Trustworthy Accounting for Wireless LAN Sharing Communities

Elias C. Efstathiou and George C. Polyzos

Department of Computer Science
Athens University of Economics and Business
Athens 104 34, Greece
{efstath, polyzos}@aueb.gr

Abstract. The Peer-to-Peer Wireless Network Confederation (P2PWNC) is designed to be an open self-organizing Wireless LAN (WLAN) roaming association where users from one WLAN can access all other WLANs belonging to the same P2PWNC. Unlike other WLAN roaming schemes, P2PWNC is open to all types of WLANs and particularly to residential networks owned by individual householders. Without an identity certifying authority, trustworthy accounting of transactions in the P2PWNC is challenging, but accounting is necessary in order to enforce the basic P2PWNC 'rule of reciprocity'. We show that even though the P2PWNC accounting mechanism and its purpose-built Public Key Infrastructure are open to Sybil attacks, there exists a user authentication algorithm that excludes all free riders and that can also make the percentage of unfair exclusions it causes very small simply by using more system memory.

Keywords. self-organized security, WLAN roaming, Sybil attack, peer-to-peer

1 Introduction

1.1 Wireless LAN Roaming Today

Wireless LAN (WLAN) technology based on the IEEE 802.11 family of standards is a success. Newer laptops and PDAs are commonly WLAN-enabled and 802.11 radios are also becoming part of consumer electronics devices such as digital cameras, MP3 players, and cellular phones [1]. Nevertheless, the economic value of these devices is greatly reduced because WLAN coverage is still not ubiquitous. This is unfortunate because it is relatively easy to set up a WLAN 'hotspot', i.e., an area where wireless Internet connectivity is provided: one only requires a WLAN access point / router and a broadband Internet connection, both of which are extremely small investments nowadays. However, the lack of universal roaming standards has resulted in a fragmented market of commercial public WLAN operators. Without roaming support, operators can provide only limited coverage, which makes it difficult to attract new customers and to fund additional investments in infrastructure. On the non-commercial

side, the majority of residential, business, and university hotspots are closed to outsiders, and the various free hotspots available in several cities cover only a small number of locations. Efforts are underway to establish WLAN roaming standards that would allow customers from one operator to access hotspots belonging to other operators. These efforts focus both on the technical [2] and on the roaming settlement issues [3]. In parallel, several *hotspot aggregators* [4, 5] are attempting to unite smaller operators by coordinating AAA and roaming settlement. These efforts still result in relatively 'spotty' coverage even for the largest of aggregators. Also, these solutions focus on commercial hotspot operators and are inappropriate, for example, for residential hotspots owned by individual householders.

1.2 A Peer-to-Peer Approach to Wireless LAN Roaming

In [6] we proposed a novel cooperative application called *Peer-to-Peer Wireless Network Confederation* (P2PWNC). P2PWNC is a system designed to unite independent hotspots in a WLAN sharing community where users that are associated with one hotspot can also access all other community hotspots. In P2PWNC, participants own and operate their local WLANs, but they also share them with roaming P2PWNC visitors that happen to be in range, much like a file-sharing community shares files. In principle, P2PWNC can incorporate various types of WLAN operators but its focus is on residential hotspots owned by individual households in an urban setting. With sufficient hotspots, built with independent and small investment decisions, we envisage citywide infrastructures through which P2PWNC participants would enjoy ubiquitous wireless Internet access. In [6], a number of issues involved with building a P2PWNC were presented and the peer-to-peer approach to roaming, where participants manage the infrastructure without relying on central authorities, was compared to existing roaming schemes and was presented as a simpler alternative to designs that attempt to replicate the complexity of cellular roaming.

The distinctive characteristics of P2PWNC are: (a) P2PWNC is completely self-organizing, requiring no central authorities, unlike the hotspot aggregation model which requires an aggregator; (b) P2PWNC is open to all hotspots that are willing to participate, unlike operator-only roaming associations that require legally binding roaming agreements; (c) joining P2PWNC is similar to joining a file-sharing community: there is no real-world communication or administrative overhead and only local actions (setting up a hotspot managed by a local software agent) are required; and (d) in the P2PWNC 'market' the only currency is payment in kind: as long as a participating hotspot is open to P2PWNC visitors, its owners can access other P2PWNC hotspots when roaming. We will call this last characteristic the *rule of reciprocity*.

In [7], P2PWNC was presented in economic terms as a *mechanism design* problem. There, the P2PWNC rules were formalized and, by using an analytic model, it was shown that as the number of P2PWNC participants becomes large, the optimal contribution policy may be approximated by a simple rule: any participant wishing to roam must be contributing a fixed amount of resources to visitors. This could be a fixed amount of Internet bandwidth in the local hotspot that is reserved for visitors. Enforcing this rule means participants that do not follow it must be *excluded* from consuming community resources.

1.3 Contributions of this Paper

In this paper we show that the P2PWNC reciprocity rule can be enforced and that free riders[1] can be excluded in a secure way even when all P2PWNC participants have equivalent roles and no authority controls system operation, not even during node initialization when new hotspots join P2PWNC. We assume that adversarial participants can tamper with P2PWNC components in an attempt to deviate from P2PWNC protocols. We assume that the main goal of an adversary is to attempt to benefit from the P2PWNC without contributing, i.e., by not operating a P2PWNC hotspot at all or by keeping the hotspot closed to visitors. We will also assume, however, that standard cryptographic assumptions hold, and that all private keys remain private.

We base our solution for trustworthy accounting of P2PWNC actions on a purpose-built Public Key Infrastructure (PKI) with no centralized Certification Authority (CA). We rely on signed receipts for accounting P2PWNC transactions, which are stored in a *decentralized storage subsystem*. We define a universal system parameter, the Time-to-Live (TTL), after which receipts expire and can be deleted from the system: the TTL is therefore a measure of the system's memory.

We claim that a P2PWNC would exhibit a *small world phenomenon* [9] and with the help of simulations we show how the fundamental *Sybil attack*[2] problem can be circumvented via a P2PWNC-specific authentication algorithm. We link the TTL to system performance, which we express as the number of *unfair exclusions* from consuming community resources. We show that the number of unfair exclusions can become very small and that there exists a TTL value above which there is no appreciable gain in system performance. Finally, we motivate future work on the P2PWNC by showing how our authentication algorithm can be made efficient.

2 System Model

2.1 Entities and Identities, Trust and Authentication

In P2PWNC, WLANs are referred to as *administrative domains*, or simply *domains*. Each domain has an associated set of *registered users* (e.g., for residential hotspots the registered users are the household members). The registered users, when accessing the Internet from their *home domain*, are considered local users and P2PWNC is not involved in their actions. When, however, *roaming users* access WLAN services from the *visited domain* within which they happen to roam, a P2PWNC *transaction* takes place. The visited P2PWNC domain is the *provider* and the home P2PWNC domain of the roaming user is the *consumer* in this transaction.

---

[1] Free riding: Consuming community resources without contributing, a problem analyzed in the context of peer-to-peer systems in [8].

[2] Sybil attack [10]: A fundamental problem in open self-organizing systems with no identity-certifying authorities. An entity can appear in the system with multiple identities and invalidate any number of system assumptions.

We define *altruistic domains* and say that a domain acts altruistically if it allows visiting users to access the WLAN without checking if their home domain follows the reciprocity rule. We say that a domain is *unfairly excluded* when, although it does follow the reciprocity rule, one of its registered users is denied access by a visited domain. The reciprocity rule states that participants must make local resources available for consumption by other participants.

A P2PWNC *Domain Agent* (DA) represents each domain. P2PWNC is a peer-to-peer network of DAs that communicate over the Internet. DAs are software programs that can run on various types of physical hosts. For residential hotspots, we can assume that the DA runs on the wireless router itself and that it accesses the Internet (and other DAs) over a broadband connection. DAs automatically (a) regulate WLAN service provision to visitors by enforcing the reciprocity rule, and (b) form P2PWNC's decentralized storage subsystem. The DA network is completely self-organizing and the network is always open to new DAs (i.e., new WLANs).

We assume that each DA, upon initialization, generates a unique public-private key pair and the DA's public key serves as the domain's P2PWNC name. Users of domains also generate their own key pairs and their public key serves as their P2PWNC username. Users store in their portable devices, alongside their private key, *user certificates* that are signed by their home domain's private key. User certificates bind a username to a home domain name; their format is depicted in Fig. 1.

The user certificates encode only a very basic trust model. In the absence of a central authority and with the requirement for an open system, P2PWNC and its trust model is more similar to open ad hoc networks than to traditional roaming infrastructures where domains must know each other's identity and address beforehand. In P2PWNC (a) domains may not know of one another beforehand; (b) any entity may claim to be a new domain or a user of that domain simply by generating appropriately formatted names (i.e., keys) and certificates; and (b) domains could have an arbitrary number of identities. For our analysis we assume that domains do not trust each other, roaming users and visited domains do not trust each other, and home domains only trust their users for a limited time, which is encoded in the expiration date of the user certificate. We relax the definition of *successful authentication* to mean only that a visited domain has somehow established (a) that a roaming user is associated with a home domain and (b) that the user's home domain follows the reciprocity rule.

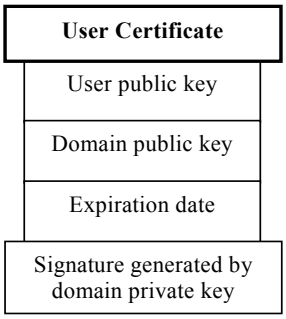

Fig. 1. P2PWNC user certificates. The public keys also serve as usernames.

2.2 Signed Receipts

When visiting users receive service from a visited domain they sign receipts that they give to the provider. Each receipt contains:

- a copy of the user's certificate;
- the name of the visited domain;
- a timestamp;
- service-specific information such as the volume of traffic that the visited domain routed for the roaming user;
- the user's signature, signing all the above.

A receipt encodes a transaction: the home domain represents the consumer and the visited domain represents the provider in this transaction. These receipts are public information and are used as input to the authentication algorithm that domains execute.

We also define the *Time-to-Live* (TTL), a universal system parameter that represents a time interval after which receipts expire.

All interested parties may verify that the user private key that corresponds to the user public key found in a receipt (as part of the user certificate) has signed the receipt. Also, they can verify that the domain private key that corresponds to the domain public key that is part of the user certificate has signed the user certificate. In essence, through one level of indirection (the user certificate), interested parties may verify the two domains that were involved in the transaction that a receipt encodes.

2.3 Disconnected Operation and Decentralized Storage

A central P2PWNC requirement discussed in [6] is that user authentication, as defined in Section 2.1, should proceed even if a user's home domain is offline. The certificates carried by a user are enough to prove his association with his home domain. However, the roaming user may not possess copies of all the receipts issued to his home domain, receipts that may be required to prove rule compliance. This is why copies of all receipts received by domains are stored in a decentralized storage subsystem where they remain available at least until they expire. For the following two sections, however, we can simply assume that all receipts received by domains are copied to an always-on logically centralized repository.

3 System Operation

3.1 Enforcing the Reciprocity Rule

So far, we have not really defined the 'reciprocity rule'. The model in [7] refers to a 'fixed-fee' type of rule. In P2PWNC, however, there is no straightforward way to check that a remote domain is constantly 'paying the fee', i.e., that it has an amount of bandwidth constantly reserved for visitors. In essence, we cannot check for availabil-

ity of bandwidth in a remote WLAN over the Internet without a trusted party at the remote WLAN that could verify this availability.

We therefore check for rule compliance indirectly: as long as there exist signed receipts stating that a specific domain provided service, we can assume that the domain did in fact provide service some time in the past (the exact time is part of the receipt). Three questions arise: (a) is the receipt 'fresh'? (b) Do we trust the signature on it? (c) To what amount of resource consumption does the receipt correspond to?

The first question is a matter of deciding on an appropriate TTL for receipts (we discuss the TTL further in Section 4). An ideal TTL would be small enough to allow receipts to be garbage-collected quickly and in the process give incentives to participants to provide anew, and large enough to allow the system to remember good providers even if for some reason they happened not to provide to visitors during a specific time frame. The second question (trusting the signature) also is the subject of Section 4. The third question we discuss below.

3.2 Signing and Storing Receipts

When a roaming user appears in a visited domain, he presents his user certificate in order to identify himself and his home domain. The visited domain also presents its own name / public key. The public keys included in the certificates are used to establish wireless session keys and secure further communication against eavesdroppers and hijackers. (This form of mutual authentication is susceptible to man-in-the-middle (MITM) attacks, but the MITM attack is irrelevant in the context of the P2PWNC trust model, in which roaming users do not trust their providers, and would probably tunnel all their traffic to a trusted Internet proxy in any case.)

For the following, we assume that the providing domain is either altruistic or that the roaming user has been authenticated successfully (as defined in Section 2.1).

The P2PWNC WLAN transaction is as a series of post-pay exchanges. In the beginning, the visited domain routes a limited volume of traffic for the visiting user. Then, the visited domain presents a receipt for this traffic and asks the user to sign it. If the user declines, no further service consumption is allowed. To avoid persistently dishonest users, the domain tracks user and device identifiers and attempts to increase the cost of dishonest behavior by delaying or denying suspect login attempts. If the user does sign the receipt and the domain confirms the signature via the user's public key, the user is allowed to continue.

At intervals specified by the visited DA, the domain presents new receipts and asks for the user's signature. In the *service-specific* receipt field the domain may account for any resource whose consumption the user can confirm as well (in order to agree to sign it). If, for example, the domain accounts for the volume of routed traffic, this field's value increases across receipts. All these receipts have the same timestamp (equal to the timestamp of the first receipt), a fact the user can confirm, and each one of these signed receipts makes the previous one obsolete: the visited domain will only store one receipt with a specific timestamp from a specific user, as other domains would not accept more than one such receipt during checks for rule compliance.

The visited domain stores all signed receipts in decentralized storage. For now, we can assume that this corresponds to a logically centralized, always-on server. Each

domain stores a list of receipts it received on this server. Other domains can access the list of receipts for any domain if they know the domain's name (public key) because the lists are placed in a hash table, keyed under a hash of the domain's name. This hash table is open for everyone to read. In Section 5 we distribute this hash table across domains. Expired receipts (according to the TTL) are deleted automatically.

4 Facing Sybil Attacks

4.1 The Problem and a Defence

The P2PWNC accounting mechanism described above is exposed to Sybil attacks [10]: an entity can generate any number of different domain identities since creating such identities only requires the generation of a key pair. The attacking entity can then generate any number of user certificates and associate them with any one of its domain identities. Then, the entity can use these user identities to sign fake receipts issued to one of its domain identities. Because receipts are used to check rule compliance, it would be trivial for an individual who does not even own a WLAN to consume without contributing: the individual need only create a fake identity for a home domain, issue a user certificate to himself, create any number of additional domain and user identities, and use these identities to sign receipts declaring that his (fake) home domain provided service to these identities. Because the system associates a Time-to-Live with receipts, the attacker would need to keep storing fresh receipts, but the cost of doing this may be significantly lower than operating a WLAN.

It is tempting to suggest that an analysis of the stored receipts by honest domains may alert them to suspicious patterns. This is indeed true. What we propose here, however, is that given the P2PWNC system model, *all* receipt patterns are suspect, except one. The objective of the authentication algorithm presented below is to detect this pattern, and only then to allow a P2PWNC visitor to login.

Consider a user *c*, associated with home domain *C*, visiting domain *P* (*C* is short for consumer, *P* for provider). In order to authenticate the user, *P* executes the breadth-first search algorithm below. We propose that this algorithm: (a) terminates eventually; (b) enforces the reciprocity rule; and (c) can result only in a very small percentage of unfair exclusions.

1. Add domain *C* to the `nodes_to_check` FIFO and to the `nodes_seen` set.
2. De-queue node from `nodes_to_check` and assign it to `current_node`.
3. Access the list of (live) receipts that show `current_node` as the providing domain. For every such receipt:
4. Check if the consuming domain on the receipt is in `nodes_seen`. If yes, get next receipt from list and repeat step 4. If consuming domain is *P*, `TERMINATE(SUCCESSFULLY)`. If not, add the consuming domain to the `nodes_seen` set and to the `nodes_to_check` queue.
5. If `nodes_to_check` is empty `TERMINATE(UNSUCCESSFULLY)` else go to 2.

This algorithm performs a breadth-first search on a tree of domains with *C* at the root. A successful termination of this algorithm is depicted in Fig. 2. This tree is a view of the greater directed graph of (live) system receipts. In the graph and in the tree, the nodes represent domains and the edges represent receipts. Edges start from the consuming domain and point to the providing domain. An edge exists if there is at least one (live) receipt connecting the two domains.

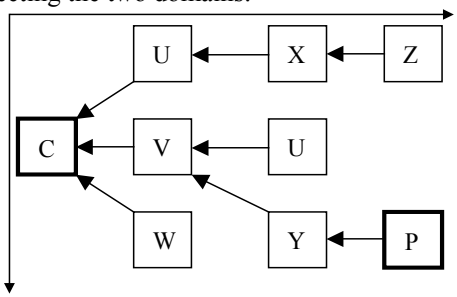

Fig. 2. A successful termination of the authentication algorithm. Domain *P* discovers itself in level 3 of the tree. Note that domain *U* appears more than once, but its child node, *X*, is accessed only once. The enclosing arrows depict the directions of the breadth-first search.

For simplifying the discussion below, assume that *P* and *C* are domains with only one registered user each, and that the identifiers *P* and *C* denote both the domain and its registered user, as meaning will be clear from context.

By following this algorithm, *P* attempts to find itself in a tree with *C* at the root and terminates successfully only if it succeeds in doing so. If *P* is found at the first level of this tree (the root is at level zero), then this means that *P* has (recently) received service from *C*. In such a case, according to the reciprocity rule, *P* should now provide service to *C*. If *P* is found at the second level of this tree, then this means that *P* has (recently) received service from somebody who (recently) received service from *C*, and so on. We propose that if this algorithm terminates successfully then there exists at least one real domain in the path connecting *P* to *C* (in addition to *P*) because *P* has signed a receipt for consuming resources, and assuming *P* trusts its roaming registered users, such a receipt can only be signed when a (real) visited domain provides resources to *P*. The identity of the domain that provided to *P* (*Y* in the example of Fig. 2) is irrelevant: what is important is that a domain that called itself *Y* provided service to *P*. Also, according to the stored receipts, *V* provided service to *Y* and *C* provided service to *V*. It would be in accordance with the reciprocity rule, now, for *P* to provide service to *C*. Even if *V*, *Y*, and *C* were all aliases of the same domain, which also means that the receipts connecting them are fake, *P* can trust that the last receipt, the one signed by *P* itself, is real, which means that *P* had received service from the *V*/*Y*/*C* domain and now has an opportunity to pay back.

If the algorithm terminates unsuccessfully, *P* exhausted the (live) receipts without discovering itself anywhere in the tree. This is bound to occur if there are a finite number of domain identities because the algorithm does not visit a domain identity more than once when searching for receipts. If *P* does not detect itself, *all* the domains seen in the tree could be fake identities created by *C* and all the edges fake re-

ceipts signed by these fake identities, in an attempt by *C* to fool the reciprocity mechanism, so *C* should be excluded in order for the right incentives to be provided.

On the other hand, by actually detecting itself in the tree, *P* is not concerned about potential Sybil attacks any more. *P* only wants to uphold the reciprocity rule: *P* was indeed offered service by the community, so *P* now gives back to the community. After contributing, however, *P* must mark the particular receipt that connected him to *C* to be 'used'. If *P* did not do this, a single act of consumption on *P*'s part could prompt many contribution acts. This way, even if the domain that actually provided service to *P* colludes with other domains, the colluding team as a whole will not be able to take advantage of a single act of contribution more than once.

4.2 Evaluation

What is the percentage of unfair exclusions that this algorithm causes? To measure it we simulated a P2PWNC community that used the above algorithm for authentication. In our simulation, all unsuccessful terminations of the algorithm were unfair exclusions because none of the simulated domains was attempting to free-ride. (The algorithm would detect such a free riding domain immediately since no receipts proving contribution would exist and the algorithm would terminate in its initial steps.)

In order to choose an appropriate simulation model, we support that a metropolitan-sized P2PWNC would most probably exhibit a *small world* [9, 11] phenomenon[3]. More specifically, we base our P2PWNC model on a small world model similar to the one presented in [9]. Our set of nodes representing P2PWNC hotspots are identified with the set of lattice points in a n x n square (see Fig. 3). For a universal constant p all nodes have *local relationships* with nodes within lattice distance p. Each node also has q *distant relationships* with nodes chosen uniformly at random during system initialization. This models relationships that could also cause interactions, and, by extension, P2PWNC transactions.

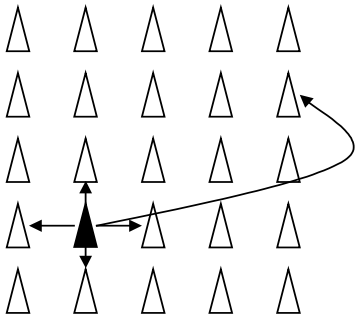

Fig. 3. A P2PWNC with 25 hotspots (n = 5). Here, p = 1 and q = 1. We focus on the relationships of the black hotspot, which we depict using arrows that start from it.

[3] Small world graphs mainly arise in social contexts but have also been used to model user ad hoc networks [12]. In running our simulations, we arrived at the same conclusions as [12] and [13], namely that due to random mobility the receipt graph is even more connected.

In our simulations, all hotspots are symmetric, with exactly one registered user. For a given set of *movement probabilities* encoded in the universal constants *p_l*, *p_d*, and *p_r* (all in the range [0, 1]), these registered users spend (a) *p_l* of their time accessing local contacts (choosing uniformly at random which one), (b) *p_d* of their time accessing one of the *q* distant contacts (choosing uniformly at random which one), and (c) *p_r* of their time accessing any hotspot chosen uniformly at random. This last probability of movement acknowledges the fact that completely random movement is also a possibility.

To allow for the first receipts to be created, a percentage of the hotspots (*p_a*, in the range [0, 1]) are altruists at the beginning of time. Altruistic hotspots (as defined in Section 2.1) authorize users without running the authentication algorithm. Nevertheless, they do ask for signed receipts. For the simulation, we also apply the following heuristic: every altruistic hotspot stops being altruistic and becomes a regular non-altruistic (i.e., rule-enforcing) hotspot when the hotspot's registered user successfully passes his first authentication in a non-altruistic hotspot. There are various local policies that hotspots may use to decide something similar. What is important is that new hotspots must start by being altruistic in order to collect a certain number of receipts and that they should eventually turn to non-altruistic hotspots in order to provide correct incentives to visitors and stop encouraging free-riders.

The main result of our simulations is that for realistic sets of relationships and mobility patterns there is always a TTL value *above* which unfair exclusions can drop below 1% of the total number of authentications, and *below* which the system eventually collapses (i.e., it surpasses 99% unfair exclusions). We observed that whenever the unfair exclusion rate shows even the slowest, but persistent, rate of increase, then the system is bound to collapse. This can be explained if we consider that as altruist numbers are reduced according to the heuristic, rule-enforcing providers increase. If an unfair exclusion due to expired receipts occurs then this would result in yet another receipt not being created. As time progresses, the total number of receipts in the system decreases and eventually we return to the initial conditions, i.e., with no receipts in the system, but without altruists anymore all visits result in exclusions.

We simulated 1024 hotspots arranged in a 32 x 32 grid with $p = 1$, $q = 2$, $p\_l = 0.05$, $p\_d = 0.05$, and $p\_a = 0.99$. In the following figures (4 through 6), we depict results from simulations with four different sets of parameters: two different values for the TTL (30 and 60) and two different values for *p_r* (0.00 and 0.02), with the remaining parameter values as above. For every step of the simulation, all users visit foreign hotspots according to the preset relationships and probabilities of movement. With the parameter values above, users stay at home 90% or 88% of the time (if $p\_r = 0.00$ or $p\_r = 0.02$ respectively). The TTL is measured in simulation steps.

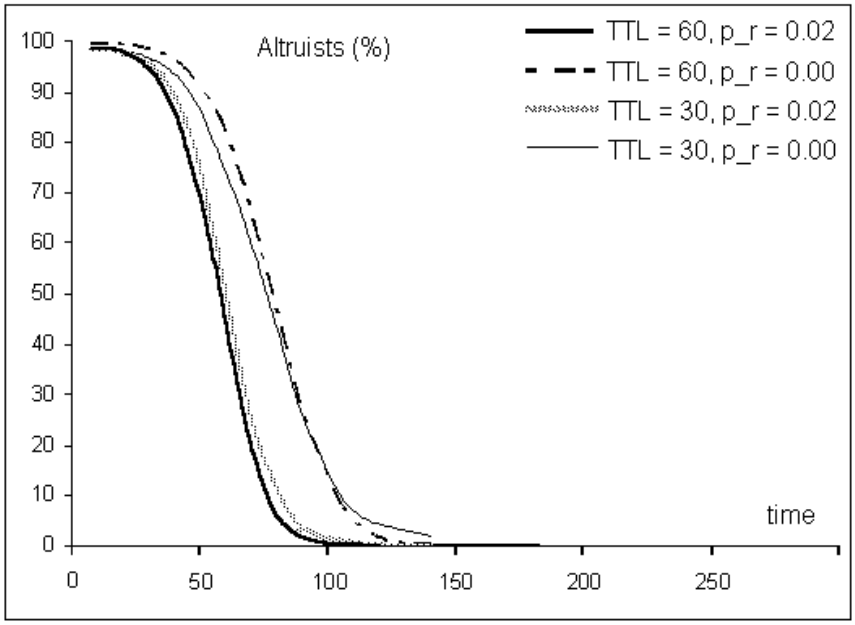

Fig. 4. The percentage of altruistic hotspots.

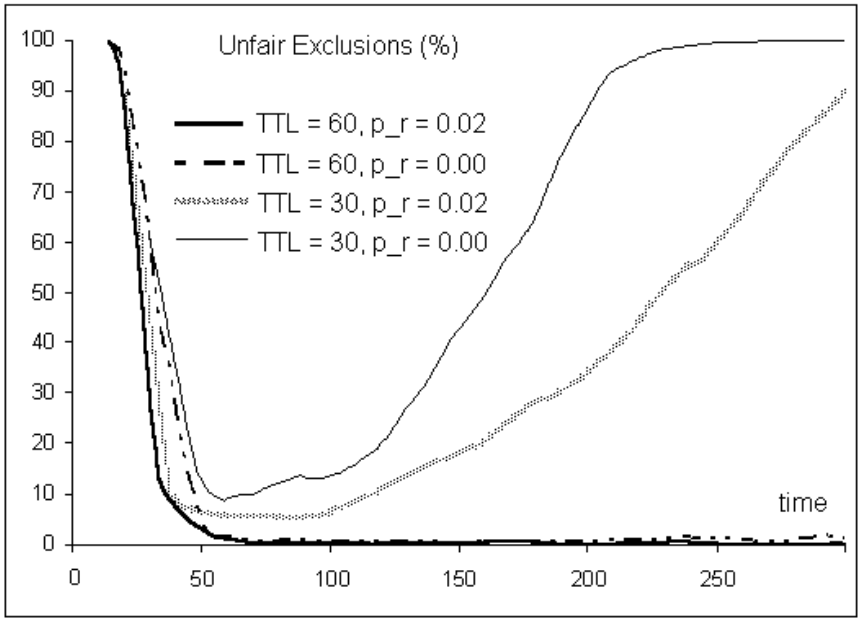

Fig. 5. The percentage of unfair exclusions.

Fig. 4 depicts the evolution of altruistic hotspots. Altruistic hotspots in all tests start at 99%. Random user movement (with $p\_r = 0.02$) across all hotspots causes our heuristic to turn altruistic hotspots to rule-enforcing faster because hotspots have a chance of interacting with users from a greater number of foreign hotspots which gives more chances to our heuristic test to run and eventually succeed.

Fig. 5 supports our main finding, namely that there exists a TTL value above which there would be insignificant gains in performance (low percentage of unfair exclusions) since unfair exclusions are very near zero, and below which the system eventually collapses (unfair exclusions reach 100% of authentications when, eventually, all receipts expire and new ones cannot be issued because altruists have disappeared). Note that random movement is not required to achieve low unfair exclusion rates. Of course, in the case of system collapse, random movement can delay the collapse somewhat (as can be seen in Fig. 5). In the real world, collapse can be avoided and unfair exclusions can be even lower if some hotspots remain altruistic (here, after the 150th simulation step, approximately all hotspots have turned to rule-enforcing hotspots – see Fig. 4). In fact, a reverse heuristic could be that after an unfair exclusion a rule-enforcing hotspot could turn altruistic, and then the original heuristic would apply again. (Also, in the real world, all new hotspots must be altruistic when they first enter the system, as we mentioned before.)

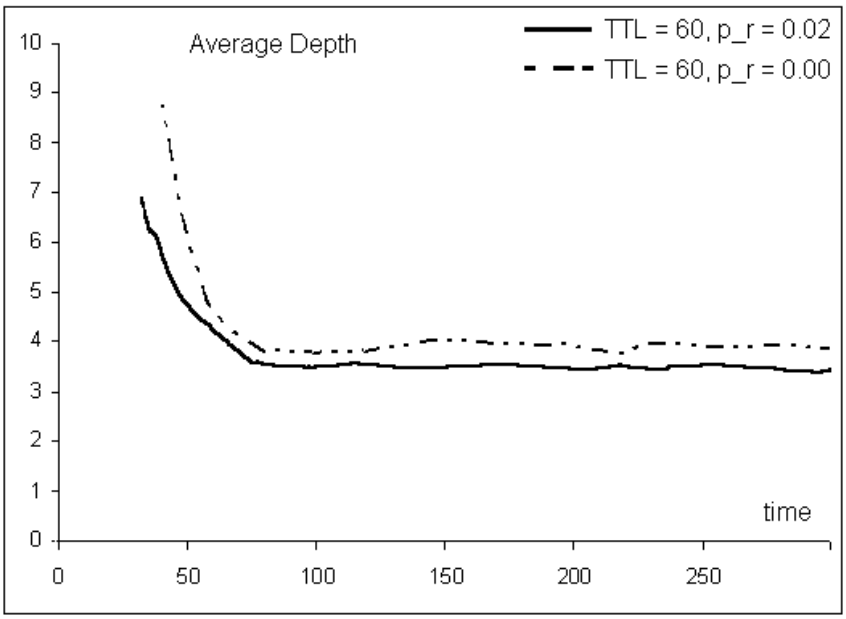

Fig. 6. The average tree depth at which a successful authentication terminates.

Fig. 6 is a direct result of our small-world model: in the two sustainable 1024 hotspot systems (those with TTL = 60) it takes only four steps on the average in the receipt graph to reach a hotspot from any other hotspot. Also, it takes slightly less than four on the average if there is some element of completely random movement.

4.3 Enhancements

Even though the algorithm appears to work theoretically there is still the practical issue of efficiency. Given that receipts are stored across different nodes in decentralized storage, it could take a long time for the authentication algorithm to access them before it terminates, which is a significant cost that the potential provider may not have an incentive to pay. One way to reduce this cost is to assist potential providers by *precomputing* paths. More specifically, the domain agent in every P2PWNC domain can keep a background process running that searches the decentralized storage for new receipts. The goal is to build a tree of receipts (like the one in Fig. 2), with the domain in question at the root of this tree, and to update it continually as receipts expire and as new receipts enter the system. The home domain could send fresh versions of this tree to its roaming registered users whenever the opportunity arose. The tree does not have to contain the receipts themselves. Instead, for every tree node, needed are only some of the least significant bits of the domain name that corresponds to this node.

The procedure above concerned domains that played the part of consumers. Providers, on the other hand, can also maintain a similar tree. These 'provider' trees would have one important difference: the root would again be the domain in question but the directed edges would point to the opposite direction.

Using standard probability computations, we can see that the probability that the provider and the consumer would have at least one node in common increases rapidly even with modest increases in the number of nodes in these trees, and even for very large P2PWNCs. A node chosen by both provider and consumer is equivalent to saying that at least one connecting path would exist connecting the two. The provider would only have to check the receipts on this path (also, unlike the mobile client, the provider can save the entire receipts, not just the least significant bits of names).

5 Decentralized Storage Subsystem

In order to build a self-sufficient receipt storage system that relies only on the domain agents themselves, we borrow ideas from the *Chord* [14] peer-to-peer lookup protocol. The P2PWNC Public Key Infrastructure we proposed is compatible with the one proposed in [15], where it Chord security extensions are presented and where it is suggested that each live node of the storage subsystem uses as *secure Chord identifier* the hash of a data structure that includes the node's public key, and its current IP address. Our main concern is the *availability* of receipts, as storage nodes join and leave the system. P2PWNC receipts are self-certifying data structures, so no additional protection is needed to ensure their integrity. Inbuilt Chord securities (such as *consistent hashing*) and the PKI extensions described in [15] ensure that an adversary cannot affect a large part of the system without first controlling a significant number of IP addresses in different subnets. In order to avoid free riders at the storage layer, nodes need to process queries only when they are coming from nodes that provably store their share of receipts according to the Chord mapping function. This is relatively easy to check at the Chord layer in order to exclude nodes that refuse to take on their share of the load.

6 Conclusions

In this paper we claimed that a metropolitan-sized P2PWNC would exhibit a small world phenomenon and with the help of simulations we showed how the fundamental Sybil attack problem can be circumvented Our goal was to ensure that participants that attempted to free ride would be excluded from consuming community resources. We linked system memory (expressed as the receipts' Time-to-Live) to system performance (expressed as the percentage of unfair terminations of the P2PWNC authentication algorithm). We showed that there exists a TTL value above which there would be no appreciable gain in system performance. We also presented our ideas on how to make the basic authentication algorithm more efficient and we outlined the foundations of the P2PWNC decentralized storage subsystem.

7 Acknowledgments

This work is partly supported by the EU research project 'Market Management of Peer-to-Peer Services' (MMAPPS, RTD No IST-2001-34201). The authors gratefully acknowledge ongoing discussions with their MMAPPS partners, and particularly Panayotis Antoniadis and Thanasis Papaioannou.

References

1. Nokia – Nokia 9500 communicator. http://www.nokia.com/nokia/0,,54106,00.html.
2. GSM Association. PDR IR.61, WLAN roaming guidelines, 2003.
3. Wireless Broadband Alliance. http://www.wirelessbroadbandalliance.com.
4. Boingo Wireless Inc. http://www.boingo.com.
5. iPass Inc. http://www.ipass.com.
6. E. C. Efstathiou and G. C. Polyzos. A peer-to-peer approach to wireless LAN roaming. 1st ACM Int'l Workshop on Mobile Applications and Services on WLAN Hotspots, 2003.
7. C. Courcoubetis and R. Weber. Asymptotics for provisioning problems of peering wireless LANs with a large number of participants. In Proc. WiOpt'04, UK, 2004.
8. E. Adar and B.A. Huberman. Free riding on Gnutella. *First Monday*, 5(10), 2000.
9. J. Kleinberg. The small world phenomenon: an algorithmic perspective. In Proc. STOC'00: 32nd ACM Symposium on the Theory of Computing, Portland, OR, 2000.
10. J.R. Douceur. The Sybil attack. 1st Int'l Workshop on Peer-to-Peer Systems, 2002.
11. S. Capkun, L. Buttyan, and J.-P. Hubaux. Small worlds in security systems: an analysis of the PGP certificate graph. New Security Paradigms Workshop, 2002.
12. S. Capkun, L. Buttyan, and J.-P. Hubaux. Self-organized public-key management for mobile ad hoc networks. *IEEE Transactions on Mobile Computing*, 2(1), 2003.
13. S. Capkun, J.-P. Hubaux, and L. Buttyan. Mobility helps security in ad hoc networks. 4th ACM Symposium on Mobile Ad Hoc Networking and Computing, 2003.
14. I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup protocol for Internet applications. *IEEE/ACM Transactions on Networking,* vol. 11, Feb. 2003.
15. E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. 1st Int'l Workshop on Peer-to-Peer Systems, 2003.