Wifi-Reports:
Improving Wireless Network Selection with Collaboration

Jeffrey Pang
Carnegie Mellon University
jeffpang@cs.cmu.edu

Ben Greenstein
Intel Research Seattle
benjamin.m.greenstein@intel.com

Michael Kaminsky
Intel Research Pittsburgh
michael.e.kaminsky@intel.com

Damon McCoy
University of Colorado
damon.mccoy@colorado.edu

Srinivasan Seshan
Carnegie Mellon University
srini@cmu.edu

ABSTRACT

Wi-Fi clients can obtain much better performance at some commercial hotspots than at others. Unfortunately, there is currently no way for users to determine which hotspot access points (APs) will be sufficient to run their applications before purchasing access. To address this problem, this paper presents *Wifi-Reports*, a collaborative service that provides Wi-Fi clients with historical information about AP performance and application support. The key research challenge in Wifi-Reports is to obtain accurate user-submitted reports. This is challenging because two conflicting goals must be addressed in a practical system: preserving the privacy of users' reports and limiting fraudulent reports. We introduce a practical cryptographic protocol that achieves both goals, and we address the important engineering challenges in building Wifi-Reports. Using a measurement study of commercial APs in Seattle, we show that Wifi-Reports would improve performance over previous AP selection approaches in 30%-60% of locations.

Categories and Subject Descriptors:
C.2.1 Computer-Communication Networks: Network Architecture and Design

General Terms: Measurement, Design, Security

Keywords: privacy, anonymity, wireless, reputation, 802.11

1. INTRODUCTION

Users expect Internet connectivity wherever they travel and many of their devices, such as iPods and wireless cameras, rely on local area Wi-Fi access points (APs) to obtain connectivity. Even smart phone users may employ Wi-Fi instead of 3G and WiMAX to improve the performance of bandwidth intensive applications or to avoid data charges. Fortunately, there is often a large selection of commercial APs to choose from. For example, JiWire [6], a hotspot directory, reports 395 to 1,071 commercial APs in each of the top ten U.S. metropolitan areas. Nonetheless, users report that some APs block applications [10] and have poorer than advertised performance [24], so selecting the best commercial AP is not always straightforward.

Commercial Wi-Fi. To verify these reports, we present the first measurement study of commercial APs in hotspot settings. Previous war-driving studies [28, 32] performed Wi-Fi measurements from streets or sidewalks, whereas we measure APs from the perspective of a typical Wi-Fi user who is inside an establishment. Our study examines the performance and application support of all visible APs at 13 hotspot locations in Seattle over the course of 1 week. We find that there is indeed a wide range of AP performance even among APs very close to each other. Yet, there is currently no way for a user to determine which AP would be best to run his applications before paying for access.

Wifi-Reports. To address this problem, we present *Wifi-Reports*, a collaborative service that provides clients with historical information to improve AP selection. Wifi-Reports has two main uses: First, it provides users with a hotspot database similar to JiWire but where APs are annotated with performance information. Second, it enables users to more effectively select among APs visible at a particular location. Wireless clients that participate in Wifi-Reports automatically submit reports on the APs that they use. Reports include metrics such as estimated back-haul capacity, ports blocked, and connectivity failures. Using submitted reports, the service generates summary statistics for each AP to predict its performance. Obtaining accurate user-submitted reports poses two challenges:

(1) Location privacy: A user should not have to reveal that he used an AP to report on it. Otherwise he would implicitly reveal a location that he visits. Users may be reluctant to participate in Wifi-Reports if their identities can be linked to their reports. At the same time, however, a few users should not be able to significantly skew an AP's summary statistics because some may have an incentive to submit fraudulent reports, e.g., to promote APs that they own. One way to meet these conflicting goals is to assume the existence of a trusted authority that is permitted to link users to their reports in order to detect fraud (e.g., in the way that eBay manages user reputations). For good reason, users, privacy groups, and governments are becoming increasingly wary about malicious or accidental disclosures of databases that can track large numbers of people [12], even if they are tightly regulated like cell phone records [4]. Therefore, we present a

MobiSys'09, June 22–25, 2009, Kraków, Poland.
Copyright 2009 ACM 978-1-60558-566-6/09/06 ...$5.00.

report submission protocol that tolerates a few misbehaving users and does not require the disclosure of location related information to anyone, including the Wifi-Reports service. Our protocol leverages blind signatures to ensure that the service can regulate the number of reports that each user submits, but cannot distinguish one user's reports from another's.

(2) Location context: Physical obstructions and the distance between a client and an AP affect the quality of the wireless channel. Therefore, we must take location context into account when estimating AP performance or our estimates will not be accurate. We describe how measurements can be categorized by the different wireless channel conditions under which they were performed. We also describe how to index and retrieve reports based on location without requiring additional equipment such as GPS.

We have implemented the key components of Wifi-Reports and used our measurement study to simulate how well it would work. Our results suggest that even if a user is only selecting among APs at a single location, Wifi-Reports performs close to optimal in more cases than existing techniques such as best-signal-strength and best-open-AP [32] because it provides information on commercial APs that cannot be tested beforehand. Also, it outperforms the strategy of picking the "official" AP for a hotspot, because, for example, the AP next door may have a better back-haul connection.

Contributions.

1. To our knowledge, we are the first to study the attributes of commercial encrypted and "pay-for-access" APs in the wild. Although previous studies have examined open APs [28, 32] observed while war driving, we find that the best performing AP for a typical user in one commercial district is most often a closed AP.

2. We show that Wifi-Reports' summary statistics predict performance accurately enough to make correct relative comparisons between different APs, despite performance variability due to competing traffic. For example, it predicts AP throughput and response time to within a factor of 2 at least 75% of the time. Since different APs' median throughputs and response times differ by up to 50× and 10×, respectively, this prediction accuracy enables Wifi-Reports to select the best AP more often in more locations than any previous AP selection approach. Moreover, unlike previous AP selection approaches, Wifi-Reports enables users to examine the characteristics of APs that not in radio range, which is useful when users are mobile.

3. We present the design, implementation, and evaluation of a practical protocol that enables users to contribute reports on APs anonymously, and that generates accurate summary statistics for each AP even if 10% of that AP's users collude to promote it. Although we use this protocol in the context of Wifi-Reports, it is applicable to other collaborative reporting services.

The rest of this paper is organized as follows. §2 presents the results of our measurement study. §3 presents an overview of Wifi-Reports' design. §4 describes how it preserves privacy and mitigate fraud. §5 describes how it distinguish client locations. §6 presents an evaluation of Wifi-Reports. §7 presents related work and §8 concludes.

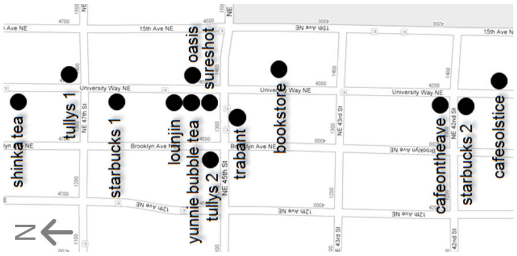

Figure 1—Measured hotspot locations near University Avenue, Seattle, WA

2. MEASUREMENT STUDY

We conducted a measurement study to determine whether existing AP selection algorithms are sufficient to choose an AP that meets a user's needs. We sought answers to three questions that illustrate whether this choice is obvious and whether it can be improved with Wifi-Reports.

Diversity. Is there diversity in terms of performance and application support of different hotspots' APs? The more diversity, the more likely a user will choose a hotspot with substantially suboptimal performance when selecting randomly from a hotspot directory.

Rankability. Is the best choice of AP at a particular location always obvious? If the best APs do not have any observable traits in common, then AP selection algorithms that use the same metric to rank APs at all locations will sometimes pick suboptimal APs.

Predictability. Is performance predictable enough so that historical information would be useful?

Our study examined hotspots around University Avenue, Seattle, WA, near the University of Washington. We believe this area is representative of commercial districts with multiple Wi-Fi service providers. It is less likely to be representative of areas that only have a single Wi-Fi service provider, such as in many airports. However, since users don't have a choice of AP providers in those environments, selecting a provider to use is straightforward. Wifi-Reports could, however, still help a user decide if purchasing access is worthwhile. Figure 1 shows the hotspot locations where we performed measurements, which included those listed in JiWire's database and some additional sites known to us.

All locations are single-room coffee or tea shops. Most APs we measured are not open. In addition to each hotspot's official AP, the APs of hotspots nearby are also usually visible. APs of the free public seattlewifi network are sometimes visible at all locations. APs belonging to the University of Washington network are sometimes visible due to proximity to campus buildings, though these were never the best performing at any location. Our study offers a lower bound on the number and diversity of APs, as more may become available.

2.1 Setup

Infrastructure. To emulate a typical user of Wifi-Reports, we collected measurements with a commodity laptop with an Atheros 802.11b/g miniPCI card attached to the laptop's internal antennas. We implemented a custom wireless network manager for associating to APs and performing measurements after association. Our implementation is based on the Mark-and-Sweep war driving tool [28].

Methodology. During each measurement trial at a location, we emulate a typical connection attempt by scanning for visible APs. We then attempt to associate and authenticate with each AP found (identified by its unique BSSID). If successful, we run our battery of measurement tests before moving on to the next AP. We manually obtain authentication credentials, if necessary (e.g., through a purchase). Since many Wi-Fi drivers do not list APs with low signal-to-noise (SNR) ratios, we only attempt to connect to APs when they appear with an SNR > 10 dB.[1]

We performed measurements at typical seating locations in each hotspot. Although the exact same location was not used for all measurements in a hotspot, §5 shows how well we can distinguish performance at different locations.

Time frame. Previous studies measured each AP at a single point in time [28, 32]. Since we want to know whether AP characteristics are predictable, we performed 8 to 13 measurements at each location (with the exception of yunnie bubble tea, where we only performed 6 trials). These measurements were taken during 7 week days in October 2008. On each day, at each location, we performed 1-2 measurements at different times of the day, so we have at least one measurement during each 2 hour time-of-day between 9AM and 6PM (or a narrower time window if the hotspot opened later or closed earlier).

2.2 Results

Basic connectivity. Figure 2(a) shows the fraction of times we were able to obtain connectivity from each AP at each location (i.e., association and authentication succeeds, we are able to obtain a DHCP address, and able to fetch `www.google.com`; we retry each step up to 3 times and for up to 30 seconds on failure). We only count times when the AP was visible in a network scan. The symbol above each point indicates whether the AP can be accessed for free (O) or not ($). The box for the official AP at each hotspot is shown in a solid color and its symbol is in a larger font.[2]

As expected, most (10 of 13) official hotspot APs were successful 100% of the time. However, some, such as the ones at tullys 1 and cafesolstice, failed several times. These were all DHCP failures and frequent users of cafesolstice say that the AP has always had DHCP problems. However, it would be difficult to detect these problems automatically because even to attempt to access the network, a user has to obtain a WPA password from the cashier. Although unofficial APs visible at hotspots tend to fail with higher regularity due to wireless loss, a few in most (8 of 13) locations succeed whenever they were visible in our network scan. Thus, even this very basic connectivity metric suggests that there is diversity.

TCP throughput. Adequate throughput is important for many applications, such as streaming video or VoIP. Figure 2(b) shows a box-plot of the TCP download throughput achieved through each AP (i.e., the bar in the middle of each box indicates the median; the ends of each box indicate the first and third quantiles; and whiskers indicate the minimum and maximum). Note the logarithmic scale. We measured throughput over the final five seconds of a ten-second transfer from a high bandwidth server under our control to estimate each AP's sustained throughput after TCP slow start. We do not count the times when we failed to associate with the AP or when TCP timed out during establishment (the failure rate above suggests how often this occurs), so we have fewer measurements for some APs than for others.

First, we note that there is a significant range in available capacities across different hotspot locations. Median capacities range from less than 100 Kbps (yunnie) to over 5 Mbps (starbucks 1 and oasis). There is variability in each AP's throughput measurements, which is attributable mostly to wireless loss or contention (similar UDP throughput measurements had less variability), but the variation at most APs is much smaller than this wide performance range. Therefore, there is diversity in AP capacity, and throughput is predictable enough to distinguish them.

Second, we observe that there is also a significant range in capacities among APs visible from a single location. As expected, most (9 of 13) official hotspot APs have the highest median throughputs at their respective locations. However, this is not true at tullys 1, yunnie, starbucks 1, and tullys 2, where better APs were available from an apartment building next door, the public seattlewifi network, a store next door, and a nearby hotel, respectively. Indeed, at starbucks 1 and yunnie, an unofficial AP always gave significantly more throughput than the official one when visible. Recall that these comparisons only include measurements when we were able to successfully pay for and obtain Internet connectivity, so a user without historical information would have to pay before discovering this.

Response time. Low network latency is another important attribute for interactive applications such as web browsing. To estimate the latency a typical web browser would experience, we measured the response time to one of the most popular web sites. Figure 2(c) shows a box-plot of the time to fetch `http://www.google.com`. Fetch time includes the time to perform a DNS lookup, which is dependent on the DNS server each AP's DHCP server assigns us.[3] Since Google's homepage is only 6KB, fetch time is dominated by latency rather than transfer time. We do not count the times when association failed.

Just as we saw with TCP throughput, there is diversity in response time, which ranges from less than 100 ms to several seconds. Response times of more than 1 second are typically noticeable by an end-user. As expected, most (10 of 13) official APs have the lowest median latency at their respective hotspot locations, but this is not true at tullys 1, yunnie, and cafeontheave. Only the disparity between the best and official APs at tullys 1 is large enough to be noticeable, but even smaller differences may impact more sensitive applications, such as VoIP. In addition, in some cases the AP with the lowest and least variable response time is not the same as the AP with the highest throughput (e.g., at starbucks 1), so ranking is dependent on application requirements. Finally, all official APs, except the one at sureshot, provide

[1] One physical AP at each starbucks advertised two virtual APs. Since we did not find any service differentiation between these virtual APs after login, we only include one of them in our study. They exist because Starbucks hotspots are migrating from T-Mobile to AT&T Wi-Fi.

[2] cafesolstice has 2 official APs because it changed APs in the middle of our measurement period. However, both APs suffered from basic connectivity problems.

[3] The CNAME and A DNS records for `www.google.com` have a TTLs of 1 week and 1 day, respectively, so they are almost always already cached at the DNS server.

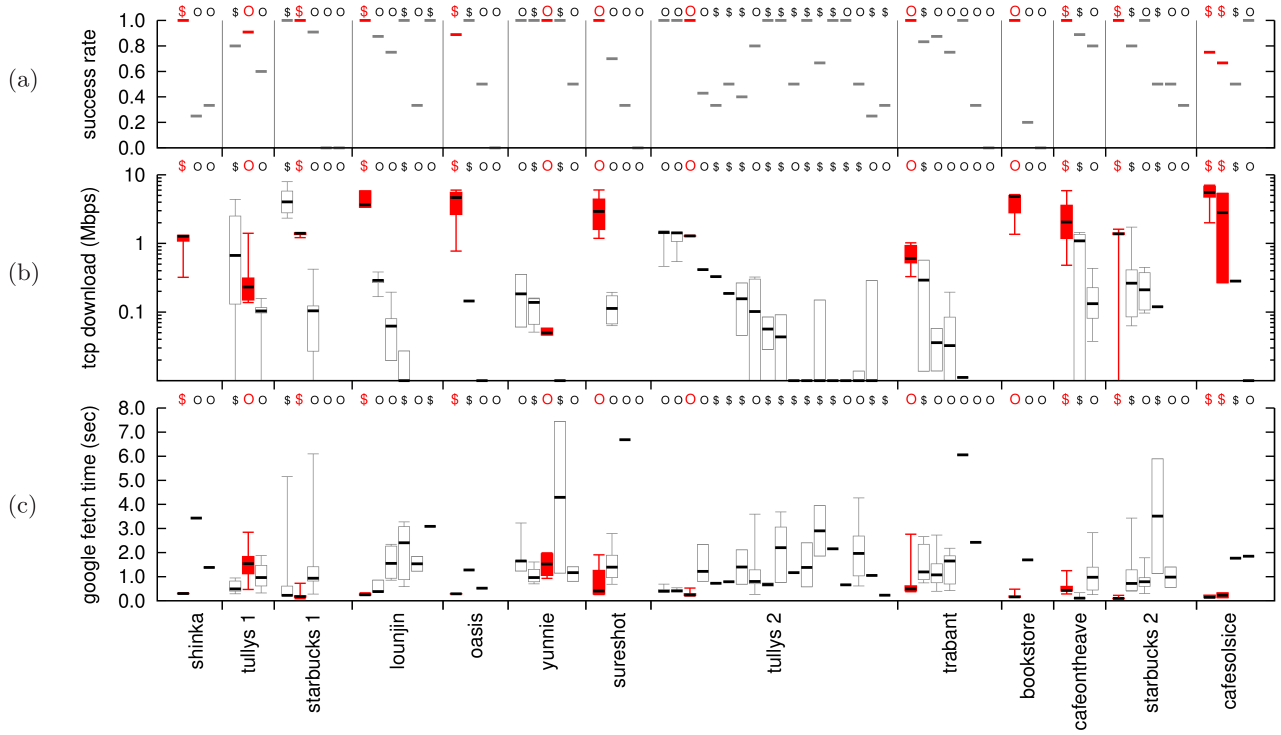

Figure 2—(a) The success rate of different APs (i.e., how often we could connect and access the Internet when each AP was visible). Each point represents one AP visible at each location. (b) A box-plot of the measured TCP download throughput through each APs. Note the logarithmic scale. (c) A box-plot of the time to fetch `http://www.google.com` using each AP. The measurements for each AP are grouped by the hotspot location where they were taken, shown on the x-axis. The symbol above each box indicates whether the AP can be accessed for free (O) or not ($). The box for the official AP at each hotspot is a solid color and its symbol is in a larger font. The APs in all graphs are sorted by their median TCP download throughput. Most of the non-free APs at tullys 2 are University of Washington APs in a building across the street.

predictable response times (first and third quantiles within a factor of 2). At least one unofficial AP at each location is just as predictable.

Port blocking. To determine if an AP blocked or redirected certain application ports, we sent 3 probes to each port on a measurement server under our control. For UDP ports, each probe consisted of 44-byte request and response datagrams, while for TCP ports, each probe tried to establish a connection and download ~32 bytes of data (in order to check for port redirection). We tested common application ports including: FTP, NTP, SSH, NetBIOS, SMTP, IMAP, SSL, VoIP (SIP), STUN, common VPN ports, World of Warcraft, Counterstrike, Gnutella, and Bittorrent. To account for packet loss, we conclude that a port is blocked only if it was never reachable in any of our measurements.

All APs blocked NetBIOS, most likely because they are configured to do so by default. Three APs blocked non-DNS packets on port 53 and only one (bookstore's official AP) blocked more ports: all non-privileged TCP ports and all UDP ports except DNS and NTP. Nonetheless, this is useful information, as common applications such as VPNs, VoIP, and games would not function.

Summary. With respect to *diversity*, we find that there is significant diversity in AP throughput and latency. With respect to *rankability*, the official AP is not the best choice at 30% of hotspot locations, so ranking APs is not always obvious. Finally, with respect to *predictability*, there is variability in performance over time, but this variability is much smaller than the range of different APs' performance, so historical information should be predictable enough to compare APs. Therefore, our results suggest that a collaborative reporting service may improve AP selection.

2.3 Discussion

Why not just use official APs? One might ask whether historical information is really necessary if the official AP is usually the best at 70% of locations. First, in §6.1, we show that historical information can get us the best AP in the remaining 30%. Second, as hotspot density increases, scenarios like these will likely become more common. Third, many users will be willing to move to find better APs and, without historical information, it is not obvious how to determine where to move to. Finally, if a user is not in range of any APs, he needs historical information to determine where to find a good one.

Other selection factors. In practice, users will likely take other factors into account besides AP performance and application support, such as cost and venue. Although these factors are important and reports in Wifi-Reports can include such information, they are also subjective, so we focus our evaluation in this paper on AP performance. In particular, we focus on download capacity and latency since these metrics are important for most applications. Our focus demonstrates Wifi-Reports' ability to help users make more informed decisions about which APs to use, whether they take cost and venue into account or not.

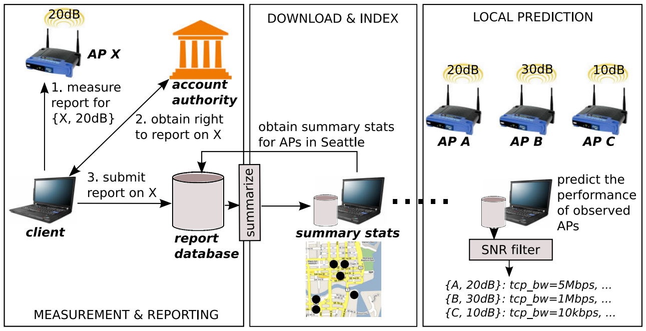
20dB
AP X
account authority
1. measure report for {X, 20dB}
2. obtain right to report on X
3. submit report on X
client
report database
summarize
MEASUREMENT & REPORTING
DOWNLOAD & INDEX
obtain summary stats for APs in Seattle
summary stats
LOCAL PREDICTION
20dB
30dB
10dB
AP A
AP B
AP C
predict the performance of observed APs
SNR filter
{A, 20dB}: tcp_bw=5Mbps, ...
{B, 30dB}: tcp_bw=1Mbps, ...
{C, 10dB}: tcp_bw=10kbps, ...

Figure 3—Wifi-Reports components and typical tasks.

3. WIFI-REPORTS OVERVIEW

Wifi-Reports is a recommendation system [14]. Users rate the services they use and submit these ratings to a report database where they are summarized. Other users download summarized ratings to evaluate services that they are considering. In Wifi-Reports, the users are wireless clients, services are APs, and ratings are key-value pairs of measured performance metrics.

3.1 Challenges

In contrast to previous recommendation systems, Wifi-Reports faces two unique challenges:

Location privacy. By reporting the use of an AP, a user implicitly reveals a location where he has been with an accuracy that is sufficient to identify sensitive places [33]. Thus, users may not be willing to participate in Wifi-Reports if their identities can be linked to their reports. A single user's reports must not even be linkable to each other, otherwise they are vulnerable to inference attacks [17, 27]. Nevertheless, we still want to limit the influence of malicious users that submit fraudulent reports, which is a common problem in recommendation systems [39, 41].

Location context. Clients will typically search for summaries by location (e.g., "all APs in Seattle"), and the location of a client with respect to an AP will affect its measured performance due to different wireless channel conditions. Since we would like clients to generate reports automatically, location context must be determined automatically.

3.2 System Tasks

The operation of Wifi-Reports consists of three main tasks (Figure 3). We present an overview of these tasks here. The next two sections describe how they can be done while addressing the challenges discussed above.

Measure and report. Clients measure and submit reports on APs that they use. For example, suppose a client attempts to connect to the Internet using AP X. If the connection fails (i.e., association, DHCP, or all TCP connections fail), the client generates the report {ap=**X**, SNR=20dB, date=11/14/2008, connectivity=false}.[4] If the connection succeeds, then the client software estimates performance metrics based on the user's network traffic or using active measurements when the connection is idle.[5] When measurement completes, it generates the report {ap=**X**, SNR=20dB, date=11/14/2008, connectivity=true, tcp_bw_down=100kbps, google_resp_time=500ms, ...}.

When the client has Internet connectivity again, it contacts an *account authority* to obtain the right to report on X, e.g., by receiving a credential. It sends this report along with the credential to a *report database*. An account authority is necessary to prevent a single malicious client from submitting an unbounded number of fraudulent reports. However, to preserve the location privacy of honest clients, neither the account authority nor the report database should learn that the client used AP X. We describe the novel protocol we use to address this problem in §4.

Download and index. The database generates summary statistics for each AP by summarizing the values for each key. To be robust against some fraudulent values, we use summary functions that are not significantly skewed by a small fraction of outliers. For example, median is used for real-value attributes (e.g., throughput), plurality voting for multinomial attributes (e.g., port blocking), and average for probability attributes with $\{0, 1\}$ inputs (e.g., basic connectivity). In addition, a summary indicates the number of reports that it summarizes as an estimate of its robustness (i.e., a user will pay more heed to a summary of 10 different reports than a summary of just 1 report). A client may choose to ignore summaries with too few reports to mitigate the impact of erroneous reports by early adopters.

Before traveling, a user downloads and caches the summary statistics of all APs in the cities that he expects to visit. In practice, client software would update this cache whenever it has connectivity, similar to the iPass [5] client. To find a suitable hotspot, reports are shown to a user on a map. In order to facilitate this operation, reports must be search-able by geographic location. Unfortunately, we cannot rely on GPS because many wireless clients are not equipped with it and it is often does not work indoors. We describe existing techniques that we leverage to obtain coarse geographic coordinates in §5.1.

Predict locally. Finally, when a user sits down at a cafe, he typically wants to find the best AP that is visible. Although the client will have downloaded summaries for these APs earlier, the expected performance of each AP depends on the wireless channel conditions between the client and the AP. For example, conditions will vary based on the observed signal-to-noise ratio (SNR). Therefore, the client must apply a filter to the summaries to obtain an accurate prediction for the current conditions. We describe how a client can perform this filtering in §5.2.

4. LOCATION PRIVACY

This section describes a novel report submission protocol that ensures *location privacy* and *limited influence*, properties that we define below. Define U to be the set of all users that participate in Wifi-Reports, S to be the current set of all APs, $u = \mathsf{submitter}(R)$ to be the user that submitted report R, and $s = \mathsf{target}(R)$ be the AP that R reports on. Suppose $C \subset U$ is the largest set of colluding malicious users that try to violate any user's location privacy or to influence an AP's summary.

[4] **X** refers to the AP's BSSID and a hash of its signing key described in §4.

[5] A number of techniques and tools exist to estimate bandwidth [34] and response time [3]. These techniques are outside the scope of this paper, but the measurements we used can be implemented as an anonymous speed test.

Location privacy. To preserve location privacy, we must satisfy three conditions. (1) No one, not even the account authority or report database, should be able to link any report to its submitter; i.e., no one should be able to guess $\mathsf{submitter}(R_i)$ with probability greater than $\frac{1}{|U \setminus C|}$, for all reports R_i. (2) No one should be able link any two reports together unless they were submitted by the same user for the same AP; i.e., no one should be able to guess whether $\mathsf{submitter}(R_i) = \mathsf{submitter}(R_j)$ with probability greater than $\frac{1}{|U \setminus C|}$, for all R_i, R_j where $\mathsf{submitter}(R_i) \neq \mathsf{submitter}(R_j)$ or $\mathsf{target}(R_i) \neq \mathsf{target}(R_j)$. (3) A user should not have to reveal the target of a report in order to obtain the right to submit the report; i.e., after obtaining the right to submit R_{k+1}, the account authority should not be able to guess $\mathsf{target}(R_{k+1})$ with probability greater than $\frac{1}{|S|}$. In practice, achieving this third condition may be too expensive, so we later relax it by restricting S to all APs in a city rather than all APs.

Limited influence. To limit the influence of dishonest users, exactly one report from each user who has submitted a report on AP s should be used to compute the summary statistics for s. To ensure that this condition is satisfied, any two reports submitted by the same user for the same AP must be linked; i.e., for all R_i, R_j where $\mathsf{submitter}(R_i) = \mathsf{submitter}(R_j)$ and $\mathsf{target}(R_i) = \mathsf{target}(R_j)$, anyone *should be able* to verify that $\mathsf{submitter}(R_i) = \mathsf{submitter}(R_j)$. When computing each summary, the database first summarizes each individual user's reports and then computes a summary over these summaries. This ensures that malicious users have at most $|C|$ votes on the final summary.

We may also want to limit the rate at which these users can submit reports on any AP. For example, we may want to prevent a malicious user from reporting on a large number of APs that he has never actually visited. We discuss how to achieve this additional property at the end of §4.3.

4.1 Threat Model

Users' location privacy should be protected from malicious users, the account authority, and report databases. To meet this goal, we don't assume any restrictions on the behavior of malicious users, but we make a few practical assumptions about the account authority and report databases.

Account authority. A challenge for recommendation systems is how to prevent malicious users from out-voting honest users, e.g., by using botnets or Sybil attacks to obtain many fake identities. Wifi-Reports, as with most existing recommendation systems, assumes that a central account authority can limit these large-scale attacks. For example, the authority can require a credential that is hard to forge, such as a token credit card payment or the reception of an SMS message on a real cell phone. These defenses are not perfect, but are enough of a deterrent that existing recommender systems work well in practice. These heuristics may also be supplemented by Sybil detection schemes (e.g., [40]). Thus, we assume that these mechanisms are sufficient to bound the number of malicious users to a small fraction of the total number of users. §6.3 shows that our system can limit the influence of this small number of malicious users. We assume that the account authority is honest but curious; that is, it may try to reveal information about users, but it does not violate our protocol. We discuss how selfish violations can be detected in the next two sections. Since the account authority is a high profile entity, we believe that the legal implications of violations are sufficient deterrents to prevent them.

Report databases. Users have to trust the report database to summarize reports correctly. To distribute this trust, we assume that there are multiple databases and that most are honest (e.g., do not delete reports prematurely). Honest users submit reports to all the databases and download summary statistics from all databases, using the report on each AP that the majority of databases agree upon. We note that the existence of a single honest database can be used to audit all databases, because any valid report that exists should exist on all the databases, and reports are independently verifiable (see the protocol below). Independent verifiability also means that each database can periodically check the others to discover and obtain reports that it is missing. We assume that users learn about the list of report databases in an out-of-band manner; e.g., it may be distributed with the software.

A report database can link reports if they are submitted from the same IP address. Therefore, we assume that users submit reports through a mix network such as Tor [23] and that the mix achieves its goal, i.e., no one can infer the source IP address of the sender's messages.

4.2 Straw Man Protocols

Anonymize reports. One approach might be to have users simply submit reports to the databases via a mix network. This means that all reports are unlinkable, thus providing location privacy. However, this protocol does not provide limited influence because a database can not distinguish when one user submits many reports on an AP versus when many users submit one report each on the AP.

Authenticate reports. For this reason, nearly all existing recommender systems today rely on a trusted central authority that limits each real user to a single account. We can limit influence with an authority A as follows: When a user u_i wants to submit a report R on AP s_j, it authenticates itself to A (e.g., with a username/password) and then sends R to A. A checks if u_i has previously submitted any reports on s_j and, if so, deletes them from the report databases before adding the new one. A explicitly remembers the user that submitted each report. If A is the only one allowed to add and remove reports from the report databases, this protocol provides limited influence because each user is limited to one report. However, it fails to provide location privacy with respect to A. Indeed, A *must* remember which reports each user submitted to prevent multiples.

4.3 Blind Signature Report Protocol

To achieve both location privacy and limited influence, Wifi-Reports uses a two phase protocol. We sketch this protocol here: First, when user u_i joins Wifi-Reports, the account authority A provides him with a distinct signed "token" K_{ij} for each AP $s_j \in S$. By using a blind signature [16], no one, including A, can link K_{ij} to the user or to any other $K_{ij'}$. This ensures location privacy. However, anyone can verify that A signed K_{ij} and that it can only be used for s_j. GenToken describes this step in detail below. Second, to submit a report R on AP s_j, u_i uses the token K_{ij} to sign R, which proves that it is a valid report for s_j. u_i publishes R to each report database anonymously via the mix network. Since u_i only has one token for s_j, all valid reports that u_i submits on s_j will be linked by K_{ij}. This ensures limited influence. SubmitReport describes this step in detail below.

Preliminaries. The RSA blind signature scheme [16] is a well known cryptographic primitive that we use in our protocol. Let $\mathsf{blind}(K, m, r)$ and $\mathsf{unblind}(K, m, r)$ be the RSA blinding and unblinding functions using RSA public key K, message m, and random blinding factor r (we use 1024-bit keys and values). Let $\mathsf{sign}(K^{-1}, m)$ be the RSA signature function using RSA private key K^{-1}, and let $\mathsf{verify}(K, m, x)$ be the RSA verification function, which accepts the signature x if and only if $x = \mathsf{sign}(K^{-1}, m)$. Let $H(m)$ be a public pseudorandom hash function (we use SHA-512). We leverage the following equivalence:

$$\mathsf{sign}(K^{-1}, m) = \mathsf{unblind}(K, \mathsf{sign}(K^{-1}, \mathsf{blind}(K, m, r)), r)$$

That is, blinding a message, signing it, and then unblinding it results in the signature of the original message.

Blind signatures have two important properties. (1) *Blindness*: without knowledge of r, $\bar{m} = \mathsf{blind}(K, m, r)$ does not reveal any information about m. (2) *Unforgeability*: suppose we are given valid signatures $(x_1, x_2, \ldots, x_k)$ for each of $(m_1, m_2, \ldots, m_k)$, respectively, where $m_i = H(\hat{m}_i)$. Without the secret key K^{-1}, it is infeasible to forge a new signature $x_{k+1} = \mathsf{sign}(K^{-1}, H(\hat{m}_{k+1}))$ for any $\hat{m}_{k+1} \neq \hat{m}_i$ for all i, under the assumption that the known-target or chosen-target RSA-inversion problems are hard [16]. However, anyone can check whether $\mathsf{verify}(K, H(\hat{m}_i), x_i)$ accepts.

Protocol description. Our protocol has two phases: GenToken and SubmitReport, described below. For now, assume that the set of APs S is fixed and public knowledge. We describe later how APs enter and leave this set.

GenToken(u_i, s_j)**.** The GenToken phase is used by user u_i to obtain a token to report on AP s_j and u_i only performs it once per s_j in u_i's lifetime. s_j identifies an AP by BSSID as well as a hash of A's signing key for that AP (see below), i.e., $s_j = \{bssid_j, H(bssid_j|M_j)\}$. We assume that u_i and A mutually authenticate before engaging in the following protocol (e.g., with SSL and a secret passphrase).

$$\begin{aligned}
A: &\quad \{M, M^{-1}\}, \{M_j, M_j^{-1}\}\ \forall s_j \in S, \\
&\quad msig_j \leftarrow \mathsf{sign}(M^{-1}, H(bssid_j|M_j))\ \forall s_j \in S \\
u_i: &\quad M, M_j, msig_j, \{K_{ij}, K_{ij}^{-1}\}, r \overset{R}{\leftarrow} \{0,1\}^{1024} \\
u_i: &\quad b \leftarrow \mathsf{blind}(M_j, H(K_{ij}), r) && (1) \\
u_i \rightarrow A: &\quad \texttt{"sig-request"}, s_j, b && (2) \\
A: &\quad sig'_{ij} \leftarrow \mathsf{sign}(M_j^{-1}, b) && (3) \\
A \rightarrow u_i: &\quad \texttt{"sig-reply"}, sig'_{ij} && (4) \\
u_i: &\quad sig_{ij} \leftarrow \mathsf{unblind}(M_j, sig'_{ij}, r) && (5)
\end{aligned}$$

The lines before step 1 show items that are obtained before the protocol begins. A has a single *master* RSA key pair M, M^{-1} and has generated a different *signing* RSA key pair M_j, M_j^{-1} for each s_j. $H(bssid_j|M_j)$ is signed by the authority's master key so that others can identify M_j as a signing key for $bssid_j$. M, M_j, and $msig_j$ are publicly known (e.g., given to users and databases by A when they join). u_i generates a new *reporting* key pair K_{ij}, K_{ij}^{-1} and a 1024-bit random value r. After step 2, A checks whether it has already sent a `sig-reply` message to u_i for s_j. If so, it aborts, otherwise it continues. After step 5, u_i checks that $\mathsf{verify}(M_j, H(K_{ij}), sig_{ij})$ accepts. At completion, u_i saves K_{ij}, K_{ij}^{-1}, and sig_{ij} for future use.

This exchange can be described as follows: A authorizes the reporting key K_{ij} for use on reports for s_j by blindly signing it with s_j's signing key M_j^{-1}. By blindness, A does not learn K_{ij}, only that the client now has a key for s_j. Thus, no one can link K_{ij} to user u_i or to any $K_{il}, l \neq j$. $\{K_{ij}, sig_{ij}\}$ is the token that u_i attaches to reports on s_j. When a report is signed with K_{ij}^{-1}, this token proves that the report is signed with an authorized signing key. Since A only allows each user to perform GenToken once per AP, each user can only obtain one authorized reporting key for s_j. By unforgeability, even if multiple users collude, they cannot forge a new authorized reporting key.

SubmitReport(u_i, s_j, R)**.** This phase is used by user u_i to submit a report R on AP s_j after a token for s_j is obtained. Let $\{D_1, \ldots, D_m\}$ be the m independent databases. R is submitted to each D_k as follows.

$$\begin{aligned}
D_k: &\quad M, M_j\ \forall s_j \in S \\
u_i: &\quad rsig \leftarrow \mathsf{sign}(K_{ij}^{-1}, H(R)) && (6) \\
u_i \rightarrow D_k: &\quad \texttt{"report"}, s_j, K_{ij}, sig_{ij}, R, rsig && (7)
\end{aligned}$$

The message in step 7 is sent through a mix network so it does not explicitly reveal its sender. After step 7, D_k checks that $\mathsf{verify}(M_j, H(K_{ij}), sig_{ij})$ and $\mathsf{verify}(K_{ij}, H(R), rsig)$ both accept. If any of these checks fail, the report is invalid and is discarded. In other words, u_i anonymously publishes a report R signed using K_{ij}^{-1}. By including $\{K_{ij}, sig_{ij}\}$, anyone can verify that the signature is generated using a key signed by M_j^{-1}, i.e., a key that A authorized to report on s_j during the GenToken phase.

Anonymizing GenToken. This protocol achieves limited influence and prevents each report from being linked to any user or any other report. However, if a user engages in GenToken(u_i, s_j) only when it reports on s_j, then it reveals to A that it is reporting on s_j. In order to satisfy the third condition of our location privacy requirement, that A cannot guess the AP with probability greater than $\frac{1}{|S|}$, u_i would have to perform GenToken on all $s \in S$ before submitting any reports so that A cannot infer which tokens were used.

When performing GenToken on all APs is too expensive, we relax this condition as follows. We allow A to infer that the AP is in a smaller set $\hat{S} \subset S$. Determining an appropriate set $\hat{S}$ is a trade-off between more location privacy and less time spent performing GenToken operations. We have users explicitly choose a region granularity they are willing to expose (e.g., a city). When reporting on an AP, they perform GenToken on all APs in this region. We believe this small compromise in location privacy is acceptable since users already volunteer coarse-grained location information to online services (e.g., to get localized news) and IP addresses themselves reveal as much. In §6, we show that using the granularity of a city is practical.[6]

Handling AP churn. To support changes in the set of APs S, A maintains S as a dynamic list of APs. Any user can request that A add an AP identified by BSSID

[6] An alternative solution is to perform GenToken on a random subset of n APs in addition to the target AP. However, since a user will likely submit reports on multiple correlated APs (e.g., APs in the same city), A can exploit correlations to infer the APs actually reported on.

and located via beacon fingerprint (see §5.1). A generates a new signing key pair and its signature $\{M_j, M_j^{-1}\}, msig_j \leftarrow \mathsf{sign}(M^{-1}, H(bssid_j|M_j))$, and the new AP is identified by $s_j = \{bssid_j, H(bssid_j|M_j)\}$. M_j and $msig_j$ are given to the user and he submits them along with the first report on s_j to each report database. AP addition is not anonymous, as the user must reveal the AP to A, so Wifi-Reports will initially depend on existing hotspot and war driving databases and altruistic users to populate S. However, over time we believe that owners of well-performing APs will be incentivized to add themselves because otherwise they will not receive any reports. An AP is removed from S if it is not reported on in 3 months (the report TTL, see below) and A sends a revocation of their signing keys to each database. Users can thus obtain new signing public keys and revocations from each database.

We take three steps to limit the impact of nonexistent or mislocated APs that malicious users may add. (1) When searching for APs on a map, the client report cache filters out APs that only have a small number of recent reports; these APs require more "locals" to report on them before distant users can find them. (2) After a sufficient number of reports are submitted, reported locations are only considered if a sufficient number are near each other, and the centroid of those locations is used. (3) A rate limits the number of additions each user can make.

Handling long-term changes. AP performance can change over time due to back-haul and AP upgrades. However, these changes typically occur at timescales of months or more. Thus, reports have a time-to-live (TTL) of 3 months. Databases discard them afterward. Determining the most appropriate TTL is a trade-off between report density and staleness and is a subject of future work.

Handling multiple reports. Our protocol allows u_i to submit multiple reports on s_j, which can be useful if they are from different vantage points or reflect changes over time; however, each report on s_j will be linked by the key K_{ij}. To ensure limited influence, a database needs to summarize each user's reports on s_j before computing a summary over these individual summaries. For simplicity, it computes an individual user's summary as just the most recent report from that user that was taken in the same channel conditions (see §5.2).[7] As a consequence, there is no need for an honest user to submit a new report on s_j unless the last one it submitted expired or if s_j's performance substantially changed. This approach also allows a client to mitigate timing side-channels (discussed below) by randomly dating his reports between now and the date in his last report on s_j without changing s_j's summary statistics.[8]

Rate limiting reports. As mentioned earlier, it may also be desirable to limit the rate at which an individual user can submit reports, say, to at most t reports per week. This can be accomplished with a straight forward extension of the SUBMITREPORT stage of the protocol: A keeps count of the number of reports that each user submits this week. Before submission of $report = \{s_j, K_{ij}, sig_{ij}, R, rsig\}$ (step 7), user u_i sends $h = \mathsf{blind}(M, H(report), r)$ to A. If u_i has not already exceeded t reports this week, A sends $lsig' = \mathsf{sign}(M^{-1}, h)$ back to u_i, and u_i unblinds $lsig'$ to obtain $lsig = \mathsf{sign}(M^{-1}, H(report))$. $lsig$ is included in the report submitted to the report databases and is verified to be correct by recipients. The user would submit the report to the database at a random time after obtaining $lsig$, so A would only be able to infer that it was requested by some user in the recent past, but not which one.

10-20 would be reasonable values for t; analysis of Wi-Fi probes shows most clients have not used more than 20 APs recently [26]. This approach only adds 4 ms of computational overhead on A per report submitted (see §6.2).

4.4 Discussion

BSSID spoofing. One obvious concern is that some APs can change their BSSID identities. For example, a poorly performing AP might spoof the BSSID of a good AP to hijack its reputation. Ideally, each AP would have a public key pair to sign its beacons. APs could then be identified by the public key instead of BSSID to prevent spoofing. In 802.11, APs can offer this signature and its public key as part of a vendor-specific information element or as part of 802.1X authentication. Without public key identities, we can still mitigate spoofing with two techniques: First, if an AP masquerades as another AP that is geographically far away, then reports on each will be summarized separately as distinct APs and users will treat them as such. Second, if an AP attempts to spoof one that is nearby, the distribution of beacon SNRs that users receive will likely have two distinct modes. This at least enables users (and the original AP) to detect spoofing, though resolution requires action in the "real world" since the 802.11 protocol cannot distinguish the two APs. Finally, laws against device fraud (e.g., [11]) may be a sufficient deterrent in practice.

Eclipse attacks. If A only reveals s_j to a single user u_i, A will know that any report for s_j is submitted by u_i. Therefore, u_i's view of the set of APs S is obtained from the report databases rather than from A. Recall that the identity of $s_j = \{bssid_j, H(bssid_j|M_j)\}$ is added to each database when s_j is added to S. Because a malicious database colluding with A could tie $bssid$ to a different signing key $M_{j'}$, clients only consider AP identities that the majority of report databases agree upon.

Side-channel attacks. Side-channels exposed in reports may potentially link reports if the adversary has additional information. For example, if only one user visits an AP on a given day, the AP can infer that any report with a timestamp on that day is from that user. If a user submits many reports on APs at a time when most users rarely submit reports, the receiving database may infer from the submissions' timing that they are linked. Since we add a small amount of noise to timestamps and submission times, we believe we can defeat most of these attacks in practice without significantly degrading accuracy.

[7] A more sophisticated summarization algorithm might use the mean or median values of all a user's reports, weighted by report age. We leave the comparison of summary functions to future work as we do not yet know how many reports real users would submit on each AP.

[8] If the owner of K_{ij} is ever exposed, then an adversary learns some approximate times when u_i used s_j. If u_i knows this, he can prevent any further disclosures by proving to A that he revoked K_{ij} and obtaining a new token for s_j using GENTOKEN; i.e., u_i can send $\{\texttt{"revoke"}, u_i, K_{ij}, ksig\}$ to A and the databases, where $ksig \leftarrow \mathsf{sign}(K_{ij}^{-1}, H(\texttt{"revoke"}|u_i|K_{ij}))$, which proves that u_i has K_{ij}'s secret key and that K_{ij} (and all reports signed with it) is revoked.

5. LOCATION CONTEXT

This section describes how Wifi-Reports obtains geographic coordinates for reports and how summary statistics are filtered by wireless channel condition.

5.1 Geographic Positioning

To obtain coarse geographic coordinates for APs, we leverage previous work on beacon "fingerprints." The set of Wi-Fi beacons and their signal strengths observed from a location can be used to obtain geographic coordinates with a median accuracy of 25 meters when paired with a sufficiently dense war driving database [31]. Existing war driving databases are sufficient to facilitate this task (e.g., Skyhook [7] is used to geolocate iPods). Thus, Wifi-Reports clients include estimated coordinates in reports. To generate the location estimate in summary statistics for each AP, the database uses the centroid of all reported positions that are close together (e.g., within two city blocks). Although these positions may be off by tens of meters, we believe that they are sufficiently accurate for locating areas of connectivity on a map. Network names can be correlated with business names to improve accuracy (e.g., from Google Maps), but doing this is outside the scope of this paper. We note that coordinates are only needed to allow clients to search for AP summary statistics by location.

5.2 Distinguishing Channel Conditions

Wireless performance differs based on channel conditions, which vary based on fine-grained location and environmental conditions. The loss rate of a wireless channel is roughly inversely proportional to the SNR, barring interference from other stations or multi-path interference [29]. The most obvious approach is to use summary statistics that only consider the k reports with SNR values closest to the currently observed SNR. However, this approach has two problems. First, it requires users to download a different summary for each possible SNR value for each AP. Second, it may not be possible to choose an appropriate k: if k is too large, summaries will consider many irrelevant reports; too small and summaries become vulnerable to outliers and fraud.

Fortunately, the continuum of SNR values can be partitioned into three ranges with respect to wireless loss: a range where clients experience near 100% loss, a range where clients experience intermediate loss, and a range where clients experience near 0% loss [29]. Therefore, Wifi-Reports categorizes reports based on these three channel conditions. In other words, clients measure the median SNR of beacons sent by their AP. Reports are annotated with this median SNR. When a client makes a local prediction about an AP, it considers only previous reports taken in the same SNR range. In practice, the database creates one summary for each of the three ranges for each AP, so the client does not need to download all the reports for an AP.

Since measured SNR depends on the AP's transmit power, these three SNR ranges may be different for each AP. We estimate these ranges as follows: Typical scenarios exhibit an intermediate loss range of 10 dB [29], so we exhaustively search for the "best" 10 dB range that satisfies the expected loss rates. Specifically, let $\overline{t_>}$ be the mean measured throughput of reports taken with SNR larger than the 10 dB range, $\overline{t_=}$ be the average throughput of reports with SNR in the 10 dB range, and $\overline{t_<}$ be the average throughput of reports with SNR smaller than the 10 dB range. We find the 10 dB range that maximizes $(\overline{t_>} - \overline{t_=}) + (\overline{t_=} - \overline{t_<})$, or the differences between the mean throughput in the three ranges.[9] We assume that reports of connectivity failures experienced 100% loss (i.e., have throughput of 0). Finally, if $\overline{t_<} < 0.75 \cdot \overline{t_=}$, we likely only have measurements in one of the 100% or 0% loss ranges, so we put all measurements in a single range.

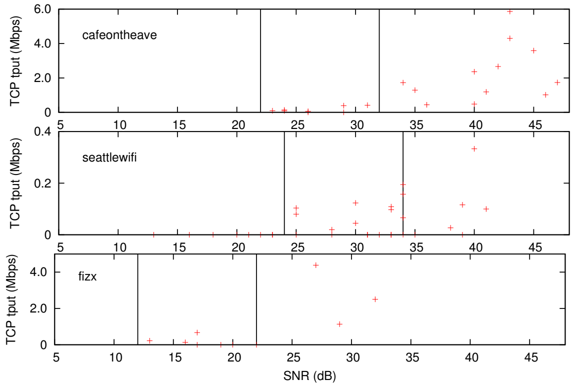

Figure 4—Estimated 100%, intermediate, and 0% loss regions for three APs in our measurement study.

Figure 4 shows the estimated ranges for several APs in our measurement study that were visible from multiple locations. We note that we do not need the distinguishing algorithm to work perfectly to obtain accurate predictions. There is already measurement noise within a single loss region due to TCP's sensitivity to loss. Thus, very inaccurate summaries typically only arise due to mixing reports in the 0% loss region with the 100% loss region so it usually suffices to estimate these regions within 10 dB. Clients could also directly measure wireless loss, either by observing other users' traffic [38] or by actively probing each AP.

5.3 Discussion

Client calibration. We use SNR to differentiate wireless channel conditions, but the reported SNR may have a bias due to manufacturing defects in Wi-Fi NICs. Therefore, different clients need to calibrate their reported SNR values. Previous work suggests that most of this error may be eliminated using a locally computed offset [29]. Reported SNR values for most cards after self-calibration may vary by 4 dB, a bias unlikely to affect our algorithm's accuracy significantly because the transitions between each SNR range are not sharply defined. To further improve accuracy, we can leverage existing self-calibration techniques that determine the biases of sensors (e.g., [15]). Implementing a distributed calibration algorithm is the subject of future work.

Other environmental factors. To improve prediction accuracy further, existing techniques can be used to measure and take into account other environmental factors that cause variation, such as multi-path interference and wireless contention [36, 38]. However, we found that contention is rare in our measurement study, so prediction accuracy is good even discounting these factors (see §6).

[9] When we have more than a few samples (i.e., ≥ 5), we use the median rather than the mean because it is more robust to outliers. Since the distribution of noise is likely Gaussian, the median is likely to be close to the mean.

User and AP mobility. To localize reports, we currently assume that users and APs are stationary. If users are mobile, performance may change over time; we can detect user mobility by changing SNR values. Our current set of active measurements are short-lived and can thus be associated with the SNR values observed when they are measured. Geolocating these mobile APs (e.g., those on a train) in a manner that makes sense is an area of future work.

6. EVALUATION

We evaluate the utility and practicality of Wifi-Reports using our measurement study (see §2) and our implementation of the reporting protocol (see §4). This section presents our evaluation of three primary questions:

- Some APs' performance changes over time and at different locations. Are reports accurate enough to improve AP selection?
- Our reporting protocol provides location privacy at the cost of token generation overhead. Can Wifi-Reports provide users with a reasonable amount of location privacy with practical token generation overheads?
- A determined attacker may be able to trick the account authority into giving it a few accounts or collude with his friends to submit multiple fraudulent reports on an AP. How tolerant are summaries to such attacks?

6.1 AP Selection Performance

Setup. We use our measurement study to simulate two scenarios: First, we evaluate the scenario where a user chooses which hotspot to go to physically based upon the predicted performance of all hotspots nearby. In this scenario, a user is primarily interested in *prediction accuracy*; i.e., we want $\mathsf{predict}(s)/\mathsf{actual}(s)$ to be close to 1 for each AP s, where $\mathsf{predict}(s)$ is the predicted performance (e.g., throughput) of s and $\mathsf{actual}(s)$ is the actual performance of s when it is used. Second, we evaluate the scenario where the physical location is fixed (e.g., the user is already sitting down at a cafe) but the user wants to choose the AP that maximizes performance. This situation is comparable to the traditional AP selection problem [32, 36, 38]; i.e., given the set of visible APs $V = \{s_1, s_2, \ldots, s_n\}$, we want a selection algorithm $\mathsf{select}(\cdot)$ that maximizes $\mathsf{actual}(\mathsf{select}(V))$, where $s = \mathsf{select}(V)$ is the AP we choose. In this scenario, a user is primarily interested in relative *ranking accuracy*; e.g., for throughput, we would like to maximize $\mathsf{actual}(\mathsf{select}(V))/\max_{s \in V}(\mathsf{actual}(s))$. In Wifi-Reports $\mathsf{select}(V) = \mathrm{argmax}_{s \in V}(\mathsf{predict}(s))$.

We simulate these scenarios using our measurement study as ground truth. That is, we assume that after the user selects an AP s to use, $\mathsf{actual}(s)$ is equal to one of our measurements of s. We evaluate performance over all our measurement trials. To simulate the $\mathsf{predict}(s)$ that would be generated by Wifi-Reports, we assume that all measurement trials except those for APs currently under consideration, are previously submitted reports. The reports for s are summarized to generate $\mathsf{predict}(s)$. This assumption implies that reports are generated by users that visit locations and select APs in a uniformly random manner. This is more likely to be the case when there are not yet enough reports in the system to generate any predictions. By counting devices associated with each AP in our measurement study, we observed that some users do currently use suboptimal APs. Thus, we believe that such reports would be obtained when bootstrapping new APs in Wifi-Reports.

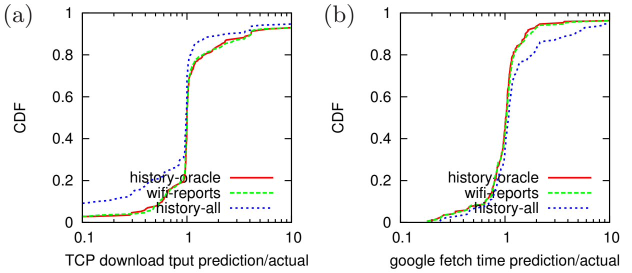

Figure 5—CDF prediction accuracy for (a) TCP download throughput and (b) Google fetch time over all trials on all official APs at their respective hotspots. Note the logarithmic scale on the x-axis.

Prediction accuracy. Figure 5 shows CDFs of prediction accuracy over all trials of official hotspot APs for TCP download throughput and Google response time. The x-axis in each graph shows the ratio of the predicted value over the actual achieved value. Values at 1 are predicted perfectly, values less than 1 are underestimates, and values more than 1 are overestimates. We compare three approaches for generating summary statistics. history-oracle shows the accuracy we would achieve if each summary summarizes only reports taken at the same hotspot location as the location under consideration; this requires an "oracle" because we would not automatically know the logical location where measurements are taken in practice. wifi-reports shows the accuracy when using Wifi-Reports' SNR filter before summarizing reports (see §5). history-all shows the accuracy when we summarize all reports to generate a prediction, regardless of the location where they were taken (e.g., even if the user is at Starbucks, the prediction includes reports of the same AP taken across the street).

In this graph, we focus on official APs, where we are sure to have some measurements in the 0% loss region, to better illustrate the impact of different channel conditions. Users in this scenario are more likely to desire a comparison of the 0% loss predictions rather than predictions in all three wireless channel conditions since they are choosing where to go. If an association or connection fails, we mark that trial as having 0 throughput and infinite response time. Recall that the summary function is median.

The graphs show that history-all underestimates TCP bandwidth and overestimates Google fetch time more often than history-oracle. This is because by including reports taken in the intermediate and near-100% loss regions, the median will generally be lower. In contrast, wifi-reports performs about as accurately as history-oracle, demonstrating that our SNR filter works well when we have some measurements in the 0% loss region. Furthermore, we note that at least 75% of predictions for both metrics are within a factor of 2 of the achieved value, while Figure 2 shows that the difference in the median throughputs and response times of official APs can be up to 50× and 10×, respectively. Therefore, most predictions are accurate enough to make correct relative comparisons.

Ranking accuracy. We now examine the scenario when a user is choosing between APs at a single location. Figure 6(a) and (b) show box-plots of achieved throughput and

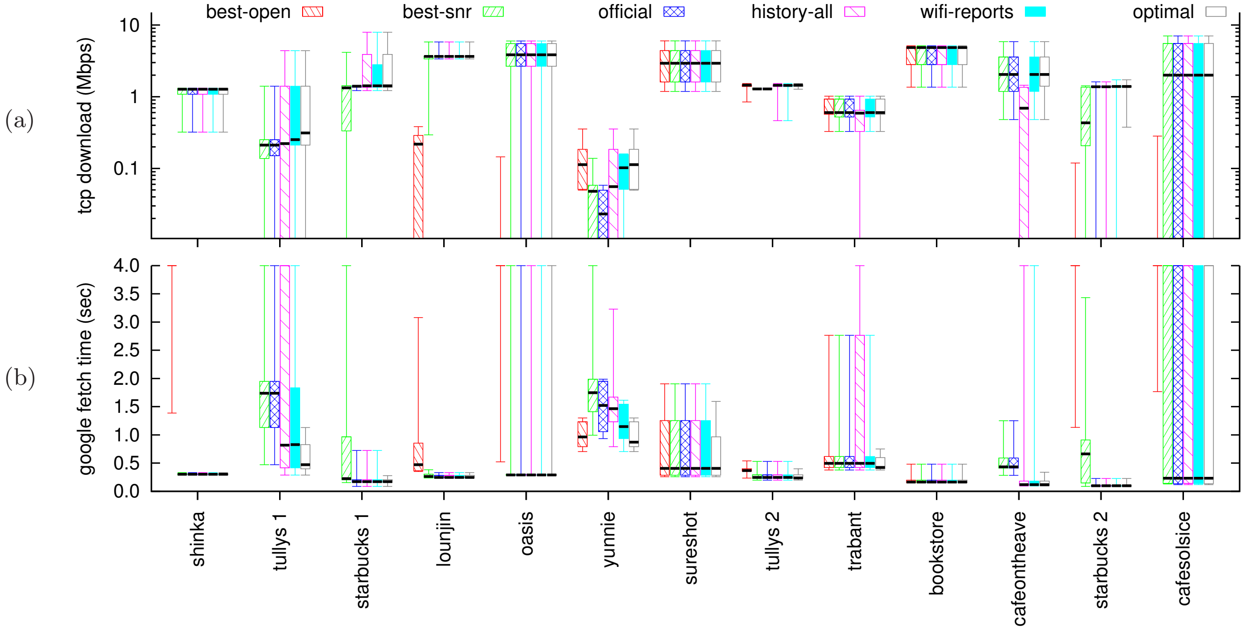

Figure 6—(a) Box-plot of achieved TCP download throughput when using each of five AP selection algorithms at each location. Note the logarithmic scale. Missing boxes for the best-open algorithm are at 0. (b) Box-plot of the achieved response time of `http://www.google.com` using each of five AP selection algorithms at each location. The whiskers that extend to the top of the graph actually extend to infinity (i.e., the fetch failed). missing boxes for the best-open algorithm are also at infinity. Each group of boxes are ordered in the same order as the key at the top.

response time, respectively, when using one of several AP selection strategies to try to achieve the best performance at each location. best-open simulates Virgil [32], an algorithm that associates with and probes all open APs before selecting the best one. best-snr simulates the most common algorithm of picking the AP with the highest SNR value. This algorithm works well when wireless channel quality is the limiting factor. official simulates using the "official" AP of each location. We expect this algorithm to work well since we showed in §2 that the official AP is the best at most locations. Obviously this approach would not work at locations without an official AP. history-all simulates Wifi-Reports without the SNR filter. wifi-reports simulates Wifi-Reports. history-all and wifi-reports only generate a prediction for an AP if we have at least 2 reports to summarize; if no predictions for any AP are generated, they fall back to selecting the official AP. Finally, optimal shows the best performance achievable.

best-open performs the worst overall, failing to achieve any connections at tullys 1, starbucks 1, and cafeontheave since no open APs were visible. best-open performs better than all other algorithms only at yunnie, where most of the APs were open. We note that best-open is qualitatively different than the other selection algorithms because it cannot select any closed AP; we include it only to demonstrate that restricting the choice of APs to open ones often results in substantially suboptimal performance. Furthermore, best-open also has more overhead (linear in the number of open APs visible) than the others because it must actively test each AP.

history-all again demonstrates the need for the SNR filter. Without the SNR filter, Wifi-Reports would achieve poorer performance than official or best-snr at least 25% of the time at tullys 1, trabant, and cafeontheave.

In contrast, wifi-reports achieves performance closest to optimal for both metrics in all cases except for two. It achieves worse TCP throughput than best-open once at yunnie and worse response time than best-snr or official once at cafeontheave. In each of these cases, the AP chosen by wifi-reports experienced an association or DHCP failure. However, a real client would quickly fall back to the second best AP chosen by wifi-reports, which was the optimal one. Furthermore, wifi-reports is able to achieve higher bandwidth more of the time than all other algorithms at yunnie and starbucks 1 and better response time more of the time than all other algorithms at tullys 1 and cafeontheave. Thus, it performs strictly better in more locations when compared with each of the other approaches individually.

Finally, we note that unlike all other approaches, Wifi-Reports enables users to rank APs that are nearby but not visible. This is useful when users are willing to move to obtain better connectivity.

6.2 Report Protocol Performance

We implemented our reporting protocol (§4) in software to evaluate its practicality. We present measurements of its processing time, total token fetch time, and message volume using workloads derived from actual AP lists. We focus on the token generation phase (GenToken) since, given a desired level of location privacy, its performance depends on actual densities of APs. The report submission phase (SubmitReport) runs in constant time per report and uses standard fast RSA primitives.

Setup. We emulate a client that obtains the right to report on APs while at home (e.g., before or after traveling). Our client has a 2.0 GHz Pentium M and our account authority server used one 3.4GHz Xeon processor (the software is single threaded). Both run Linux and all cryptography operations used openssl 0.9.8. The bottleneck link between

	mean	min	max	std dev	description
Server	58.918	33.18	421.26	59.056	generate key
Server	3.979	3.87	6.29	0.222	sign
Client	95.517	18.00	560.45	47.364	generate key
Client	0.150	0.14	22.21	0.222	verify
Client	0.058	0.03	1.43	0.134	unblind
Client	0.006	0.00	1.88	0.027	hash
Client	0.003	0.00	1.88	0.019	blind

Table 1—Microbenchmarks of cryptographic processing times. All keys are 1024 bit RSA keys and SHA-512 is used as the hash function. All values in milliseconds with a resolution of 10 microseconds. 1000 trials were executed.

the client and server is the client's cable Internet connection (6 Mbps down, 768 kbps up). The round trip time from client to server is 144 ms.

Processing time. Table 1 presents microbenchmarks of each step of the protocol. All times are in milliseconds. The most heavyweight steps are the generation of 1024 bit RSA keys by both the client (K_{ij}) and server (M_j).[10] However, both keys can be generated anytime beforehand so these operations need not be executed inline in the GenToken protocol. The remaining steps must happen inline, but have very low processing times. A server can sign a blinded message in under 4 ms, so it can process about 250 tokens per second, while a client can perform the verification and unblinding steps in roughly 0.2 ms, or 5000 times per second.

Token fetch time. A user who wants to obscure his locations within a region must perform GenToken on all APs in that region. Figure 7(a) shows the end-to-end time to fetch tokens for all APs in each of the ten cities that JiWire [6] reports to have the most APs (as of November 15, 2008). JiWire lists commercial APs that service providers or users have manually added, which parallels how most APs are added to Wifi-Reports. Nonetheless, some commercial hotspots may not be listed by JiWire, so this graph serves to establish a lower bound for cities with many APs. Since a user can fetch these tokens at any time before submitting a report, even the longest delay, 5.5 seconds for all of New York, is completely practical. Even obtaining tokens for several cities at once is practical since each client only does this once in its lifetime.

WiGLE [9] is a database of all APs that war drivers have overheard, including both commercial and private APs. Figure 7(b), presents fetch times for all WiGLE APs in a 32 km square centered at each city. Since most APs listed are not intended to be used by the public (e.g., home APs) and WiGLE does not filter out erroneous or stale measurements, this graph serves as a loose upper bound on fetch times. Even so, the worst fetch time (Seattle) is 20 minutes. Since a client can batch `sig-request` messages for multiple APs, a reasonable approach would be to request all tokens and then retrieve them at a later time. In addition, by choosing a region granularity of less than a city, a client can achieve much better delay and still mask his locations to a reasonable extent. Figure 7(c) shows the CDF of number of WiGLE APs in 1km^2 areas in each of the cities. Most cells in all cities have fewer than 188 APs, which only takes about 1 second to fetch, and no cell has more than 7400, which only takes about 30 seconds to fetch. Since commercial areas in most cities are not spread out, most will be covered by a small number of cells. Finally, we note that the server can parallelize the generation of each token to improve performance.

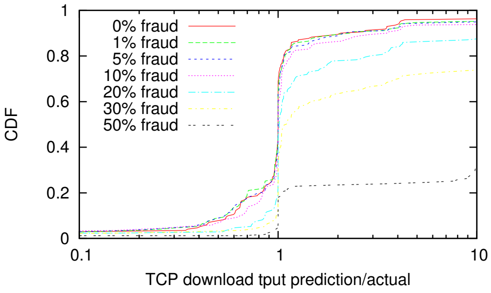

Figure 8—CDF of prediction accuracy for TCP download throughput of all official APs at their respective hotspots. We vary the percentage of fraudulent reports that claim throughput is 54Mbps. Note the logarithmic scale on the x-axis.

Message volume. A request for tokens transmits 173 bytes per token, while the response transmits 529 bytes per token. Therefore, our protocol is CPU-bound on the server even for a client on a cable modem. For example, it takes our client 8.7 minutes to send all requests for Seattle APs on WiGLE and 3.4 minutes to receive the replies (these latencies are included in the token fetch times reported above).

Admission rate and server cost. We next estimate the rate at which users can join given limited server resources. To simulate "average" American users joining the system, we assume that each user requests all tokens from one of the cities shown in Figure 7, chosen at random weighted by each city's population (according to 2007 U.S. census data [37]). While a user may request more, the authority rate limits each user to prevent denial-of-service attacks.

Suppose the authority has x CPUs. For JiWire APs, it can admit 27,455x new users per day. For example, if the authority has 100 CPUs, it can admit the entire population of these cities in 5.6 days. How much would this overhead cost over a system that stores reports without privacy? If deployed on Amazon's EC2 [1], this would only cost about 0.02 cents per user for CPU and bandwidth resources. For all WiGLE APs, the authority can admit 165x new users per day and the overhead cost would be about 2.6 cents per user. This one-time cost is a very small fraction of the $5+ each user would have to spend to use most commercial APs just for one day. There are also recurring costs incurred for computing tokens for new APs that are added and, if enabled, signing reports for rate limiting (see the end of §4.3). However, these costs are also trivial. For example, even if 10 new hotspots appear in each city every week and every user submits 10 new reports per week, the recurring cost would only be about 0.02 cents per user per year.

6.3 Resistance to Fraud

Summary values are robust to fraudulent reports that try to boost or degrade an AP's value because we use summary functions that are resilient to outliers. However, since there is variability in honest reports as well, a small number fraudulent reports may still be able to degrade prediction accuracy, e.g., by shifting the median higher or lower.

Setup. We consider the same scenario as in §6.1. To evaluate the extent that fraudulent reporting can degrade accuracy, we simulate an adversary that tries to boost the pre-

[10] The standard deviation for key generation is high because the algorithm has a random number of iterations.

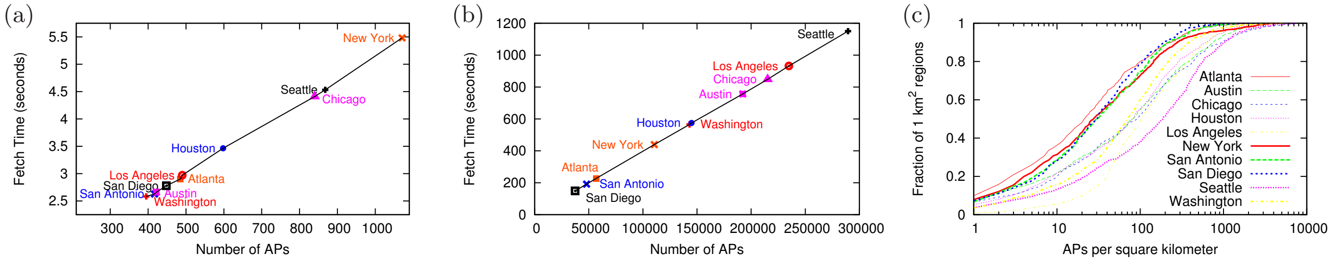

Figure 7— (a) Time to acquire the right to report on all APs listed by JiWire in the top ten cities. (b) Time to acquire the right to report on all APs listed by WiGLE in each of the same ten cities. (c) CDF of the number of APs listed by WiGLE in each 1 km^2 region of a 32 km x 32 km grid centered on each of ten cities.

dicted TCP download throughput of an AP by submitting reports that claim the AP achieves 54 Mbps, the maximum theoretically possible in 802.11g. In this evaluation users only consider each AP's 0%-loss summary, so we assume that each adversarial user submits one report with SNR in the middle of this range. Although he could submit more, they would not change the summary since only one report per user is used. We vary the power of the adversary by varying the number of users that collude to submit these fraudulent reports. A typical AP would also have many honest reports. Therefore, we simulate each AP with 100 reports total: x are the fraudulent reports described above and $100 - x$ are honest reports that are randomly sampled (with replacement) from our ∼10 actual measurements per AP. Note that even if the total number of reports is different, our results still hold on expectation if the ratio of fraudulent to total reports remains the same. The remainder of our simulation setup is identical to §6.1. For comparison to Figure 5(a), we again focus on official APs.

Accuracy. Figure 8 shows Wifi-Reports' prediction accuracy on official APs as we vary the percentage of fraudulent reports. Negligible degradation of accuracy is observed when up to 10% of reports are fraudulent. Even with 30% of fraudulent reports, most predictions are still correct within a factor of 2. However, when 50% of reports are fraudulent, most predictions are gross overestimates. This result is expected since the median function is not robust to 50% or more outliers larger than the actual median.

Discussion. We note that even if an adversary is successful in luring honest clients to a poor AP, those clients will submit reports that correct the summary statistics. Successful fraud attacks that degrade a good AP's reputation (or contract its 0%-loss SNR range) are harder to correct because honest users may be dissuaded from using that AP. However, since cost, venue, and other external factors will influence selections in practice, we believe some honest users will eventually report on these APs and correct their summary statistics.

7. RELATED WORK

Wifi-Reports is related to five areas of previous work: AP selection, electronic cash and secure voting, recommender systems, collaborative filtering, and collaborative sensing.

AP selection. Salem et al. [35] also propose a reputation-based protocol for AP selection. In contrast to Wifi-Reports, their protocol requires changes to the standard 802.11 protocol, it does not protect clients' location privacy, it assumes APs can predict their performance, and it does not address varying wireless channel conditions. In addition, unlike this paper, their work did not evaluate its feasibility on empirical data.

[32, 36, 38] argue for metrics other than signal strength for ranking access points, but only consider metrics that can be instantaneously measured by a single client. We showed in §6 that leveraging historical information out-performs direct measurement [32] because it isn't always possible to test an AP before use. In addition, Wifi-Reports is the only system that enables users to evaluate APs that are not in range, such as when searching for an AP in a hotspot database. Nonetheless, our work is complementary to [36] and [38], which can better estimate the quality of the wireless channel when it is the performance bottleneck.

Electronic cash and secure voting. Wifi-Reports uses blind signatures in a manner similar to well-known electronic cash [20, 21] (e-cash) and secure voting [25] (e-voting) protocols. However, unlike traditional e-cash protocols where a user has multiple tokens that can be spent on any service, a user of our reporting protocol has a single token per service that can only be used for that service. Traditional e-voting protocols typically assume that all users vote (e.g., report) on all candidates (e.g., APs) before tallying the votes, whereas reports are continuously tallied in Wifi-Reports but a precise count is not necessary. As a consequence, our reporting protocol is simpler than traditional e-cash and e-voting protocols, but, like these protocols, it relies on an account authority and distributed talliers (e.g., report databases) to prevent attacks.

Recommendation systems. Having users report on items or services to ascertain their value is a well known idea [14]. Wifi-Reports shares the most similarities with Broadband reports [2], which rates ISPs using user-reported speed tests (e.g., [8]) that measure their back-haul capacities. Unlike Wifi-Reports, Broadband reports takes few measures to prevent fraud. This may be because, unlike the identity of an AP, it is difficult to forge the IP address that identifies the ISP in a speed test. Furthermore, it is easier to limit sybil attacks because a user is identified by an IP address, which is hard to spoof while maintaining a TCP connection. Finally, in contrast to wireless APs, broadband measurements generally do not depend on the location of the user.

Collaborative filtering. Some recommendation systems use collaborative filtering (CF) (e.g., [39, 41]) to identify users that submit many bad reports. However, these techniques require that all reports from the same user are linked and thus do not protect privacy, which is important when location information is at stake. Some proposed CF techniques can limit the exposure of this information by using secure multi-party voting [18, 19]. However, these techniques require all users to be simultaneously online to update summary statistics, and thus are impractical for services that have many users and continuous submission of reports.

Collaborative sensing. A number of recent proposals use mobile devices as collaborative sensor networks (e.g. [30, 13]), but they do not address the unique challenges of AP measurement and reporting. Anonysense [22] is one such platform that ensures that reports are anonymous by using a mix network like Wifi-Reports. However, Anonysense relies on a trusted computing base (TCB) to prevent fraudulent reports and cannot prevent non-software based tampering (e.g., disconnecting a radio antenna). Wifi-Reports does not rely on trusted software or a TCB, but it is more reliant on an account authority to ensure that most reports are honest (though Anonysense is not immune to sybil attacks either). The Wifi-Reports measurement client could also leverage a TCB to mitigate fraud even more.

8. CONCLUSION

In this paper we presented the first measurement study of commercial APs and showed there is substantial diversity in performance. Hence, selecting the best AP is not obvious from observable metrics. We presented Wifi-Reports, a service that improves AP selection by leveraging historical information about APs contributed by users. Wifi-Reports can handle reports submitted at different locations, protects users' location privacy, and is resilient to a small fraction of fraudulent reports.

We have implemented the reporting protocol and a Linux measurement client. We are currently working on clients for smart phone platforms. Although some engineering challenges remain, such as deploying independent report databases, we believe Wifi-Reports can greatly improve users' ability to select good APs.

9. ACKNOWLEDGMENTS

We thank our shepherd Jason Flinn, Vyas Sekar, David Wetherall, and the anonymous reviewers for their comments and suggestions. This work is funded by the National Science Foundation through grant numbers NSF-0721857 and NSF-0722004, and by the Army Research Office through grant number DAAD19-02-1-0389. Damon McCoy was supported in part by gifts from IBM.

10. REFERENCES

[1] Amazon elastic compute cloud. http://aws.amazon.com/ec2/. Accessed on 03/26/2009.
[2] Broadband reports. http://www.dslreports.com/.
[3] Http analyzer. http://www.ieinspector.com/httpanalyzer/.
[4] Illegal sale of phone records. http://epic.org/privacy/iei/.
[5] iPass. http://www.ipass.com.
[6] Jiwire. http://www.jiwire.com.
[7] Skyhook wireless. http://www.skyhookwireless.com/.
[8] Speedtest.net. http://www.speedtest.net/.
[9] Wireless geographic logging engine. http://www.wigle.net/.
[10] You can't send secure email from starbuck's (at least not easily). http://staff.washington.edu/oren/blog/2004/04/you-cant-send-s.html.
[11] Fraud and related activity in connection with access devices. Homeland Security Act (18 U.S.C. §1029), 2002.
[12] Wireless location tracking draws privacy questions. CNET News.com, May 2006. http://news.com.com/Wireless+location+tracking+draws+privacy+questions/2100-1028_3-6072992.html.
[13] T. Abdelzaher, Y. Anokwa, P. Boda, J. Burke, D. Estrin, L. Guibas, A. Kansal, S. Madden, and J. Reich. Mobiscopes for human spaces. *IEEE Pervasive Computing*, 6(2):20–29, 2007.
[14] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005.
[15] L. Balzano and R. Nowak. Blind calibration of sensor networks. In *IPSN*, 2007.
[16] M. Bellare, C. Namprempre, D. Pointcheval, and M. Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003.
[17] A. R. Beresford and F. Stajano. Location privacy in pervasive computing. *IEEE Pervasive Computing*, 2(1):46–55, 2003.
[18] J. Canny. Collaborative filtering with privacy. In *IEEE Security and Privacy*, 2002.
[19] J. Canny. Collaborative filtering with privacy via factor analysis. In *SIGIR*, New York, NY, USA, 2002.
[20] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology*, pages 199–203. Springer-Verlag, 1982.
[21] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO*, pages 319–327, 1990.
[22] C. Cornelius, A. Kapadia, D. Kotz, D. Peebles, M. Shin, and N. Triandopoulos. Anonysense: privacy-aware people-centric sensing. In *MobiSys*, pages 211–224, 2008.
[23] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security*, 2004.
[24] C. Doctorow. Why hotel WiFi sucks. http://www.boingboing.net/2005/10/12/why-hotel-wifi-sucks.html, Oct. 2005.
[25] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *ASIACRYPT*, 1993.
[26] B. Greenstein, D. McCoy, J. Pang, T. Kohno, S. Seshan, and D. Wetherall. Improving wireless privacy with an identifier-free link layer protocol. In *MobiSys*, 2008.
[27] M. Gruteser and B. Hoh. On the anonymity of periodic location samples. In *Security in Pervasive Computing*, 2005.
[28] D. Han, A. Agarwala, D. G. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan. Mark-and-sweep: getting the "inside" scoop on neighborhood networks. In *IMC*, 2008.
[29] G. Judd and P. Steenkiste. Using emulation to understand and improve wireless networks and applications. In *NSDI*, 2005.
[30] A. Krause, E. Horvitz, A. Kansal, and F. Zhao. Toward community sensing. In *IPSN*, 2008.
[31] A. LaMarca, Y. Chawathe, S. Consolvo, J. Hightower, I. Smith, J. Scott, T. Sohn, J. Howard, J. Hughes, F. Potter, J. Tabert, P. Powledge, G. Borriello, and B. Schilit. Place lab: Device positioning using radio beacons in the wild. In *Pervasive*, 2005.
[32] A. J. Nicholson, Y. Chawathe, M. Y. Chen, B. D. Noble, and D. Wetherall. Improved access point selection. In *MobiSys*, 2006.
[33] J. Pang, B. Greenstein, D. Mccoy, S. Seshan, and D. Wetherall. Tryst: The case for confidential service discovery. In *HotNets*, 2007.
[34] R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy. Bandwidth estimation: Metrics, measurement techniques, and tools. *IEEE Network*, 17:27–35, 2003.
[35] N. B. Salem, J.-P. Hubaux, and M. Jakobsson. Reputation-based Wi-Fi deployment protocols and security analysis. In *WMASH*, 2004.
[36] K. Sundaresan and K. Papagiannaki. The need for cross-layer information in access point selection algorithms. In *IMC*, 2006.
[37] United States Census Bureau. Table 1: Annual estimates of the population for incorporated places over 100,000. 2007. http://www.census.gov/popest/cities/tables/SUB-EST2007-01.csv.
[38] S. Vasudevan, K. Papagiannaki, C. Diot, J. Kurose, and D. Towsley. Facilitating access point selection in IEEE 802.11 wireless networks. In *IMC*, 2005.
[39] K. Walsh and E. G. Sirer. Experience with an object reputation system for peer-to-peer filesharing. In *NSDI*, 2006.
[40] H. Yu, P. B. Gibbons, M. Kaminsky, and F. Xiao. SybilLimit: A near-optimal social network defense against sybil attacks. In *IEEE Security and Privacy*, 2008.
[41] H. Yu, C. Shi, M. Kaminsky, P. B. Gibbons, and F. Xiao. DSybil: Optimal sybil-resistance for recommendation systems. In *IEEE Security and Privacy*, 2009.