Proceedings of the 2007 IEEE
Workshop on Information Assurance
United States Military Academy, West Point, NY, 20-22 June 2007

SWAP: **S**hared **W**ireless **A**ccess **P**rotocol (using Reciprocity)

Matthew W. Dunlop, Ginger Perng, David G. Andersen

Abstract— **Wireless access points are becoming more and more prominent in the home, yet there is no incentive to encourage access point owners to share their service. We introduce SWAP, a lightweight protocol that uses reciprocity to motivate users to share service. Each node participating in SWAP stores perishable receipts that are used to calculate a user's rating (how much the user shares his or her access point). SWAP does not use a centralized authority to store or validate receipts nor does it place an excessive burden on peers. SWAP is also robust against collusion, which we show through analysis and implementation. As demonstrated by an implementation of the most computationally expensive portions of the protocol, SWAP imposes little overhead even on mobile devices.**

Keywords—**Community Wireless, Reciprocity, Wireless Security**

I. Introduction

Recent years have seen an explosion in the use of wireless networking technology. Unfortunately, while city-wide wireless Internet access is being explored in a few locations [1], [2], [3], [4], most communities are still far from having ubiquitous wireless. Users can often rely only on finding access through a coffee shop, hotel, or airport, and must typically navigate one of several payment systems to make use of one of these wireless access points. The coverage area could be greatly expanded if users could access any of the home access points in a community, but owners of these access points have no incentive (and many disincentives) to share their resources with the community.

We address this hurdle by proposing a mechanism, the Shared Wireless Access Protocol (using Reciprocity), or SWAP, that guarantees reciprocity to those who grant others access to their resources. Reciprocity, as it applies to networking, is the ability for a user who has provided resources to another user to receive services in return. We have expanded this definition to include *transferable reciprocity* under which a user can receive service from any other user as long as she shares her service.

SWAP is a lightweight protocol that encourages users to share service, detects collusion, and limits freeloading. It does not require either a trusted third party or a large number of peers to validate each exchange of service. SWAP detects collusion *without* auditing every transaction, and requires little communication between participants.

To present SWAP in context, Sec. II reviews related work and outlines the key features that make SWAP unique. Sec. III discusses our design considerations. Sec. IV identifies the entities that make up the system model. Sec. V describes the SWAP protocol itself. Sec. VI presents an analysis of the random audit process. Sec. VII covers the implementation of several portions of SWAP. We conclude and discuss future work in Sec. VIII.

M. W. Dunlop: U. S. Military Academy, West Point, NY.
G. Perng: Carnegie Mellon University, Pittsburgh, PA.
D. G. Andersen: Carnegie Mellon University, Pittsburgh, PA.

II. Related Work

Approaches to sharing the upstream capacity of fixed wireless access points typically use one of three techniques: reputation, micropayments, or reciprocity.

Reputation-based schemes use good reputation to draw users to access points (APs) [5], [6]. These schemes typically use monetary incentives to encourage APs to maintain good reputations and track reputations at a central authority. A notable reputation scheme is MoB [5], which allows mobile users to choose their provider based upon some criteria (e.g. price, bandwidth, reliability, etc.). SWAP focuses instead on providing different levels of service to users based upon how much they share their own APs. SWAP does so without monetary incentives or a central reputation authority.

Micropayment-based schemes use credit as digital money. A user can acquire service from any other user by spending digital money. Micropayment-based schemes rely on either a centralized authority or other peers to verify the authenticity of the digital money. PPay [7], for example, limits the use of a central authority by allowing peers to "own" coins. Coins may be transferred from peer to peer as long as the original owner of the coin is involved. Karma [8] is a decentralized peer-to-peer micropayment scheme in which a set of participant nodes maintains a user's credit rating (karma) and information regarding the user's recent payments. Every transaction a user makes must be recorded by the set of nodes responsible for her credit rating, potentially creating a heavy burden on those nodes. Like micropayment schemes, SWAP does not require previous contact between two users in order to receive service. Unlike micropayment schemes, SWAP does not rely on a central authority nor does it excessively burden peers for credit tracking.

Reciprocity-based schemes have peers record who has granted them service and grant service only to those entities who have granted them service. Direct reciprocity is when A grants service to B because B granted service to A. Under indirect reciprocity [9], A grants service to B who grants service to C who grants service back to A. Reciprocity-based schemes typically involve an exchange of credits between users. Once those credits are exhausted, a node cannot receive service from a particular node or set of nodes until it again gains credit from that node or group. These schemes do not rely on a central authority.

ISBN 9-9999-9999-9/99/ $20.00

Reciprocity-based schemes include Samsara [10] and P2PWNC [11]. Samsara is a peer-to-peer storage system that enforces fairness among peers via transferable "claims." A peer using storage from another peer gives a claim that allows the other peer to use its space. P2PWNC groups users into teams. Wireless users access foreign teams' APs based on their team rating. The algorithm builds a receipt graph of directed paths to determine who should be granted service. Service is granted only if there is a direct or indirect path from the consumer to the provider. Like other reciprocity-based schemes, SWAP does not require a centralized authority to manage credit. Unlike other reciprocity-based schemes, SWAP does not require previous contact between two users in order to receive service. Additionally, SWAP's credits are not depleted during connections to foreign access points. Instead, credits expire over time. Users may thus use other APs as long as they share their own APs.

III. Design Considerations

SWAP is designed with two considerations in mind: the incentives for a user participating in a shared wireless access protocol, and the requirements, such as computational feasibility, that the system must meet.

A. Incentives

SWAP's primary incentive for users is delayed reciprocity: the *promise* of receiving service for providing service [12]. In order to access the Internet using SWAP while away from their home access points, users must first share their own service. The system can promote sharing if users receive better service by sharing more of their own service with others.

A second incentive is the network effect: As more users share their service, users are more likely to be able to receive service at a particular location [13].

A third benefit to users is cost/performance: SWAP is free if you share your own AP, and 802.11 access points are typically much faster than other wireless access methods such as 3G cellular data networks. (Users will often travel to an access point to avoid slow 3G connection speeds [14].) Finally, some users might share their service for ideological reasons of sharing or undermining the telecommunication giants [14].

These incentives encourage users both to provide service and to improve the service provided. For example, a typical indoor 802.11b/g wireless access point has a range of 46 *m*[15]. Outdoors, the same wireless access point can exceed 92 *m*. Users can extend the range of their APs in other ways, such as using antennas or amplifiers [16]. Improved service will attract more users, which in turn grants the AP's owner better service when away from home. Such "competition" expands the coverage and quality of the entire system.

B. Requirements

A reciprocity-based credit system for wireless access networks should meet five requirements:

1. The system should motivate users to share their resources. The system should limit "freeloading" (consuming resources without contributing resources).
2. The system should allow users with credit to receive service from any other user. In other words, credit for providing service should be transferable, and access to resources should not be based solely on direct reciprocity between two users.
3. The system should not need a trusted-third party (TTP). SWAP's goal is to allow users to benefit from each others' access points without incurring extra costs. Introducing a TTP to maintain the credit accounts of all users in the system may incur costs that undermine the goal of being entirely community-based.
4. The computational, storage, and latency overhead of the protocol should be low. For example, the time to issue credit for having received service should not be a significant portion of the service time.
5. The system should be able to detect peers who collude to subvert the system. Furthermore, the system should minimize the gains peers can receive from collusion.

IV. System Model

SWAP brokers access between two entities: *mobile nodes* that wish to receive broadband service; and *access points* (APs), wireless routers that provide access to the Internet. Each mobile node is associated with one *home* AP and freely receives service from this AP. *Roaming* mobile nodes wish to receive service from *foreign* APs. SWAP allows roaming nodes to receive service from foreign APs provided that their home APs are willing to service other roaming nodes.

To give APs credit for providing service, each mobile node M has a private key sk_M that it uses to sign receipts for service from APs. Receipts are stored in a distributed hash table (DHT) using a consistent hashing protocol which we discuss in more detail in Sec. V. A mobile node is bound to its associated AP through a certificate signed by a node in the DHT that verifies both the mobile node's public key and the AP's network address. The certificate issuer assigns each mobile node a unique identification number (ID) used to identify itself to foreign APs.

All entities must have clocks loosely synchronized to within a few minutes. The clock is used only to timestamp receipts.

V. SWAP protocol

To gain access from a foreign AP, A_f, a roaming mobile node, M, first sends its certificate (containing its ID and the network address of its AP) to A_f. The foreign AP then contacts the mobile node's AP, A_M, to retrieve receipts that prove that A_M has given access to other mobile nodes. A_f then verifies the validity of the receipts it receives. It may also randomly audit some of the receipts (Sec. V-C). From the receipt set, A_f determines how much bandwidth to give the roaming node. After this negotiation, the mobile node can access the AP. Throughout the active session, the mobile node generates and sends receipts for the service it receives. The simple protocol is shown in Fig. 1.

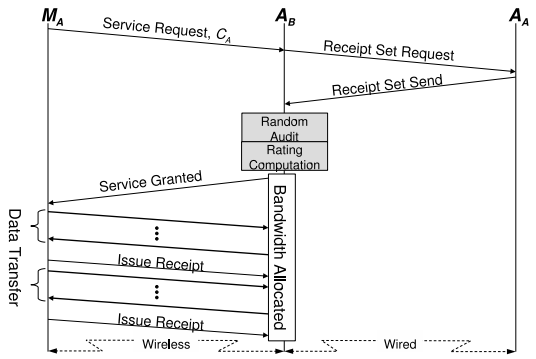


Fig. 1. SWAP protocol

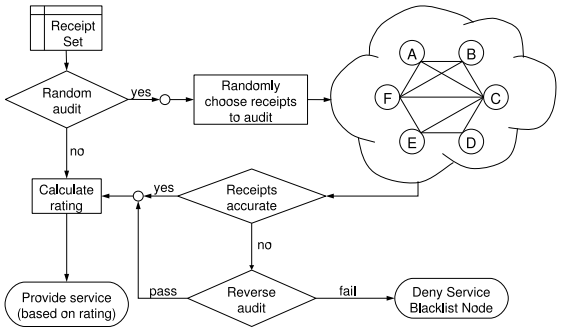


Fig. 2. Random Audit Process

The AP logic for accepting a mobile node's request for access is shown in Fig. 2.

We next explain six specific components of the SWAP protocol: receipt generation, receipt storage, random audits, rating calculation, service denial, and the way nodes join the system.

A. Receipt Generation

When a mobile node M receives service from an AP A_f, it credits the AP by generating a *receipt* for the AP. The receipt is of the form:

$$R = \{A_f, \mathsf{ID}_M, b, t_{\mathsf{start}}, t_{\mathsf{end}}\}_{\mathsf{sk}_M}$$

where ID_M represents the ID of M, b represents the number of bytes transferred, and t_{start} and t_{end} represent the start and end time of service, respectively. The five fields are signed by M's private key which is denoted by $\{\cdot\}_{\mathsf{sk}_M}$.

Receipts are created periodically for the AP granting service. Periodic receipts discourage corrupt nodes from refusing to credit the AP after receiving service. If a mobile node fails to generate a receipt after the fixed time interval, the AP can deny the mobile node further access. To reduce storage overhead, receipts express cumulative amounts of time such that a new receipt will replace the previous one stored at the AP.

Analogous to receipts are *nonreceipts*: certificates summarizing the times which a mobile node is idle and has not been accessing a foreign AP. Nonreceipts are receipts signed by a mobile node to its own AP. They allow non-roaming mobile nodes to receive credit. This dissuades nodes from giving receipts to APs from which they have not received service, as they can credit themselves instead. Nonreceipts have the same form as regular receipts, and are stored in the same manner as receipts. Therefore, we limit our discussion to only receipt storage.

B. Receipt Storage

When an AP receives a receipt R, it stores the receipt both locally and on one other designated AP, A'. A' is chosen using consistent hashing [17]. Similar to many peer-to-peer distributed hash tables (DHTs) (e.g., [18], [19], [20], [21]), SWAP uses the hash function to map a receipt to an AP that stores it.

A DHT gives nodes and objects identifiers generated by a globally-known hash function, $g(\cdot)$ (e.g., SHA-1 [22]). A node's identifier determines the objects for which the node is responsible. DHT protocols are designed to scale to a large number of nodes; therefore, the primary goal of these protocols is to quickly determine the location of an object. Any DHT implementation can fulfill this requirement for SWAP. One possibility is an external "service" DHT such as OpenDHT [23].

An AP's identifier is determined by hashing the network address of the AP. Receipt identifiers are determined by hashing the start time of the receipt, rounded down to the nearest hour and the ID of the mobile node that generates the receipt: $g(t_{\mathsf{start}}, \mathsf{ID}_R)$. This allows all receipts generated by the same mobile node in the same start hour to be mapped to the same AP. This mapping is enforced to reduce the overhead in random audits, while ensuring that a single compromised node does not affect a wide time range for any mobile node.

Unlike conventional credit, which does not expire, receipts are only valid for a limited time. When a foreign AP requests receipts from a mobile node's AP, only receipts for the *active time period* are sent. When a receipt is older than the current active time period, it is expunged from the sets of receipts that the AP stores. This reduces the overhead of storing receipts to storing only receipts valid at the current active time period.

C. Random Audit

The random audit allows A_f to catch mobile nodes that generate multiple receipts for the same time period (impossible, since a node is only allowed to use one AP at a time). When an AP receives a set of receipts S from a mobile node's AP, it may decide to randomly audit some of those receipts. The AP audits a receipt by querying each AP that is responsible for every hour within the receipt's start and end times. For each receipt R that A_f wishes to verify, it does the following:

1. Round down the start time and end time of the receipt to the nearest hour, h_{start} and h_{end} respectively.
2. For each hour h from h_{start} to h_{end}:
   (a) Determine the AP A_h that stores the receipts for the mobile node that generated the receipt M_R for that hour by calculating its identifier, $g(h, \mathsf{ID}_R)$.
   (b) Send an audit request message containing R to A_h.

When A_h receives an audit request, it looks at all of the receipts it has stored for the hour and determines if the generator of R, M_R, created more than one receipt R' that overlaps R's time period. If not, A_h returns NOCOLL to the querying AP, signifying that R is valid. Otherwise it returns COLL to the querying AP and provides the set of overlapping receipts from the cheating node. While a cheating AP could falsely return NOCOLL, the proof of cheating is self-verifying, since it consists of receipts that only the cheating node could generate.

If any A_h returns COLL for any receipt R, A begins a *reverse audit* for the node requesting service, M. When R and R' collide, the culpable party is *not* necessarily M, the node requesting service. Rather, it is the node that generated the receipt, M_R for A_M. However, if M_R and M are colluding, it is likely that M has also generated receipts for M_R. Therefore, the reverse audit checks all receipts generated by M during the active time period. If there are collisions between any receipts during the reverse audit, then M is also guilty of colluding. If there are no collisions, A gives access to M. The logic for A_f granting access is shown in Fig. 2. Confirmed cheating nodes are either temporarily or permanently banned from the community as described in Sec. V-E.

To evade collusion detection, a malicious AP may refuse to push its receipts to the DHT (this way no false receipts are stored in the DHT). To prevent this, A_h issues NOFINDs to the querying AP for each receipt R that is audited, but not found. If a querying AP receives NOFINDs for a significant number of the receipts it has queried, it denies the roaming node access. Additionally, if the querying AP receives a NOFIND from A_h for R, the querying AP sends R to A_h. This is safe to do since A_h can verify the validity of R. Since the querying AP sends missing receipts to the appropriate node in the DHT, those receipts intentionally omitted by the malicious AP will get stored in the DHT simply by requesting service. Thus, collusion can still be detected during subsequent service requests.

Note that there may be cases when A_h might legitimately not respond to an AP's audit request (e.g., A_h is temporarily offline or the audit request gets dropped by the network). In such cases, the AP simply ignores the receipt stored at A_h in its bandwidth calculation for the node.

Optionally, A_f can query the DHT to determine if the access point belonging to the receipt issuer, A_R, has provided service to any mobile node other than itself or M. If not, A_R is suspected to be a Sybil node [24] of A_M. For verification, A_f queries the DHT to determine if any A_{R^*} in A_M's receipt set has provided service to any mobile nodes other than itself or M. If most nodes never receive service from anyone else, it is reasonable to accuse M of employing a Sybil attack and ban her from the community.

D. Rating Computation

Once mobile node M passes the random audit, AP A_f computes the node's rating, which determines the amount of service A_f grants M. The rating is based upon the total amount of time that A_M provides service:

$$r_M = \sum_{i=1}^{U} t_i \tag{1}$$

where r_M is M's rating, U is the number of users that received service from A_M, and t is the total amount of time service was provided to each user.

Time is used as a linear measure for calculating rating in order to provide the greatest reward to APs that provide the most service. We assume that users who receive poor or low-bandwidth service from an AP will terminate service of their own accord and not return to the poor AP. Therefore, APs are discouraged from trying to game the system by providing poor service to many people. The incentive for APs is to provide the minimum level of service to users that keeps them happy, which is in line with SWAP's goals.

While our scheme works well in most scenarios, it is ineffective for APs in isolated areas. These APs may see only a few roaming nodes for short periods of time, and the resulting ratings will be similar to that of a non-sharing AP. To encourage such APs to provide service (the service may be quite valuable to those few users who use it, since access is presumably rare), (1) can be modified such that there is a higher reward to APs for smaller amounts of sharing. However, this modification allows a malicious AP to modify its behavior to maximize rewards without providing honest sharing. For this reason, a nonlinear rating equation should only be used in a trusted environment.

After calculating M's rating (r_M), A_f grants service to M proportionally to r_M:

$$allocated\_bandwidth_M = \frac{total\_bandwidth \cdot r_M}{\sum_{U'} r} \tag{2}$$

where U' is the number of users currently accessing A_f.

E. Service Denial

A node confirmed as cheating is added to a revocation list and denied service. If a node fails a random audit, it is suspected of cheating. The accuser then requests that the AP located at the hash of the current time and the ID of the accused node conducts a confirmation audit. If this audit also fails, then the accused node is confirmed as cheating. Nodes added to the revocation list can either remain on the list permanently or decay over time.

F. Bootstrapping

A node joins SWAP by registering with the DHT to receive an identity certificate. It can then build its receipt set by allowing mobile nodes to roam on its AP. It can also increase its receipt set by generating nonreceipts for itself. Thus, an idle mobile node may later receive some service from foreign APs even though it has only recently joined the system.

VI. Analysis

This section presents an analysis of how SWAP prevents cheating. We assume that most users of the system are not ma-

licious. We begin by discussing how a malicious node may attempt to cheat. This is followed by a discussion of the countermeasures we take to mitigate collusion in SWAP. We then present the mathematical likelihood of detecting collusion based on our primary countermeasure, random audit. A demonstration of the validity of our calculations is given through implementation of the audit process.

A. Possible Ways to Cheat

There are many ways a malicious node may attempt to cheat in SWAP. We address each of these and show how SWAP mitigates the threat.

Sybil attack. An AP might augment its receipt set with receipts from fake mobile nodes. This type of cheating is known as a Sybil attack [24]. SWAP prevents this by randomly checking receipts to see if receipt issuers are sharing service (described in Sect. V-C). The rationale is that a Sybil node will only receive service from itself, the "real" node, or not at all. A node using a second layer of Sybil nodes to provide service to the first layer of Sybil nodes can escape detection. We accept this risk in the interest of keeping SWAP completely decentralized.

Service theft. A mobile node may attempt to steal service by not issuing receipts to an access point for service provided. As described in Sect. V-A, this cheating is mitigated by the access point terminating the connection with the mobile node after the first interim receipt is not received. The gain from this attack is less than or equal to the minimum interim receipt period.

Service hoarding. An access point may share service with only one or a few mobile nodes. According to the semantics of the protocol, the access point's corresponding mobile node will not be denied service. However, the amount of service this node receives will be reduced, according to the rating calculation discussed in Sect. V-D.

Non-participation. A selfish user may participate in the protocol but refuse to share her access point with anyone. Instead the user may build up her rating solely by signing nonreceipts to herself. We do not prevent this attack because nonreceipts are useful to bootstrap into the system as well as mitigate collusion.

Although it is possible for a selfish user to receive service from other APs this way, her bandwidth allocation will be smaller than that of someone who shares. For example, in a specified time period a sharing user can accumulate U times as much time as a user who only signs nonreceipts. As a result, a sharing user will receive U times as much bandwidth as a selfish user. This effect is compounded by the number of sharing users accessing the same foreign access point as the selfish user. Moreover, a selfish user cannot generate nonreceipts while roaming to other access points.

Drop attack. When an access point stores duplicate receipts at a remote access point (see Sect. V-B), the remote access point might drop these receipts instead of storing them. The access point may simply be selfish and not wish to store receipts for others. More dangerously, it may drop receipts in an attempt improve its own service. Since storage is cheap and we assume most users of the system are not malicious, we do not consider the first to be a compelling reason for this attack. The second reason, although more compelling, is not likely to produce noticeable results.

Consider the possibility that a remote access point stores a duplicate of one receipt out of every AP's receipt set (due to the distributed nature of how receipts are hashed in the DHT, this assumption is unlikely). If a malicious remote AP drops these receipts, it has the potential to reduce each AP's rating from $\sum_U t$ to $\sum_{U-1} t$. This potential is only realized if the dropped receipt is chosen by A_f during an audit. The probability p that this occurs is $p = \frac{\text{number of receipts audited}}{\text{total receipts in set}}$. Regardless of whether the dropped receipt is audited, if U is large the impact of one missing receipt is small.

Recall that (2) uses the total rating of all connected users to calculate the allocated bandwidth for any particular user. Even if the malicious node is effective at getting A_f to drop one receipt from each connected user, the total sum of receipts of all connected users is likely large. Therefore, the expected gain by the malicious node is small.

Receipt masking. If an access point has receipts in its receipt set it knows are false, it may choose not to store duplicate copies in the DHT. By not duplicating these receipts, these inconsistencies are less likely to be detected during random audits as the audit will return NOFINDs. However, each time a NOFIND is received, the access point conducting the audit will push a copy of the receipt to the DHT as described in Sect. V-C. Missing receipts will thereby get copied to the DHT for verification in subsequent audits. If too many audited receipts from an access point return NOFINDs, the mobile node requesting service will be denied.

Ignoring Rating. An AP might ignore a mobile node's rating when determining the amount of bandwidth to provide. We allow this limitation because the added complexity of preventing this attack outweighs any potential damage it causes. We categorize the damage as minimal because the main incentive for this attack is for the AP to improve its own rating. To do this, the AP needs to maximize the quality of service it provides users (i.e. same bandwidth to all users) while still providing users with enough bandwidth to keep them satisfied. Maximizing the number of satisfied users is in line with the goals of the system. The danger of this attack is that if all nodes behaved in this manner, the incentive for sharing would disappear. For this reason, we assume that *most* users in the system are honest.

Collusion. Mobile nodes and APs may *collude* to give the appearance they are providing more service than they really are. Collusion is defined as malicious mobile nodes issuing false receipts (receipts for service not granted) to malicious APs. Nodes can collude in many ways. For example, a node can issues false receipts to multiple APs simultaneously. Alternately, a node can issue (false) receipts to a friend while actually idle. Collusion is a difficult problem faced by reciprocity schemes, and is addressed in the remainder of this section.

B. Reducing the Incentives to Collude

SWAP includes countermeasures to discourage collusion. Using these countermeasures, the risk of detection and the resultant consequences outweigh the gains from collusion.

The strongest countermeasure is the ability to probabilistically detect a colluder and impose a punishment for collusion. Collusion is detected using random audits of a user's receipt set as discussed in Sect. V. In SWAP, mobile nodes connect to only one access point at a time. If the same mobile node simultaneously issues receipts to multiple APs, it is cheating. If the cheating is detected, the colluder can be fined, removed from the system, or subjected to other appropriate punishment.

While audits detect mobile nodes issuing multiple receipts to different access points, they cannot detect nodes that issue false receipts when they are idle. To prevent this, SWAP nodes sign "nonreceipts" for themselves when they are idle, as discussed in Sect. V-A. Since a mobile node can improve its own rating during idle periods, it has no incentive to issue false receipts when idle. This countermeasure levels the playing field for honest users by giving an equal advantage to all users during idle periods.

The final countermeasure is selecting the access point at which to store receipts using both the node ID and the receipt time. If receipts were mapped according to ID only, a malicious node could collude with the access point responsible for storing its receipts. By including time, receipts from any particular node are spread out among the access points in the DHT.

C. Time to Detect Collusion

The previous section showed how SWAP can probabilistically detect collusion. In this section, we examine how long a colluder can cheat before getting caught. We do so by calculating the number of audits that take place before we can say with some percentage of certainty that we will detect collusion.

Recall from Sect. V-C that an access point can randomly audit any receipts from the receipt set. The probability of detecting collusion in a single audit is:

$$P_D = 1 - \frac{\binom{N-B}{n}}{\binom{N}{n}} \tag{3}$$

where N is the number of entries in the received receipt set, B is the number of false receipts in set N, and n is the number of receipts drawn from N to audit.

Equation 3 can be expanded to show the cumulative probability of detecting collusion within z audits.

$$P^*(Z \leq z) = \sum_{x=1}^{z} \prod_{i=1}^{x} \left(1 - \frac{\binom{N_i - B_i}{n_i}}{\binom{N_i}{n_i}} \right) \tag{4}$$

where N_i, B_i, and n_i vary with each random audit. If an access point does not conduct an audit, we ignore that connection when computing the probability of detecting collusion because the probability of detection is zero.

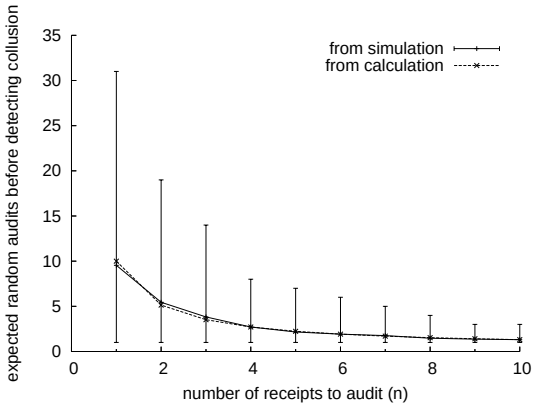


Fig. 4. The audit in which collusion was detected in simulation and analysis, using $N = 20$ and $B = 2$. The simulation is the mean and the extreme values of 100 runs for each value of n.

Assume that: (a) an AP's receipt set reaches steady-state (receipts expire at the same rate as they get added); (b) a colluder colludes at a constant rate; and (c) during each audit, the same number of receipts are selected for audit. These assumptions make N, B, and n constant over all random audits and allow us to rewrite the cumulative probability of detecting collusion within z audits as:

$$P(Z \leq z) = \sum_{x=1}^{z} P_D (1 - P_D)^{x-1} \tag{5}$$

Fig. 3(a) uses (5) to illustrate how long collusion can continue before it is detected. For example, assume the receipt set has 20 receipts and two of them are false. By auditing five receipts per random audit, we expect to detect collusion with 95% accuracy within five audits. SWAP can thus detect collusion with high probability without excessively burdening other peers with audit requests.

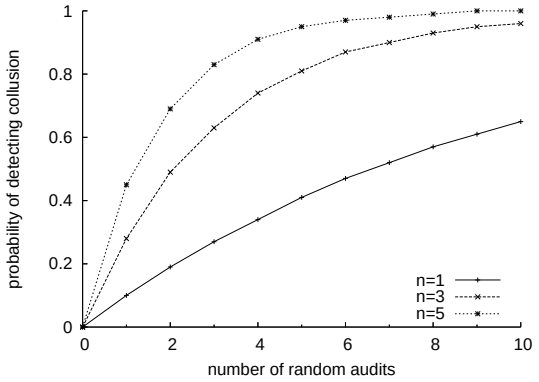
These results provide a useful upper bound on the probability of detecting collusion. In expectation, it takes fewer audits to detect collusion:

$$E[\#\text{ audits}] = \frac{1}{P_D} \tag{6}$$

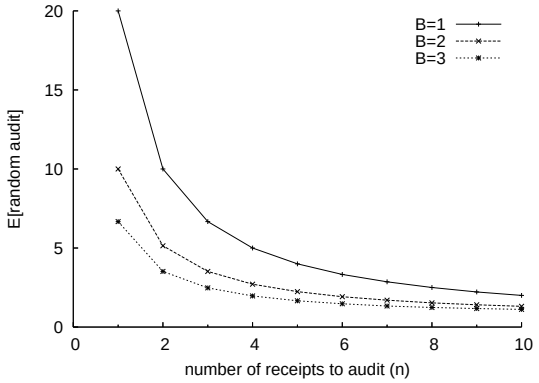
Using the example above with N=20, B=2, and n=5, an AP can expect to detect collusion on the second audit; cheaters cannot operate for long before being caught. Fig. 3(b) shows how the expected value varies with B and n.

As a proof of concept, we implemented the random audit process in Java to test how long the system takes to detect collusion compared to what we calculated. The implementation assumes uniform N, B, and n so it can use (5) and (6). It also assumes that APs perform an audit on each service request.

For each access point, we create a receipt set of size N and randomly replace B receipts with false receipts. The AP randomly chooses n receipts to audit. If the AP does not detect collusion, the mobile node is granted access for that connection. When the mobile node connects to another AP, a random audit is again conducted. This process continues until collusion is



(a) The probability of detecting collusion on or before the number of audits depicted on the x-axis assuming that 2 of 20 receipts in the set are invalid. Each line represents the number of receipts from the receipt set checked per audit.



(b) The expected audit during which collusion is detected using a receipt set of N=20, with various numbers of receipts sampled. Each line represents a different number of false receipts in the receipt set.

Fig. 3. Likelihood of detecting collusion

TABLE I
RSA AND DSA'S KEY SIZE (KS), KEY GENERATION TIME (KG), SIGNATURE GENERATION TIME (SG), SIGNATURE VERIFICATION TIME (SV), AND SIGNATURE SIZE (SS).

	KS (bits)	KG (ms)	SG (ms)	SV (ms)	SS (bytes)
RSA	1024	1083.5	257.3	6.2	128
DSA	512	179.4	40.0	224.4	47

detected. Fig. 4, compares the average audit during which collusion was detected in simulation vs. the results from (6). As the graph illustrates, the two curves closely overlap.

VII. MICROBENCHMARKS

To test the computational and storage overhead of using SWAP, we implemented the computationally intensive parts of the SWAP protocol (receipt generation, signature validation, and random audit). SWAP is implemented using Java 1.4.2 with the Java Cryptography Extension (JCE) [25] for digital signatures. All experiments were run on a Pentium 4 1.79 GHz processor with 640 MB of RAM running Microsoft Windows XP.

A. Digital Signatures

The largest computational overhead in SWAP is creating digital signatures. We examine two digital signature schemes, RSA and DSA, to determine which scheme is best for SWAP. We compare the two schemes on key size, time to generate a public/private key pair, time to validate a signature, and the size of the signature. Table I shows the average of ten runs.

RSA takes ten times longer than DSA to generate a key pair. However, SWAP's public/private key generation is a one-time cost per mobile node. DSA's signature generation is faster and the resulting signatures are much smaller than using RSA, but RSA verifies signatures much more quickly than DSA. As signature verification is more frequent than signature generation, we chose RSA to minimize the verification time. Additionally, RSA puts its expensive operation (signature generation) onto the person who is benefiting from the service.

B. Certificate and Receipt Sizes

Certificates contain a 4 byte network address, a 4 byte mobile node ID, and a 128 byte public key. Each unsigned certificate is thus 136 bytes long, and is 264 bytes when signed.

Receipts consist of a 4 byte network address, a 4 byte mobile node ID, 4 bytes for the number of bytes transferred, and 4 bytes for each of the start and end times. Unsigned receipts are therefore 20 bytes, and 148 bytes with their RSA signature.

C. Protocol Overhead

To estimate the overhead mobile nodes and access points incur while using SWAP, we use results from a study of wireless access patterns at Dartmouth University [26]. These results are based on traces generated by 7000 users on 550 APs using Dartmouth's wireless network during the Fall and Winter terms of 2003/2004. Relevant results from the study include:

1. An AP serves a maximum of 91 concurrent users.
2. An AP services an average of seven users per day.
3. The median session duration (roughly, the period of time a node is accessing an AP) is approximately ten minutes.
4. A node is on-line for an average of seven hours.

These results provide a gross over-estimate of the type of traffic an AP may encounter in SWAP. SWAP requires that each mobile node be associated with an AP to receive service, which is not the case in a campus environment. However, these results give a rough estimate of mobility patterns in a community wireless network.

We begin by estimating the worst case scenario for an AP receiving receipts from nodes using its service. If there are 91 users concurrently connected to an AP, and all nodes send their receipts at approximately the same time, then an AP must spend $91 \cdot 6.2\,ms = 562.4\,ms$ to verify the receipts. However, an AP services only an average of seven nodes per day. Even if all seven nodes access the AP at the same time, the AP would require only 43.3 *ms* to verify the receipts.

The computational overhead for mobile nodes to compute receipts while roaming is also small. The median session during which a node is associated with an AP lasts less than ten minutes. If the receipt period (the period of time which a mobile node must generate a receipt before an AP denies the mobile node further service) is greater than the session time, then a freeloading mobile node could refuse to generate a receipt for the AP and still receive all of the bandwidth for the average session length. To account for this, we arbitrarily set the receipt period to five minutes (half the median session duration). Accordingly, the time spent generating receipts for an average session is 514.6 *ms*, or 0.0058% of the total session time.

Finally, the storage required on each remote AP to store duplicate requests is also small. Since the average session time for a node is ten minutes, we assume that a node roams to six foreign APs in an hour generating six receipts. Therefore, the remote AP needs $148\text{ bytes} \cdot 6 = 888$ bytes to store this node's duplicate receipts for the hour. If we assume nodes roam during their average of seven hours a day on-line, then the storage needed across all APs to store duplicate receipts for all nodes' receipts for a day is equal to $888\text{ bytes} \cdot 7 = 6.22$ KB. Since the node ID is used as a factor in determining where in the DHT to store duplicate receipts, the storage requirement scales with the number of participants in the protocol.

VIII. Future Work and Conclusion

One area that is lacking in all works related to community wireless networks is user mobility patterns. It is unfortunate that we are unable to show the storage and computation overheads per AP as these depend on the mobility patterns of users in the community wireless networks. A future area of research would be to investigate mobility patterns in such networks and then analyze how real-world mobility patterns would affect SWAP.

SWAP currently assumes that each access point has only one associated mobile node. There are many scenarios in which multiple nodes are associated with a single AP (e.g., a family with multiple laptops). Multiple nodes may form a group and be associated with the same AP as long as they all share the same secret key. Only mutually trusting nodes should share a secret key as the dishonesty of one member affects the service that other members receive. A future area of research is to extend SWAP to support groups that share the resources of a single access point, or of multiple "clustered" access points.

To conclude, SWAP is a shared wireless access protocol that encourages members in a community to share their wireless access points. SWAP is a lightweight protocol that promotes reciprocity via receipts that are generated by nodes who roam on others' access points. SWAP detects collusion probabilistically and does not require a centralized server for each transaction. The analysis and simulation showed that SWAP can rapidly catch colluders with low communication and storage overhead, and the microbenchmarks showed that SWAP is computationally feasible. We therefore believe that a system such as SWAP is a promising step towards enabling more wide-spread community wireless sharing for ubiquitous network access.

References

[1] "Austin Wireless," http://www.austinwireless.net/.
[2] "NYCwireless," http://www.nycwireless.net/.
[3] "Wireless Philadelphia," http://www.phila.gov/wireless/.
[4] "Seattle Wireless," http://www.seattlewireless.net/.
[5] R. Chakravorty, S. Agarwal, S. Banerjee, and I. Pratt, "MoB: A mobile bazaar for wide-area wireless services," in *ACM MobiCom*, (Cologne, Germany), August 2005.
[6] N. B. Salem, J.-P. Hubaux, and M. Jakobsson, "Reputation-based Wi-Fi deployment," *ACM MC2R*, vol. 1, July 2005.
[7] B. Yang and H. Garcia-Molina, "PPay: Micropayments for peer-to-peer systems," in *ACM CCS*, (Washington, DC, USA), October 2003.
[8] V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer, "KARMA: A secure economic framework for peer-to-peer resource sharing," in *P2PEcon*, (Berkeley, CA, USA), June 2003.
[9] B. Greiner and M. V. Levati, "Indirect reciprocity in cyclical networks - an experimental study -," Discussion Papers on Strategic Interaction 2003-15, Max Planck Institute of Economics, Strategic Interaction Group, 2003. available at http://ideas.repec.org/p/esi/discus/2003-15.html.
[10] L. P. Cox and B. D. Noble, "Samsara: Honor among thieves in peer-to-peer storage," in *ACM SOSP*, (Bolton Landing, NY, USA), October 2003.
[11] E. C. Efstathiou, P. A. Frangoudis, and G. C. Polyzos, "Stimulating participation in wireless community networks," in *IEEE INFOCOM*, (Barcelona, Spain), April 2006.
[12] R. L. B. Bird and D. W. Bird, "Delayed reciprocity and tolerated theft: The behavioral ecology of food-sharing strategies," *Current Anthropology*, vol. 38, pp. 49–78, February 1997.
[13] "Melbourne Wireless," http://melbourne.wireless.org.au/.
[14] T. Schmidt and A. Townsend, "Why Wi-Fi wants to be free," *Commun. ACM*, vol. 46, no. 5, pp. 47–52, 2003.
[15] "Wireless/networking FAQs," http://compnetworking.about.com/cs/wirelessproducts/f/wifirange.htm.
[16] J. Bicket, D. Aguayo, S. Biswas, and R. Morris, "Architecture and evaluation of an unplanned 802.11b mesh network," in *ACM MobiCom*, (Cologne, Germany), August 2005.
[17] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigraphy, "Consistent hashing and random tree: Distributed caching protocols for relieving hot spots on the world wide web," in *ACM STOC*, 1997.
[18] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A scalable content-addressable network," in *ACM SIGCOMM*, pp. 161–172, 2001.
[19] A. Rowstron and P. Druschel, "Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM Middleware*, 2001.
[20] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup service for internet applications," in *ACM SIGCOMM*, 2001.
[21] B. Y. Zhao, J. Kubiatowicz, and A. D. Joseph, "Tapestry: an infrastructure for fault-tolerant wide-area location and routing," tech. rep., April 2001.
[22] "FIPS 180-1. secure hash standard," *U.S. Department of Commerce/NIST, National Technical Information Service*, April 1995.
[23] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu, "OpenDHT: A public DHT service and its users," in *ACM SIGCOMM*, 2005.
[24] J. R. Douceur, "The Sybil attack," in *IPTPS*, (Cambridge, MA, USA), March 2002.
[25] "Java 2 platform, standard edition, v 1.4.2. API specification," http://java.sun.com/j2se/1.4.2/docs/api.
[26] T. Henderson, D. Kotz, and I. Abyzov, "The changing usage of a mature campus-wide wireless network," in *ACM MobiCom*, 2004.