Fully Self-Organized Fair Peering of Wireless Hotspots

Elias C. Efstathiou and George C. Polyzos

Mobile Multimedia Laboratory
Department of Computer Science
Athens University of Economics and Business
Athens, Greece
{*efstath*, *polyzos*}*@aueb.gr*

Abstract – We argue that wireless hotspot bandwidth can be exchanged as a peer-to-peer resource and we promote the idea of hotspot sharing, analogous to the traditional idea of file sharing. With hotspot sharing, users are encouraged to open their privately owned hotspots and offer Internet access to passersby. Payments are only "in kind" and not necessarily simultaneous. We are motivated by the widespread availability of wireless equipment that allows individuals to offer cellular-type services to their peers. We propose a fully self-organized approach to unified hotspot roaming that assumes absolutely no central trust, and we present and evaluate the design of a decentralized system that implements a fair peering algorithm applicable to hotspot sharing. Our mechanism exploits privately owned WLANs and builds a unified roaming infrastructure without the administrative overhead that is usually associated with such efforts.

I. INTRODUCTION

Imagine you are walking down a city street, your brand new Wireless LAN-enabled phone in hand. You are surrounded by WLAN signals – it seems that everybody is buying a wireless router for their DSL line these days. "What a waste of perfectly good bandwidth" you think when you realize that you are not allowed to access any of it. Still, however, for the first time in the history of telecommunications individuals can readily provide telecom services to their peers.

Can we share WLANs the same way we share files? That is the question we attempt to answer in this paper. If the question were a narrow technical one, the answer would be *yes*. However, two broader issues arise. First, a system for the peer-to-peer sharing of wireless hotspots may not reach its full potential if it relies on a trusted authority for its operation. Commercial hotspot aggregators [16, 18] know this only too well: their idea of uniting public hotspots under one "service mark" sounded promising, however the aggregator market today is, ironically, segregated. Perhaps if we completely remove roaming brokers from the equation we can allow for a unified roaming system to emerge. A second issue is that any proposal for peer-to-peer sharing of resources has to provide a very good answer to this question: "why should *I* share?" In this paper, we will present our proposal for a hotspot peering system that is fully self-organized and in which rational entities have an incentive to collaborate.

Briefly, our proposal is this. People should organize themselves into small groups that we call *teams* and start playing a game with three simple rules. First, each team must operate a number of public hotspots. Second, members of teams may be freely serviced by hotspots belonging to other teams only if they can prove that their team also freely services members from other teams. Third, there is no referee.

The assumption behind the game is that the threat of exclusion coupled with the promise of free roaming is a good enough reason to share one's hotspot with nearby visitors. If such a system were in operation, free roaming would be available to people for the price of a single subscription to a broadband ISP – something that millions already have. The system could also be built in small incremental steps and with no economic risk for anyone involved. However, as there is no referee and the players are economically rational entities, there is always the temptation to under-provide. We will show how, by following a simple protocol that is secured by standard cryptographic primitives, *free-riders* can reliably be detected and excluded from the system – which would provide incentives for the system to grow as more willing collaborators joined it.

In our design we use a fully self-organized public-key infrastructure and we assume all system identities are only simple public-private key pairs, i.e. they are *zero-cost* identities. We assume that teams distrust each other completely and no team interactions are possible – unless *both* teams have something to gain. This of course rules out the use of distributed protocols that assume cooperative nodes. We also assume all teams have equivalent roles, which rules out distributed authorities as well. Obviously our community is exposed to Sybil attacks [6] as every entity can join it with multiple identities, but our peering algorithm makes the Sybil attack irrelevant. We further assume no system module is tamperproof, as this would imply a trusted authority; therefore, all entities can deviate from our proposed protocol and the burden is on us to show that adherence to the protocol is rational.

We define *fair* peering to simply mean that teams *consume in proportion to their contribution*, and we discuss the issue of *unfairness attacks*. To guarantee fairness, we rely on the existence of proofs of prior transactions, which are essentially *receipts* signed by consumers that circulate through the system and form a graph which can identify free-riders – assuming that peers can indeed build a partial view of that graph in a decentralized and incentive-compatible manner. We use teams and not individuals to represent the peers because we want to include people that would otherwise not be able to consume enough even if they *were* willing to contribute (e.g. people that can only set up hotspots in the periphery of a city where demand for wireless Internet may be low). We assume complete intra-team trust and we argue for it in Section V.

Specific contributions of this paper include a generic fair peering algorithm based on non-simultaneous *n*-way exchanges (Section III-A) and a completely decentralized protocol that applies the fair peering algorithm to our hotspot-sharing system (Section III-C). We evaluate our protocol using simulations in Section IV.

II. Related Work

Other proposals are also addressing the problem of fuelling hotspot deployment. In an appropriately titled paper [2], Ben Salem *et al.* present a framework that motivates Wireless ISPs (WISPs) to provide access to each other's users by using a reputation mechanism that is maintained by a Trusted Central Authority

(TCA). The TCA could be a federation of WISPs or be under the control of a worldwide organization. Their proposal is an improvement on hotspot aggregation but still requires central trust.

Other efforts to fuel hotspot deployment include the various Free Networks [17] that are being deployed in cities worldwide. The free hotspots that are part of these "freenets" can be set up without centralized coordination. However, freenets in general rely on the altruism of their participants, which hinders their deployment. Our proposal can be seen as an incentive mechanism applicable to freenets.

Commercial platforms like *Linspot* [21] or *Netshare* [22] resemble the hotspot-sharing aspect of our proposal. In these services, users are encouraged to share their residential hotspots with nearby visitors and receive compensation for their contribution. In principle, this model is no different from hotspot aggregation and has to face the same segregation problems that affect the public hotspot market.

Generic attempts that provide incentives for fair sharing of resources in electronically mediated communities include micropayment mechanisms. An approach to micropayments suitable for peer-to-peer systems is the *PPay* proposal, presented in a paper by Yang and Garcia-Molina [15]. PPay unfortunately inherits the problems of currency-based economies, i.e. it requires a centralized broker.

The *Nuglets* approach [3] for stimulating cooperation in mobile ad hoc networks, and the *Karma* approach [14] for stimulating P2P resource sharing are two proposals for a decentralized currency economy. However, the Nuglets approach assumes that users first obtain tamperproof modules that manage their currency and protect against theft and forgery. Karma is susceptible to the Sybil attack: new entrants are provided start-up funds and therefore have an incentive to join the system using multiple identities. The system attempts to increase the cost of such action by requiring identities to include the solution to a cryptographic puzzle, which is only a small setback, as this limits only the *rate* at which new identities can be created. Another approach to a decentralized economy is adopted by the popular *Kazaa* [19] file-sharing system, where the user's software agent manages the user's own participation level (similarly to a Nuglets tamperproof module). This agent, however, can easily be tampered with [20].

The work most closely related to ours is by Anagnostakis and Greenwald [1] on exchange-based incentive mechanisms for file sharing. In their paper, the authors assume an environment without any form of central trust (i.e. the do not use cash- or credit-based mechanisms) and structure their system as an exchange economy. Quoting from their paper "Requests from peers that can provide a simultaneous, symmetric, service in return (*exchange transfers*) are given higher priority. The service need not be directly to the provider (a *two-way exchange*), but more generally priority is given to peers who participate in n-*way exchanges* to which the provider currently belongs [which are] implemented as *rings* of *n* peers, where each peer is served by its predecessor and serves its successor in the ring." In our approach, we adopt the principle of detecting *n*-way exchanges as an incentive-compatible way to reward contributors and exclude free-riders, and we generalize it to include *non-simultaneous* exchanges. Other

loosely related proposals that use exchange-based mechanisms include two storage-sharing systems: *Samsara* by Cox and Noble [5], and a proposal by Ngan *et al.* [13].

Two proposals that, like ours, require the realization of a graph in a decentralized manner, are NICE [11] and the work by Capkun *et al.* [4] on self-organized public-key management for mobile ad hoc networks. The NICE proposal describes a P2P trust graph that can help a "source" peer infer a trust value for a "sink" peer. The authors also discuss the incentives of peers to share their partial view of the graph with others. The work by Capkun *et al.* [4] is applicable to fully self-organized civilian ad hoc networks where users adopt PGP-style keys and certificates but can only have a partial view of the certificate graph. Similarly to our proposal, the authors exploit user mobility in order to circulate information via the merging of certificate repositories whenever two users meet.

The theoretical work on cooperation incentives by Feldman *et al.* [9, 10] provided useful insights to our design. More specifically, the authors studied among others the effect of the threat of exclusion, the effect of punishing *traitors* (peers that turn to free-riders after a period of collaboration – see Section IV-C), and the associated value of keeping limited history of prior transactions. Finally, a paper by Mahajan *et al.* [12] addresses many of the practical issues faced by designers of incentive-compatible protocols.

In our earlier position paper [7], we presented our idea for a P2P approach to roaming, but there we adopted the approach of a distributed bank that manages a currency-based economy, i.e. not a fully self-organized approach. In a follow-up paper [8], we replaced the bank with a centralized repository of receipts, and we assumed zero-cost identities and discussed the problem of Sybil attacks. In this paper we present a generic, fully self-organized fair peering algorithm that is appropriate for various types of resource-sharing communities, and we present and evaluate the design of a completely decentralized and incentive-compatible application of that algorithm to our hotspot-sharing community.

III. DESIGN

A. Fair peering via non-simultaneous n-*way exchanges*

Our aim is to induce contribution in a resource-sharing community of autonomous peers where the peers can both provide and consume the resource in question. Assuming rational and self-interested peers, the resource will be under-provisioned, i.e. *free-riding* on the contributions of other peers will prevail [9]. The threat of *excluding* free-riders from the community can work as an incentive for peers to contribute and for more resources to become available [10]. How can peers easily identify free-riders and ignore their requests? In a controlled cash- or credit-based economy, identifying and excluding those who do not contribute is straightforward. For example, with currency that is controlled by trusted banks, non-contributors would eventually run out of currency (assuming that currency cannot be forged). However,

as we require a completely decentralized system without any (distributed or centralized) trusted parties, we try to answer the above question in the context of a simpler exchange economy.

Our exchange-based mechanism for excluding free-riders is similar to the approach of simultaneous *n*-way exchanges used by Anagnostakis and Greenwald [1] that we presented in Section II, which we generalize here to include *non-simultaneous n*-way exchanges. Requiring simultaneity was too restrictive for hotspot peering as we define it. In hotspot peering, an example of a 2-way simultaneous exchange would be a visitor *C* that accesses a foreign hotspot *P* and is simultaneously providing service through *C*'s own hotspot(s) to the owners of hotspot *P*. Generalizing this to simultaneous *n*-way exchange rings could be less restrictive, however it would still mean that roaming visitors have to provide service to *someone* through their own hotspots *at the same time* that they access another hotspot as visitors, which still feels more restrictive than necessary.

To achieve non-simultaneity, we introduce *proofs of prior transaction*. To better describe our proof-based fair peering algorithm, we will present it in terms of an abstract model. Consider a directed graph *G*, where the vertices represent the peers and the edges represent the proofs of prior transaction. A single directed edge will be denoted $C \rightarrow P$, where *C* and *P* are the peers that it connects. In the $C \rightarrow P$ edge, *P* represents the peer that *provided* service, and *C* represents the peer that *consumed* service, i.e. an outgoing edge represents a consumption action and an incoming edge a contribution action. When a peer provides service to another, a new directed edge is added to *G*. We allow for multiple edges connecting the same two peers and pointing to the same direction (i.e. *G* is a *multigraph*), and we also assume that every edge has a unique label that distinguishes it from any other. So, if *P* provides service to *C* on two occasions, the graph will contain two distinct $C \rightarrow P$ edges, each with a unique label.

We define an *n-way exchange ring* to be an ordered list of *n* peers with the first (*n*–1) peers pointing an edge to their list successor, and with the *n*'th peer pointing an edge to the first peer. We further define a *potential n*-way exchange ring to be an ordered list of *n* peers with only the first (*n*–1) peers pointing an edge to their list successor. A potential ring consists of *n* peers and (*n*–1) edges, and can become an *n*-way exchange ring if a directed edge is added, pointing from the *n*'th peer to the first peer. Figure 1 shows examples of *n*-way and potential *n*-way exchange rings.

Our fair peering algorithm includes two steps. First, *if peer* C *requests service from peer* P, P *should provide service only if a new* $C \rightarrow P$ *edge can complete a potential* n-*way exchange ring*. Plainly, this means that *P* should provide service to *C* only if *P* had in the past consumed service from *C* (a non-simultaneous 2-way exchange), or if *P* had consumed service from another peer who in turn had consumed service from *C* (a non-simultaneous 3-way exchange), and so on. Second, *if an* n-*way exchange ring can be completed, the new* $C \rightarrow P$ *edge should be added to graph* G, *and* P *should* discard *the remaining* (n–1) *labeled edges of the ring*, where by *discard* we mean that, *only* from *P*'s point of view,

and if *P* is again asked to provide service in the future, these labeled edges should be ignored when searching *G*. Because edges cannot be removed from *G*, *P* is required to remember the edges it discards.

Discussion

Unfairness attacks. If peer *P* did not discard some of the (*n*–1) edges then a number of *unfairness attacks* would be possible. First, assume *P* had consumed in the past from peer *X* and that *G* contained edge $P \rightarrow X$. If *X* requested service from *P*, *P* would detect a potential 2-way exchange and provide service to *X*. However, if *P* did not discard the $P \rightarrow X$ edge in question, *X* could repeatedly request service from *P* taking advantage only one prior contribution, i.e. *X* would be able to consume disproportionately to its contribution. Second, assume *G* contained the edges $P \rightarrow A$ and $A \rightarrow B$. If *B* requested service from *P*, *P* would detect a potential 3-way exchange and would provide service to *B*. If *P* discarded only the $P \rightarrow A$ edge, and if *A* provided service to *P* again, *B* could take advantage of its one contribution to *A* and request service from *P* every time *A* provided to *P*.

Note that the algorithm requires that *P* should ignore $P \rightarrow X$ in the first case *irrespective of the identity of future requestors*. This is to protect against Sybil attacks: a cheating *X* could create multiple identities and "fake" edges that are pointing to them (e.g. $X \rightarrow X_1$, $X \rightarrow X_2$, $X \rightarrow X_3$, …) and exploit ad infinitum its one contribution to *P* by assuming one of its (X_1, X_2, X_3, …) aliases when requesting service from *P* (*P* would think it detected a potential 3-way exchange with a new peer). Instead of performing the Sybil attack, *X* could also *collude* with real peers and collectively perform this unfairness attack, i.e. *collectively consume disproportionately to their contribution*. Finally, if peers did not discard *all* (*n*–1) edges, a dishonest peer *D* could provide service only once to an honest peer *H* and take advantage of this ad infinitum: if the $H \rightarrow D$ edge, which would always be part of the dishonest peer's ring, were allowed by providers to persist, a combination of Sybil attacks by *D* (i.e. the addition of fake paths to the right of $H \rightarrow D$, such as $D \rightarrow D_1 \rightarrow D_2 \rightarrow \ldots$) and of ongoing random honest contributions by *H* (i.e. the addition of edge paths to the "left" of $H \rightarrow D$) would allow *D* to repeatedly exploit a single prior contribution.

Bootstrapping. This algorithm should not be used if *P* just joined the community because *P* would never be able to detect an exchange ring (as *P* still has no outgoing edges in *G*). It is therefore required of new peers to provide service at least once without searching for rings, as we also discuss in Section III-C.

Edge weights. In our model we assumed the edges in *G* had no weights (so far we only assumed a unique label for every edge). We can also associate a weight with each edge corresponding to the *amount* of resource consumed during the transaction that the labeled edge represents. In Section V we discuss ways for peers to determine the reciprocal amount of resource they should contribute during a transaction.

B. Hotspot system entities

We now present the entities that comprise our hotspot peering system:

Teams. We focus on a community of people organized into *teams* with each team managing a number of wireless hotspots. We want to encourage teams to contribute in proportion to their consumption. A team *contributes* when it provides Internet access to members of other teams through its own hotspots, and *consume*s when its members access hotspots belonging to other teams. There are three advantages to using teams and not individuals to represent our peering entities. First, households that participate in the community may include several members and it would be unreasonable to require all of them to set up a different personal hotspot in order to use the system. Second, teams can include people that are willing to contribute to the system *but can only set up hotspots in areas where demand is limited* (e.g., they reside in the outskirts of a city). Third, teams effectively lower the number of peers and increase each peer's total wireless coverage thereby increasing the probability of transactions between two peers. Even if two peers do not usually interact directly, the potential *n*-way exchange rings that connect them can be quite small.

Members and member certificates. We assume the leader of a team generates a unique public-private key pair for the team, keeps the private key secret, and uses it to sign *member certificates* for all team members. Members have their own unique public-private key pairs and their certificates bind a member's public key (PK) to the team's public key, i.e. *member cert.* $= \{member\ PK, team\ PK\}_{team\ signature}$

The member certificates are included in the proofs of prior transaction that consuming members sign, thereby creating a signature chain that shows, effectively, which *team* (which peer) is signing the proof. We assume that the team leader provides member certificates only to individuals that the team can trust implicitly. The reason for this is that it is relatively easy for over-consuming members to reduce the total consuming ability of the rest of their team. However, if something like this is detected, the honest members of a team can simply form a new team by creating a new team key pair and member certificates, and by excluding the dishonest member. This is easier than using expiration dates and renewing the certificates of honest members. In what follows, by *member* we will mean either the person or the wireless client the person is using, which must also contain the member certificate and private key.

Access Routers (ARs). Each team operates one or more *Access Routers* (ARs) that provide wireless Internet access to visiting members of other teams.

Receipts. When ARs provide access to roaming members, they request a *receipt*, i.e. a digitally signed proof of the transaction. These receipts are depicted in Figure 2, with field sizes in bytes, and assuming 1024-bit RSA keys. Receipts contain: i) the public key of the providing team, ii) the consuming member certificate, iii) a timestamp noting the start time of the wireless session, iv) a *weight* containing the amount of traffic the AR routed for the member, and v) the signature of the member, signing the above.

Team servers. Each team operates exactly one *team server*. This server is used to store state information (in the form of receipts, see III-C) and serves as a single point of contact for both the team's ARs and the team's members who will communicate with it over the Internet (see III-C).

In Figure 2 we depict all entities presented above as part of a very small system with only three teams.

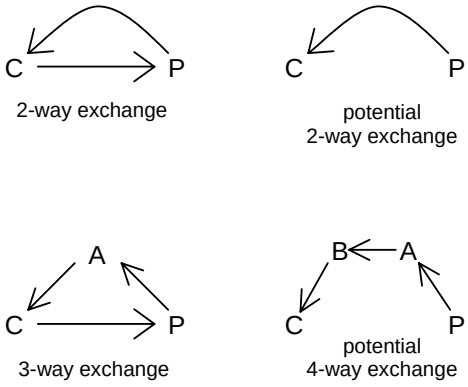

Fig. 1. Examples of *n*-way and potential *n*-way exchange rings. Arrows point from the consuming peer to the providing peer.

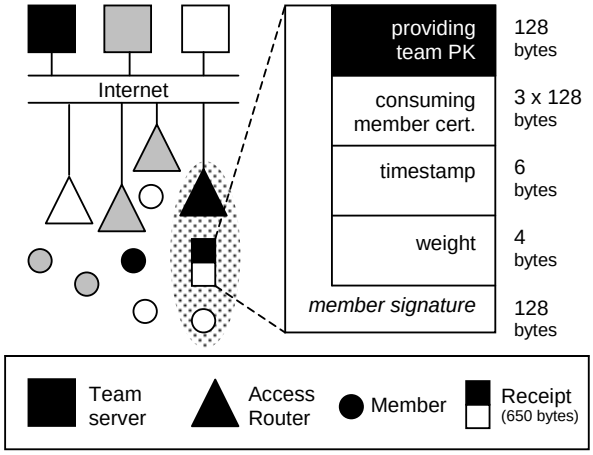

Fig. 2. A community with three teams (Black, Gray, and White). A member from the white team visits an AR of the black team and signs a (White → Black) receipt.

C. Decentralized fair peering

We present a decentralized protocol that uses the fair peering algorithm from Section III-A and is tailored to our hotspot-sharing community. An important requirement is to assume that all system entities involved are non-obedient and that they would perform an action only if it is in their interest. (We assume no module is tamperproof and that devices can be reprogrammed to deviate from our protocol.)

Receipt repositories. Every team server maintains three repositories: *IR*, *OR*, and *RR*, short for Incoming Receipts, Outgoing Receipts, and Random Receipts. *IR* contains receipts where the team is the provider. *OR* contains receipts that have been signed by a member of the team, i.e. where the team is the consumer. *RR* contains receipts that are neither incoming nor outgoing, i.e. they represent transactions between other pairs of teams. Their sizes (maximum numbers of receipts they can store) are respectively s_{IR}, s_{OR}, and s_{RR}. In addition, each team server maintains a *DR* repository, short for Discarded Receipts. *DR* stores receipts that were discarded by the fair peering algorithm and should be ignored. *DR* does not need to store entire receipts because hash values calculated using the {*providing team PK*, *consuming member cert.*, *timestamp*} fields of a receipt are enough to serve as a unique receipt identifier. *DR*'s size is s_{DR}. We assume that a team's ARs have read/write access to their server's repositories over the Internet.

Protocol

Step 1. Provider searches for exchange rings. We focus on a visitor *c* (lower case *c*, short for *consumer*), member of team *C*, who wants to access an AR belonging to team *P* (short for *provider*). According to the fair peering algorithm, *P* should provide service to *C* if a potential *n*-way exchange ring could connect *P* to *C*. The ring should not contain receipts that have been discarded by *P*. Without having access to all of the receipts in the system, all *P can* do (see Discussion below for our reasoning here) is to search for an exchange ring in *C*'s and *P*'s combined repositories, specifically in *P.OR* and *P.RR*, and in *C.IR* and *C.RR* (see Figure 3). Note that we will use the "dot" notation from now own to denote a team's repositories. How can *P* access *C.IR* and *C.RR*? We propose that it is in the interest of member *c* to bring copies of these two repositories and to present them to *P* (along with *c*'s member certificate). This is easy if *c* is not required to have the most current version of *C.IR* and *C.RR*. With 1024-bit keys, receipts are 650 bytes long. Assuming $s_{IR} = s_{RR} = 100$, the two repositories can fit in a 127 KB file. Members can opportunistically contact their team server and obtain copies of these repositories when, for example, they connect to the Internet through this system, or through another system such as GPRS.

Step 2. Provider discards receipts and deletes outgoing receipt. If *P* detects that a potential *n*-way exchange ring could connect it to *C*, *c* will be admitted. *P* would then discard (by storing their hashes in *P.DR*) the (*n*–2) receipts from the ring that do not involve *P*. According to our definition of the potential *n*-way exchange ring (Section III-A), the *n*'th receipt that could complete the ring is the *potential C* → *P* receipt that has not yet been signed by the consuming member. *P*'s *outgoing* receipt that was part of the ring can simply be deleted from *P.OR* without storing a hash value for it in *P.DR*. (Note that deleting would not be advisable for receipts that were part of the ring but came from *P.RR*, because *RR* is also sent to a team's roaming members. Remember that a receipt is only discarded locally to avoid unfairness attacks. It is still reasonable to send locally discarded receipts to the team's members since the teams that they will visit may not have discarded them. As there is no point in sending a team's *OR* to members because the receipts in there would not help in the search for potential *n*-way exchange rings, discarded receipts from *OR* can simply be deleted from *OR*. As a consequence, when temporarily combining *C.IR* and *C.RR* with *P.RR* and *P.OR*, an important constraint is that the outgoing receipt of *P* that is part of the potential *n*-way exchange ring must come from *P.OR* and not from *C.IR* or *C.RR* – this way *P* can be sure that it is not considering an outgoing receipt that it had deleted from its *P.OR* previously.)

Step 3. Provider updates Time Horizon. If the *DR* size limit is reached, the entry corresponding to the receipt with the oldest timestamp is deleted. Because this way the server can *forget which receipts it should ignore*, the server also sets a *Time Horizon* variable equal to the timestamp of the receipt whose entry was just evicted from *DR*. Servers consult their time horizons while searching for exchange rings in

order to be sure that they do not consider receipts that they had discarded in the past, at the cost of ignoring potentially many (non-discarded) old receipts (see Discussion below for our reasoning here).

Step 4. Consumer signs a new (provider-incoming and consumer-outgoing) receipt. Provider also updates the Random Receipts repository. If *c* is admitted, *P* will eventually store in *P.IR* a new receipt ($C \rightarrow P$) that completes the *n*-way exchange ring, and which is generated according to a procedure detailed in the following subsection (III-D). Member *c* must then send this receipt to *c*'s own team server to be stored in the *C.OR* repository. This can happen opportunistically. For example, when *c* receives updates of *C.IR* and *C.RR*, *c* can also send all unreported outgoing receipts. Finally, *P* updates its *P.RR* repository with some of the receipts that *c* presented (but only if *c* was successfully admitted). More specifically, each receipt in *C.IR* and *C.RR* is considered for addition to *P.RR*, irrespective of whether *P* should locally ignore it based on *P's* time horizon and *P.DR*. The only rule is that any new receipt that *P* adds to *P.RR* should not involve *P* as consumer or provider (*P* already has those in *P.OR* or *P.IR*) and if during this update *P*'s s_{RR} is reached, *P* would remove from *P.RR* the receipt with the oldest timestamp.

Bootstrapping. As is required by the fair peering procedure, a newly joined peer must provide service to at least one other peer without searching for exchange rings (at the cost of potentially servicing cheaters) in order to have a non-zero chance of being serviced by peers that do search for exchange rings. We say that a newly joined team operates in *Uncontrolled Access Mode* if its ARs admit all community members without searching for exchange rings. At the same time, these teams start filling their *IR* repository with the receipts they earn, and their *RR* repository with the receipts that their visitors bring them. We define a parameter called a team's *patience*: if, after trying to be granted access (and probably failing several times), the new team is granted access to foreign hotspots for a total of *patience* times, the team switches to *Controlled Access Mode* and starts to admit visitors only after searching for exchange rings. The intuition behind the *patience* parameter is that the number of successful admissions is a simple-to-measure proxy for how potentially "well connected" the team has become.

Receipt weights. The basic version of the fair peering algorithm does not consider weights on the graph edges, but in Section V we will discuss ways to determine the amount of reciprocal contribution per transaction using the weights of the receipts in the potential *n*-way exchange ring.

Discussion

We discuss here some of the incentive issues that arise during protocol execution. The main objective of the protocol is to detect potential *n*-way exchange rings. If *P* does not follow the protocol and admits cheaters that present no receipts (or present only fake receipts), these receipts will not be useful when *P*'s members try to access teams that do follow the protocol. Ideally, *P* would want to be able to access all system receipts at zero cost. However, other teams have a *disincentive* to share their receipts because *P*

may eventually discard some of these receipts, which would cost the other teams directly in possible future visits by their members to *P*. Only *C* has a clear incentive to assist *P* in considering as many receipts as possible when *P* is searching for a ring, and the best *C* can do is to show *P* its *C.IR* and *C.RR* repositories. Even though some of the receipts in there could also be eventually discarded, *C* wants to show *P* as many receipts as possible because *C* cannot know which receipts *P* has already discarded. As an important side effect, *P* gains potentially useful receipts for its *P.RR* repository, and of course, the new $C \rightarrow P$ receipt for its *IR* repository. However, to be sure that the *P.RR* repository is not filled with fake receipts, *P* updates *P.RR* with receipts from *C.IR* and *C.RR* only if *c* is successfully admitted.

So far we have assumed that the more recent a receipt is (assuming it is not a fake) the more valuable it is, and we perform all repository receipt replacements based on this assumption. The intuition behind this is that the more recent a receipt, the less time it had to circulate through the system, and therefore the smaller the probability of having being discarded by other servers (by being marked in their *DR*s or by being older than their time horizons). As a side effect, teams are induced to contribute continually in order to refresh their repositories and obtain receipts that are newer than most time horizons. If a team were at some point to simply turn off its ARs (turn *traitor*, see IV-C), its receipts would be "worth" less and less as time went on, and eventually the team would not be able to be serviced. It would then have to go through the equivalent of the bootstrap phase once more (of course, for certain teams this may be acceptable, especially if they only consume from the system in small bursts that are very far apart and whose timing can be predicted by the team in advance of the burst).

On a final note, assuming complete intra-team trust (we will revisit this assumption in Section V), the consuming members *want* to send their new outgoing receipts to their team server, since outgoing receipts are required when providers search for rings while following the fair peering algorithm – which excludes cheaters, admits contributors, and earns providers useful receipts in the process.

D. Wireless session details

Detecting Access Routers. A roaming member detects the presence of a nearby AR by listening for link-layer beacons that ARs broadcast. We assume that the link-layer identity of the wireless network (e.g. the SSID in IEEE 802.11) includes a well-known tag used by all community ARs. An access point that is not part of the peering community has nothing to gain by pretending to be one: community members view all foreign ARs with distrust (even ARs belonging to the community) and security-conscious roamers would first tunnel to a trusted proxy, such as their team server, before accessing the Internet in any case.

Access control. After the link-layer connection between AR and member is established, the member can start sending and receiving IP packets. Each AR hands out dynamic IP addresses to members and performs Network Address Translation (NAT) in order to be able to support many concurrent users.

Before the protocol described in III-C terminates, the AR only allows connections to a local access control port. Only if a member is admitted does the AR allow Internet traffic to pass.

Signing receipts. Periodically, ARs broadcast their team's public key. This is a signal for connected visitors to sign a receipt and send it to the AR. Visitors are required to do this if they wish to continue their wireless session – the AR would block the IP address of a member that does not return a signed receipt within a time interval. To produce a signed receipt, roaming members use i) the contributing team's public key and then further add ii) their own member certificate, iii) a timestamp that is identical for all receipts signed by the member during the wireless session, and is equal to the session's start time, iv) the weight of the receipt, which is equal to the amount of traffic that the AR has forwarded for the member so far (we assume that members can keep track of the amount of traffic they send and receive), and v) their signature, signing all of the above.

Each time the AR receives a receipt from a member it verifies that i) the timestamp value is correct (we assume that ARs and members are loosely synchronized) and equal to that session's start time, ii) the weight on the receipt is in agreement with the AR's own measurements, and iii) the member certificate and signature are in agreement with the certificate presented during access control. As a side effect of these verifications, the AR can know that an outside attacker has not hijacked the wireless session because an outsider cannot be in possession of the member's private signing key.

Sessions end after members stop signing receipts or after one of the verifications mentioned above fails, in which case the AR blocks the relevant IP address and returns it to the address pool. Out of all the receipts received from a member during a session, the contributing team will only store the latest one, as other teams would not accept as evidence of contribution more than one receipt from the same member with the same timestamp (discarded receipts are marked using a hash of the {*providing team PK*, *consuming member cert*, *timestamp*} fields which does not include the weight). See also Figure 4.

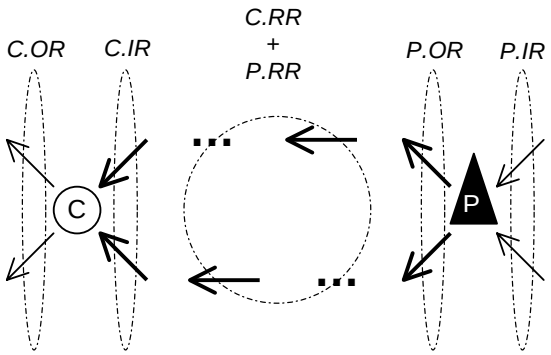

Fig. 3. *P* is searching for a potential *n*-way exchange with *C* after temporarily combining *C.IR* and *C.RR* with *P*'s own *P.RR* and *P.OR*.

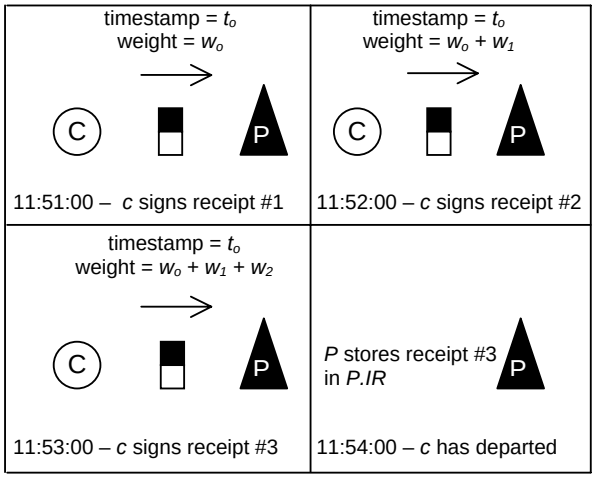

Fig. 4. Receipt signing: assume member *c* is admitted at 11:50:30, AR broadcasts receipt requests once every minute, and *c* departs at 11:53:30.

IV. EVALUATION

A. Simulator

We programmed our own custom simulator in Java in order to perform the experiments that we describe in the next subsections. All four variants of the simulator, one for each type of experiment, can be downloaded from `http://mm.aueb.gr/~efstath/fairpeering/simulators.zip`.

Each simulation consists of *rounds* during which every team interacts exactly once with another team (i.e. exactly one member of the consuming team visits an AR of the providing team) chosen uniformly at random from the population of teams. We also simulate the evolution of the system by allowing one new team to join in after every round, up to a possible of N total teams. In the experiments we use 300 and 1000 for N. The simulation begins at round 2 when two teams (teams 1 and 2) are in the system. Therefore, for the rounds up to round N, the round number is equal to the number of participating teams, and teams are identified by the order in which they joined. All teams share the same values for s_{IR}, s_{OR}, s_{RR}, and s_{DR} (the repository sizes), and we fix s_{OR} to 400, which is large enough never to be reached, as teams also *delete* receipts from their *OR* repositories. (Remember the contents of *OR* are not sent to roaming members so teams can afford to have a relatively large *OR* repository.)

B. Failure rates

We wanted to see how having only a partial view of the receipt graph affects teams. In the following experiments we consider a community where all teams are following the protocol honestly. We observe that sometimes exchange rings cannot be detected even though the requesting team is not free-riding. This is the central inherent disadvantage of our protocol: by being strict, it sometimes excludes honest collaborators. However, we show that there is usually less than 1% chance of this happening. More specifically, in Figure 5 we depict the *mean failures per visit*, i.e. the rate at which a requesting member will be denied service. We plot results for three different values of the *patience* parameter (see Section III-C) and we set $s_{IR} = s_{RR} = 100$, $s_{DR} = 10$, and $N = 300$. For the curves in Figure 6, we set $N = 1000$.

In both Figures 5 and 6 we see that systems with 300 and 1000 teams can maintain a failure rate of less than 0.01, which we feel is quite acceptable. Observe that the mean rate tends to increase when teams are still joining as new teams fail because they are still new (until round 300 and round 1000 in Figure 5 and 6 respectively), but then shows a decreasing trend. More *patience* when teams are first joining helps keep overall failure rates low. We also assume that if a team is ever denied service it decides to go through its bootstrap phase with the same *patience*, thinking that the system has “forgotten them” – otherwise its failure rate can slowly increase over time. So the meta-rule here is that a team should follow the algorithm but also turn to Uncontrolled Access Mode for a few visitors every time the system causes an injustice, which for collaborating teams can be as low as once every thousand requests.

In Figure 7 we focus on the average size of the exchange rings after 1000 rounds of two experiments with *patience* = 10, N = 300, $s_{IR} = s_{RR}$ = 100. Most exchanges are 3-way exchanges. In our experiments we detected even up to 20-way exchanges (ring sizes above 6 are extremely rare though).

C. Punishing treason

In Figure 8 we turn our attention to *traitors*, i.e. teams that stop contributing after a period of collaboration, but who continue to consume. For two experiments with *patience* = 10, N = 300, $s_{IR} = s_{RR}$ = 100, we let team 1 turn traitor starting from round 301. We observe how quickly the traitor is "identified" when asking for service, by plotting the traitor's failure rate compared to the average failure rate of the others. Quickly, it approaches 100%. However, because there are always some teams operating in Uncontrolled Access Mode (more if s_{DR} = 1, which causes time horizons to be closer to the present and consequently causes more failures in detecting rings which in turn cause more bootstraps according to the meta-rule we described above) a traitor would still, on rare occasion, receive service, even though the receipts that involved the traitor were purged from the system long before.

D. Punishing extravagance

Similarly to Figure 8, in Figure 9 we plot two experiments with the same parameters as above (and we fix s_{DR} = 1) where team 1 turns *extravagant* from round 301, i.e. it consumes (visits) *extravagance* more times than the other teams. An extravagant team can consist of too many regular members or a few extravagant members. The failure rates for 10- and 100-times extravagant teams can approach respectively 20% and 80% compared to the usual <1%.

V. Discussion

We will close our proposal with a discussion of five remaining issues not covered in detail so far.

Intra-team trust. So far we assumed that teams are relatively small and that intra-team trust is absolute. Through the power of receipt-signing, a dishonest member could become *extravagant* (see IV-D) and harm the other team members, or start signing fake receipts in collusion with other teams or in collusion with free-riders that want to appear to be offering service (and, e.g., pay the member in return). However, there is still hope: the protocol requires that all members send every receipt they sign to their team server. There is no way to *force* members to do this. If a member *does* send the receipt, the team knows where, when, and how much the member consumed, and we assume that strong social control (and no anonymity within the team) is enough to stop extravagant members. As a last resort, the team can create a new identity for itself, sign new member certificates, and exclude the extravagant member. Even if a member *does not* send an outgoing receipt home for fear of consequences, the member cannot control how the receipt will circulate through the system, so it can possibly arrive to the team server via a random visitor's

RR repository. If the provider detects an outgoing receipt it has not seen before this way (let us assume it can store *hashes* of previously deleted outgoing receipts for a reasonable amount of time), this can mean the equivalent of "instant exclusion" for the dishonest member and the forming of a new team. Note that we discussed (in III-C, Step 2) that a provider is normally confident that its own outgoing receipts are in its *OR* (or not, but only if they were deleted), and therefore visitors may have no incentive to show this type of receipts. Still, there is a non-zero chance of a visitor showing such a receipt to the provider *if the visitor does not still know the identity of the provider* – which is indeed the case the first time a visitor visits a new location and the local AR has not yet sent a broadcast of its public key asking for receipts, but has started the admission control procedure (mainly requesting and viewing the repositories of the visitor). As random visitors are possibly indifferent about helping a provider to punish its own members, we can assume that the above scenario may indeed happen. The fact that, after all, intra-team trust is not absolute is the reason the team leader does not hand out the team's private signing key to members, but instead signs them a member certificate.

Simulation. Eventually the protocol will have to be simulated with realistic user mobility models. However, even if the algorithm is never able to connect a visitor from Europe to a hotspot in America, being able to be identified within one's own city certainly covers the bulk of user expectations. (Furthermore, we also need to simulate the effect of opportunistic synchronization between a team's server and members.)

Implementation. We believe it is easy to supply an open specification for this peering protocol that can be implemented in access points and in wireless card drivers. Indeed, we have started our own prototype implementation on top of the Linux-based Linksys WRT54GS router. Furthermore, team servers could actually be implemented as distributed entities running on top of a team's ARs. (Our protocol was designed with specifically such resource-constrained embedded environments in mind.)

Denial-of-Service. We focus on two system-specific DoS attacks: i) *ARs that are asking members to sign a lot of receipts*. Such an irrational AR would quickly "run out of business" and directly harm its team. ii) *Members that keep re-logging-in to avoid signing the first receipt.* The receipts the provider would discard every time it searches for a new exchange ring would harm the member's team (and, also, ARs can delay all logins).

Weights. Instead of searching for a single ring, the AR could run a max-flow algorithm on the receipt graph and see how much traffic it "owes" to the visitor. It is important to consider only the minimum weight on each ring to avoid unfairness attacks, but it would be necessary for providers to perhaps offer double or triple the amount in order to keep receipt weights from slowly dwindling to zero (which could happen if every new receipt has a smaller weight than the minimum of several others). Much more simply, assuming it is not very costly for ARs to let sessions continue for a long time, weights could simply be used as indications by ARs to see if visitors come from teams that are exhibiting constant frugality in their contributions.

VI. Conclusions

In conclusion, we believe that most of the ideas presented here are workable. It is difficult, however, to predict all possible attacks against a fully self-organized incentive mechanism, and it is even more

difficult to accurately model the incentives of autonomous peers who may not fall neatly into *rational* or *irrational*. Perhaps all we can do with a system like peer-to-peer hotspot sharing is to design it as simple as possible and hope for the best. Although traditional economies are centralized with good reason, it is an interesting experiment to see if full self-organization and exchange economies are directly applicable to certain domains, especially where traditional structures seem to fail, as in the case of wireless hotspot roaming where the players are potentially too numerous "to be brought together under one roof."

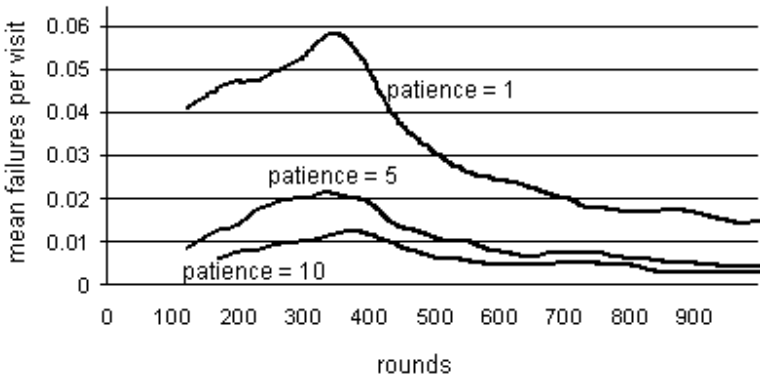

Fig. 5. Failure rates with 300 honest teams.

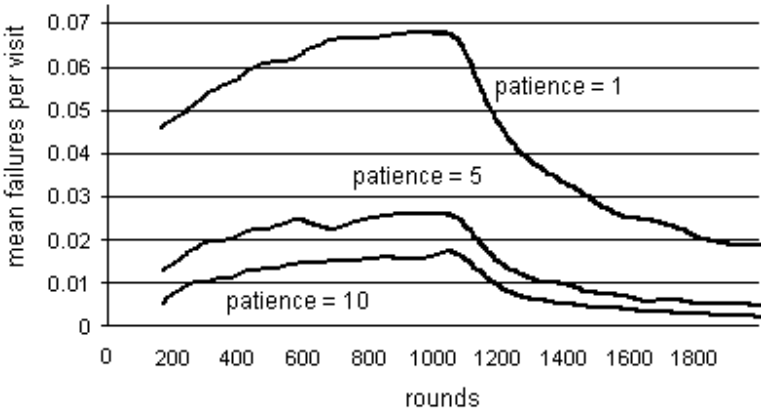

Fig. 6. Failure rates with 1000 honest teams.

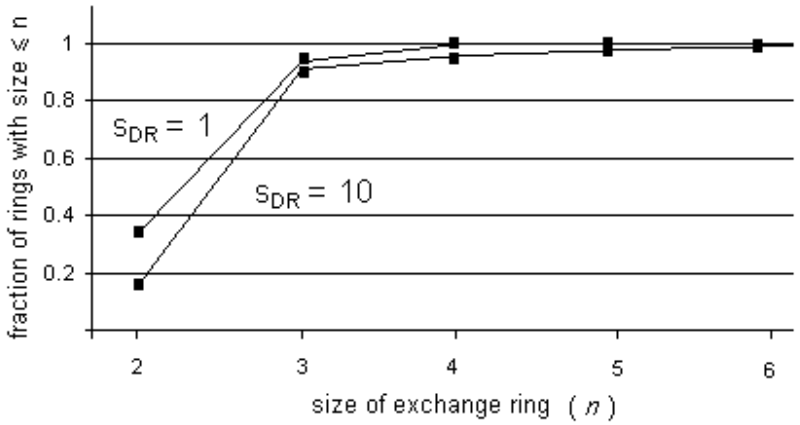

Fig. 7. Observing the sizes of exchange rings.

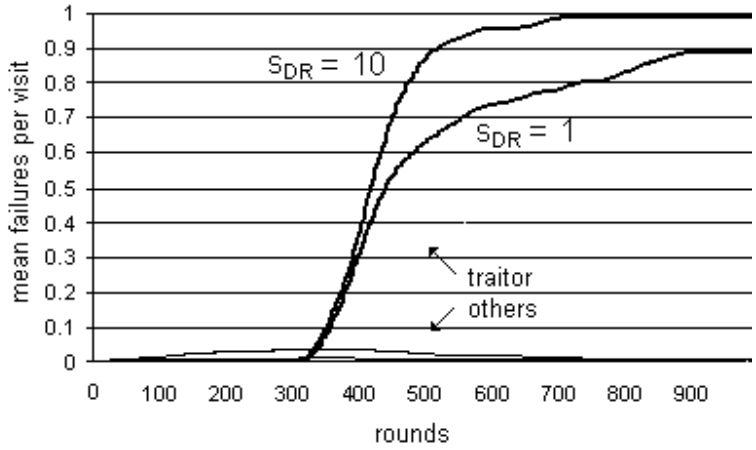

Fig. 8. Failure rates for a team that turns traitor (at round 301), compared to that of other teams.

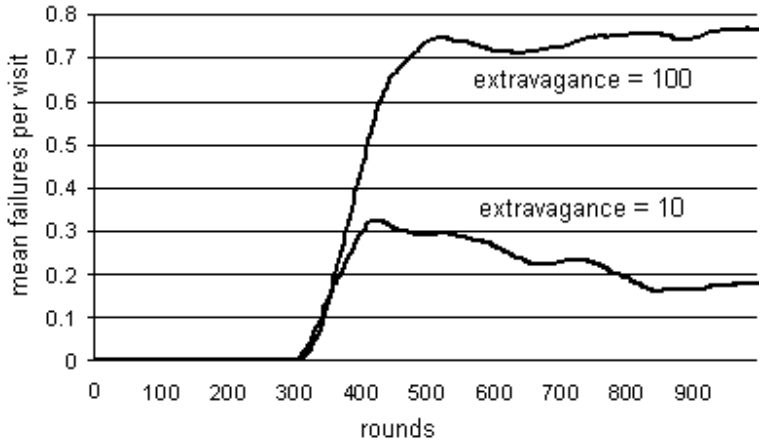

Fig. 9. Failure rates for extravagant teams.

REFERENCES

[1] ANAGNOSTAKIS, K. G., AND GREENWALD, M. B. Exchange-based Incentive Mechanisms for Peer-to-Peer File Sharing. In *Proceedings of ICDCS'04: 24th IEEE International Conference on Distributed Computing* (2004).

[2] BEN SALEM, N., HUBAUX, J.-P., AND JAKOBSSON, M. Fuelling WiFi Deployment: A Reputation-based Solution. In *Proceedings of WiOpt'04: Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks* (2004).

[3] BUTTYAN, L., AND HUBAUX, J.-P. Stimulating Cooperation in Self-Organizing Mobile Ad Hoc Networks. In *ACM/Kluwer Mobile Networks and Applications*, 8(5), (2003).

[4] CAPKUN, S., BUTTYAN, L., AND HUBAUX, J.-P. Self-Organized Public-Key Management for Mobile Ad Hoc Networks. In *IEEE Transactions on Mobile Computing*, 2(1), (2003).

[5] COX, L. P., AND NOBLE, B. D. Samsara: Honor Among Thieves in Peer-to-Peer Storage. In *Proceedings of SOSP'03: ACM Symposium on Operating Systems Principles* (2003).

[6] DOUCEUR J. R., The Sybil Attack. In *Proceedings of IPTPS'02: First International Workshop on Peer-to-Peer Systems* (2002).

[7] EFSTATHIOU, E. C., AND POLYZOS, G. C. A Peer-to-Peer Approach to Wireless LAN Roaming. In *Proceedings of WMASH'03: First ACM International Workshop on Wireless Mobile Applications and Services on Wireless LAN Hotspots* (2003).

[8] EFSTATHIOU, E. C., AND POLYZOS, G. C. Trustworthy Accounting for Wireless LAN Sharing Communities. In *Proceedings of EuroPKI'04: First European PKI Workshop – Research and Applications, Springer LNCS 3093* (2004).

[9] FELDMAN, M., LAI, K., STOICA, I., AND CHUANG, J. Robust Incentive Techniques for Peer-to-Peer Networks. In *Proceedings of EC'04: Fifth ACM Conference on Electronic Commerce* (2004).

[10] FELDMAN, M., PAPADIMITRIOU, C., CHUANG, J., AND STOICA, I. Free-Riding and Whitewashing in Peer-to-Peer Systems. In *Proceedings of ACM SIGCOMM'04 PINS Workshop: Practice and Theory of Incentives in Networked Systems* (2004).

[11] LEE, S., SHERWOOD, R., AND BHATTACHARJEE, B. Cooperative Peer Groups in NICE. In *Proceedings of IEEE INFOCOM* (2003).

[12] MAHAJAN, R., RODRIG, M., WETHERALL, D., AND ZAHORJAN, J. Experiences Applying Game Theory to System Design. In *Proceedings of ACM SIGCOMM'04 PINS Workshop: Practice and Theory of Incentives in Networked Systems* (2004).

[13] NGAN, T.-W., J., WALLACH D. S., AND DRUSCHEL, P. Enforcing Fair Sharing of Peer-to-Peer Resources. In *Proceedings of IPTPS'03: Second International Workshop on Peer-to-Peer Systems* (2003).

[14] VISHNUMURTHY, V., CHANDRAKUMAR, S., AND SIRER, E. G. KARMA: A Secure Economic Framework for P2P Resource Sharing. In *Proceedings of the First Workshop on Economics of Peer-to-Peer Systems* (2002).

[15] YANG, B., AND GARCIA-MOLINA, H. PPay: Micropayments for Peer-to-Peer Systems. In *Proceedings of CCS* '03*: Tenth ACM Conference on Computer and Communication Security* (2003).

[16] Boingo Wireless Inc. `http://www.boingo.com`

[17] Free Networks.org association. `http://www.freenetworks.org`

[18] iPass Inc. `http://www.ipass.com`

[19] Kazaa by Sharman Networks. `http://www.kazaa.com`

[20] Kazaa Hack. `http://www.khack.com`

[21] Linspot by Biontrix b.v.b.a. `http://www.linspot.com`

[22] Netshare by Speakeasy Inc. `http://www.speakeasy.net/netshare/`