, , and Kate Lockwood. The value of reputation on ebay: A controlled experiment. Working paper, available at http://www.si.umich.edu/ presnick/papers/postcards/index.html, 2004.

[12] Hui Zhang, Ashish Goel, Ramesh Govindan, Kahn Mason, and Benjamin Van Roy. Making eigenvector-based reputation systems against collusions. In *The Third Workshop on Algorithms and Models for the Web Graph*, 2004.

6 Appendix

In this section, we wish to prove the following theorem:

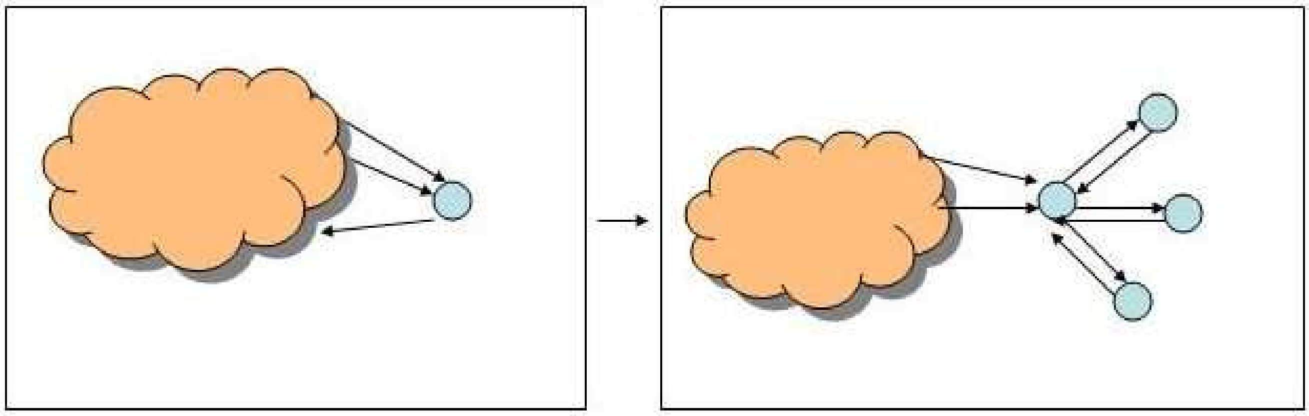


Figure 3: On the left: a single node with both outlinks and inlinks from the rest of the graph (cloud). On the right: the node removed its outlink, and created 3 sybils, arranged in the "petal" formation.

Theorem 3 *Let π be the old PageRank value vector, and ρ be the new PageRank vector when node i creates k sybils by strategy A, keeping all other nodes fixed. Then, if $d(i) > 0$, we have the following bounds:*

$$\rho_i \leq \frac{\pi_i + \epsilon(1-\epsilon)k}{\epsilon(2-\epsilon)}$$

$$\rho_i \geq \pi_i + k\frac{1-\epsilon}{2-\epsilon}$$

Let $G = (V, E)$, be a directed graph with $V = \{1, \ldots, n\}$. For $j \in V$, let $d(j)$ be the outdegree of the node j. We define $M(G)$ be the $n \times n$ matrix such that

$$M(G)_{ij} = \left\{ \begin{array}{cc} \frac{1}{d(j)} & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{array} \right.$$

Define $\tilde{M}, v, w$ such that

$$M(G) = \left\{ \begin{array}{cc} \tilde{M} & w \\ v^T & 0 \end{array} \right\}$$

WLOG let $i = n$. Let $G' = (V, E')$ be the graph where n removes its outlinks and creates a self loop. Let G'' be the graph where n has k sybils as in strategy A.

Let π be the original PageRank vector for G, with $\|\pi\|_1 = n$, and let π' be the PageRank vector for G', with $\|\pi'\|_1 = n$. Let ρ be the $n+k$ vector such that $\rho_x = \pi'_x$ for all $x < n$, $\rho_n = \frac{1-\epsilon}{2-\epsilon}k + \frac{1}{2-\epsilon}\pi'_n$ and $\rho_x = \frac{1}{2-\epsilon} + \frac{1-\epsilon}{2-\epsilon}\frac{\pi'_n}{k}$ for all $x > n$. By considering the matrices $M(G'), M(G'')$ in block form as above, an easy computation shows that $(1-\epsilon)M(G'')\rho + \epsilon\overrightarrow{1} = \rho$. Therefore, ρ is the unique PageRank vector of G'' (normalized to $n+k$).

It suffices then to show that $(2-\epsilon)\pi_n \leq \pi'_n \leq \frac{\pi_n}{\epsilon}$

Lemma 4 $\pi'_j \leq \pi_j$ *for all* $j < n$.

Proof: Note that the outdegrees of nodes $j < n$ in G' are equal to their outdegrees in G, so we can write the outdegree of x for $x < n$ as $d(x)$. Recall the iterative version of PageRank: $(\pi'_j)^t = (1-\epsilon)\sum_{x \to j} \frac{(\pi'_x)^{t-1}}{d(x)} + \epsilon$, for $t \geq 1$, and $(\pi'_j)^0 = 1$ for all j. Since $(\pi'_j)^t \to \pi'_j$ as $t \to \infty$, it suffices to show that $(\pi'_j)^t \leq \pi^t_j$ for all t, and for all $j < n$. This is trivially true for $t = 0$. By induction, assume that $(\pi'_x)^{t-1} \leq \pi^{t-1}_x$ for all $x < n$. Consider some node $j < n$.

$$\begin{aligned} (\pi'_j)^t &= (1-\epsilon) \sum_{x:(x,j)\in E'} \frac{(\pi'_x)^{t-1}}{d(x)} + \epsilon \\ &\leq (1-\epsilon) \sum_{x:(x,j)\in E, x<n} \frac{(\pi_x)^{t-1}}{d(x)} + \epsilon \\ &\leq \pi^t_j \end{aligned}$$

The first inequality follows from induction and the fact that n doesn't point to any $j < n$ in G'. ■

Plugging in π_j for each π'_j in the PageRank formula for π_n gives the upper bound. For the lower bound, we have the following lemma:

Lemma 5 $\pi'_n \geq (2-\epsilon)\pi_n$.

Proof: Note that we require the assumption that $d(n) > 0$ for this lemma. Consider a node $i \neq n$. In the graph G', we have

$$\pi'_i = (1-\epsilon) \sum_{j \to i, j \neq n} \frac{\pi'_j}{d(j)} + \epsilon,$$

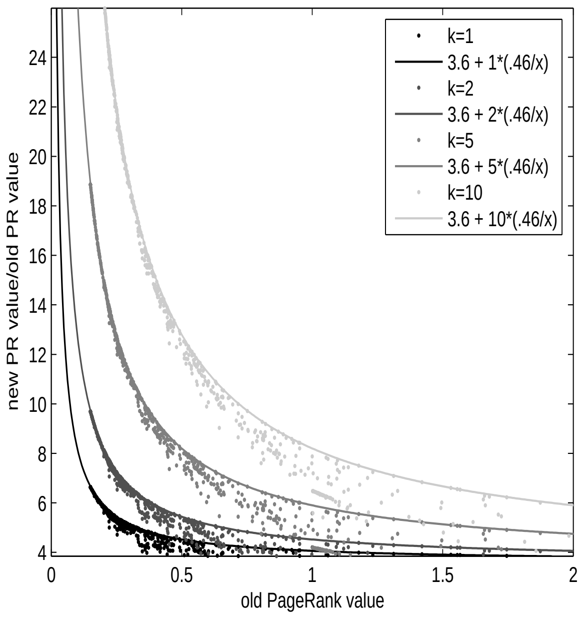


Figure 4: Old PageRank value (x axis) versus old value/new value ratio (y axis) for $k = 1, 2, 5, 10$. The lines are the theoretical upper bounds for the various values of k

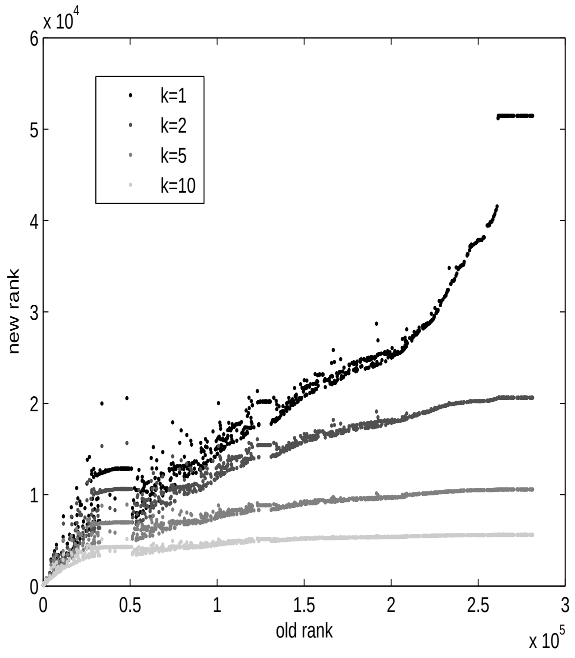


Figure 5: Old rank versus new rank for $k = 1, 2, 5, 10$, $\epsilon = 0.15$

by the fact that n points to no nodes other than itself in G'. Applying the previous lemma, we have

$$\pi'_i \leq (1-\epsilon) \sum_{j \to i, j \neq n} \frac{\pi_j}{d(j)} + \epsilon.$$

Note that $\pi_i = (1-\epsilon) \sum_{j \to i, j \neq i} \frac{\pi_j}{d(j)} + (1-\epsilon)\delta_{ni} \frac{\pi_n}{d(n)} + \epsilon$, where $\delta_{ni} = 1$ if $(n, i) \in E$, and 0 otherwise. Therefore, we have

$$\pi'_i \leq \pi_i - (1-\epsilon)\delta_{ni} \frac{\pi_n}{d(n)}.$$

We can sum the inequality over all $i \neq n$:

$$\sum_{i \neq n} \pi'_i \leq \sum_{i \neq n} \pi_i - (1-\epsilon)\pi_n,$$

where we note that there are exactly $d(n)$ nodes among $i \neq n$ with $\delta_{ni} = 1$ (n had no self-loops in the original graph). Adding $\pi_i + \pi'_i$ to both sides, we have

$$\pi_i + \sum_{i \in V} \pi'_i \leq \pi'_i + \sum_{i \in V} \pi_i - (1-\epsilon)\pi_n.$$

Finally, by normalization, $\sum_{i \in V} \pi_i = \sum_{i \in V} \pi'_i = n$, so $\pi_i \leq \pi'_i - (1-\epsilon)\pi_n$. Rearranging, we get the desired inequality: $(2-\epsilon)\pi_i \leq \pi'_i$ ■

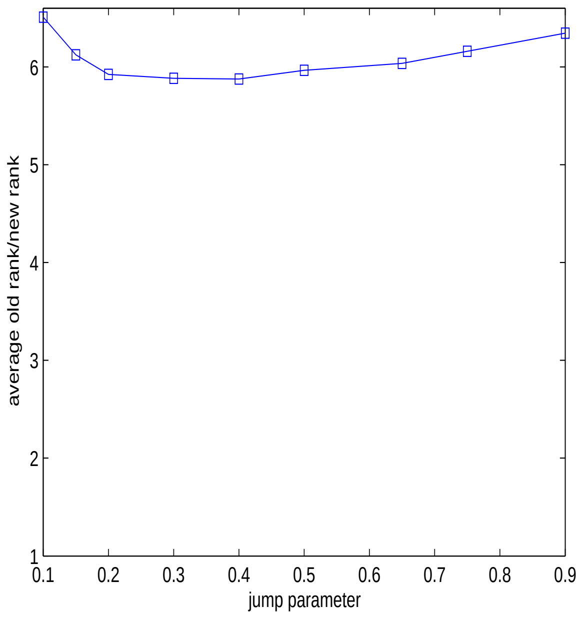


Figure 6: Jump parameter epsilon vs. average old rank/new rank ratio