Tracking Unmodified Smartphones Using Wi-Fi Monitors

A.B.M. Musa
Department of Computer Science
University of Illinois at Chicago
amusa2@uic.edu

Jakob Eriksson
Department of Computer Science
University of Illinois at Chicago
jakob@uic.edu

Abstract

Smartphones with Wi-Fi enabled periodically transmit Wi-Fi messages, even when not associated to a network. In one 12-hour trial on a busy road (average daily traffic count 37,000 according to the state DOT), 7,000 unique devices were detected by a single road-side monitoring station, or about 1 device for every 5 vehicles.

In this paper, we describe a system for passively tracking unmodified smartphones, based on such Wi-Fi detections. This system uses only common, off-the-shelf access point hardware to both collect and deliver detections. Thus, in addition to high detection rates, it potentially offers very low equipment and installation cost.

However, the long range and sparse nature of our opportunistically collected Wi-Fi transmissions presents a significant localization challenge. We propose a trajectory estimation method based on Viterbi's algorithm which takes second-by-second detections of a moving device as input, and produces the most likely spatio-temporal path taken. In addition, we present several methods that prompt passing devices to send additional messages, increasing detection rates an use signal-strength for improved accuracy.

Based on our experimental evaluation from one 9-month deployment and several single-day deployments, passive Wi-Fi tracking detects a large fraction of passing smartphones, and produces high-accuracy trajectory estimates.

1 Introduction

Smartphone sales and use have seen explosive growth in the past several years. In addition to a big display and compelling apps, virtually all smartphones come with a Wi-Fi network interface, allowing them to offload their high data demands to Wi-Fi networks when available. To detect when Wi-Fi networks are available, these phones periodically scan the Wi-Fi band for access points, which typically involves probe message transmissions.

By deploying Wi-Fi monitoring equipment in an area of interest, it is possible to detect these transmissions, providing a coarse-grained location trace for each phone that passes through the area without modifying the phones. Every Wi-Fi transmission contains a unique device identifier (MAC address). Given the popularity of smartphones today, this may be an attractive means of, for example acquiring aggregate movement statistics in an area of interest. In this paper, we study the feasibility of passive Wi-Fi tracking of smartphones through a number of real-world deployments, and present a method of estimating the spatio-temporal trajectories of phones given a set of detections by our monitors.

Tracking vehicles and individuals by the electronic devices they carry is not a new idea. Previous work has performed similar tasks using Bluetooth transmissions [1, 2, 11], tollway transponders, cellphones[3], passport RFID chips [8] and even jogging shoes [24]. In [22], Wi-Fi detections were used to predict bus/train arrival times based on Wi-Fi access points installed in these vehicles. To our knowledge, however, this paper represents the first study of using Wi-Fi transmissions for passive tracking of Wi-Fi clients, and we present the first trajectory estimation method for sparse, RF-based tracking systems in general. §2 contrasts this sparse RF tracking against the more common RF localization methods in the literature.

The primary contributions of this paper are as follows:

- A probabilistic, HMM-based method for estimating smartphone trajectories from Wi-Fi detections.
- A characterization of Wi-Fi scanning and power saving behaviors of a number of popular smartphone models.
- Three methods for increasing the number of detected phones, and per-phone detection count as it passes by a monitor.
- Experimental measurements verifying the widespread use of probe messages by smartphones in the field.
- Evaluation of trajectory estimation accuracy against GPS ground-truth.
- An unexpected finding showing very large numbers (60,000+) of MAC addresses with unlisted OUI prefixes.

Through fixed and temporary deployments, we have found Wi-Fi smartphones to be both numerous and gener-

SenSys'12, November 6–9, 2012, Toronto, ON, Canada.

ous with their Wi-Fi transmissions. In one 12-hour trial using 7 monitors across 2.8 kilometers of arterial road, over 23,000 unique phones were observed. On average, if Wi-Fi is turned on, a monitor detects a passing smartphone 69% of the time. Combined with our probabilistic trajectory estimation method, this resulted in mean error across the entire trajectory of 67 meters compared to GPS ground-truth. In more concrete terms: using seven $70 access points with updated firmware, we were able to produce GPS-equivalent traces with 67 meter mean error for up to 23,000 passing devices over 12 hours.

The remainder of the paper is structured as follows. Background on RF-based tracking and other related work is provided in §2. We describe the overall operation of our system in §3. In §4, our probabilistic trajectory estimation technique is described, followed by three methods for increasing Wi-Fi emissions in §5 and our compact logging method in §6. Deployment options, cost and operational challenges in §7, followed by our evaluation results in §8. §9 concludes the paper.

2 Background

Radio-frequency localization and tracking is a research area with a long history, and a rich area of ongoing research. The majority of work in this area, such as RADAR fingerprinting [6], outdoor Wi-Fi localization [10, 17] and sensor network localization [9, 27, 12, 20, 7] relies on the localized device being an active participant in the process.

In a sense, our passive Wi-Fi localization method is the converse of the now commonplace Wi-Fi localization [10] technique. In our scheme, a non-instrumented device is localized by placing monitors in the area of interest. Active Wi-Fi localization works by modifying the device to listen for un-instrumented stationary access points.

In localization with RF monitors, some prior work has used a dense set of overlapping monitors to triangulate the location of a passive device. In [14], authors tracked devices that were associated with various APs in a campus environment. This work assumes that phones are configured to associate to the network in question. By contrast, our proposed technique tracks unmodified phones and produces trajectory estimates from sparsely deployed monitors.

Most of the traffic monitoring and tracking systems deployed today instrument streets with special purpose sensors such as magnetic loops [16, 15, 18], cameras [5], and toll-tag readers. These methods tend to focus on highways, and are typically very costly to deploy. Some commercial vendors [1, 2] offer vehicle tracking using Bluetooth where traffic flow and travel time is estimated from re-identification of bluetooth devices inside the car. This is the closest work to ours. However, these products produce pairwise time estimates, whereas our proposed method estimates the entire vehicle trajectory.

Nericell [19] combined multiple sensors available in today's smartphone to monitor road conditions and traffic. The possibility of tracking vehicles using the wireless tire-pressure sensors already present in contemporary vehicles is explored in [23]. However, the radio of the tire-pressure sensor has very limited range. This, combined with custom hardware requirements makes for a costly system to deploy. In [3], a similar method is described for tracking users by their cellphone signal. By comparison, we produce full device trajectories, and use off-the-shelf Wi-Fi hardware.

In [21], authors showed that the past location history of a person can be discovered based on probe requests sent by their Wi-Fi devices. This history is also used in our opportunistic AP emulation technique, see §5.

Traffic monitoring is often done by instrumented probe vehicles [26, 25, 13]. Using passive Wi-Fi tracking, we are able to essentially turn non-instrumented vehicles into probes, in the area of interest. A problem with vehicle probes is that the coverage may be sparse both spatially and temporally. The large number of probes produced by passive Wi-Fi tracking may help address this problem.

From a privacy standpoint, passive Wi-Fi tracking is relatively benign, in that it only tracks phones in an area of interest. For further privacy protection, storing only seeded hashes of MAC addresses would be prudent.

3 Passive Wi-Fi Tracking

A passive Wi-Fi tracking system consists of a number of Wi-Fi monitors, and a central tracking server[1]. The focus of this paper is on the collection and processing of Wi-Fi detections. Here, detection processing at the server turns a series of detections of a single phone, into a single consistent and highly likely spatio-temporal trajectory, analogous to a GPS trace for each detected phone.

Device detections are made by capturing Wi-Fi transmissions from the device in question: such transmissions all carry a device-unique address (MAC address), enabling device re-identification across monitors. Unfortunately, passive Wi-Fi detection is an unreliable and highly noisy source of location information by modern localization standards. Phones are unmodified, and thus transmit at their discretion. Wi-Fi transmissions from a cellphone may be received at up to 300 meters, or may be too faint to detect at 20 meter range depending on transmit power, path loss and fading effects. Finally, Wi-Fi tracking deployments are likely to be sparse, potentially leading to extended periods of time without any detections at all. §4 describes the trajectory estimation problem in more detail, and proposes a probabilistic trajectory estimation technique to address it.

Figure 1 gives an operational overview of the passive Wi-Fi tracking system. As a phone travels along a spatial network of roads or paths, it passes by a series of Wi-Fi monitors deployed in the region of interest. Any successfully received transmission is logged as a detection, and reported to the central server on a second by second basis. Upon receipt, the server processes the data from the monitors to reproduce the vehicle's most likely trajectory, which is exported in the form of a second by second location trace.

The quality of the trajectory estimate is typically tied to the number of detections made of a given phone. One way to increase this number is to increase the number of monitors. As we describe in §7, the cost of the necessary equipment

[1] For online tracking, each monitor needs some online means of sending messages to the server, such as Wi-Fi mesh networking, a nearby Wi-Fi Internet access point, etc.

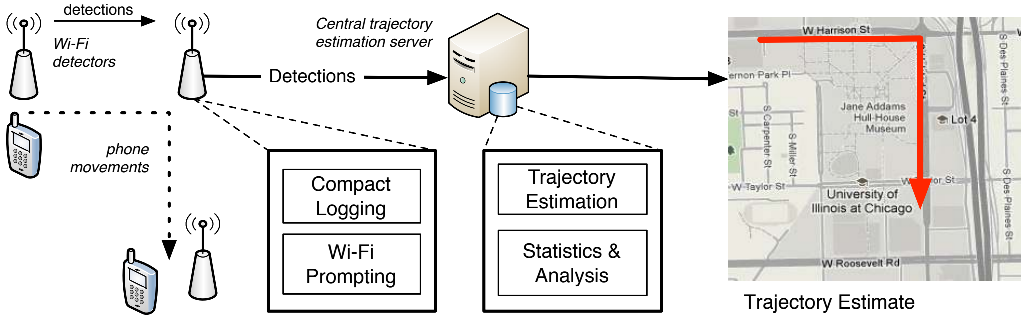

Figure 1. Operational overview of our passive Wi-Fi tracking system. Monitors detect passing phones, forward compact detection records to central estimation server, which produces a GPS-like, second-by-second trajectory estimate.

and connectivity for a dense deployment is low, though external factors may still restrict the number of monitors. Another, complementary option is to increase the number of detections made by a single monitor. As detections are only possible when a phone transmits a Wi-Fi packet, a mechanism is needed by which passing phones may be encouraged to increase their number of transmissions. In §5 we describe three different techniques that prompt passing phones to send additional packets.

4 Estimating Smartphone Trajectories

In our proposed system, Wi-Fi monitors detect transmissions from passing phones. We make the assumption that phones travel along a spatial network of roads or paths: i.e. tracking free movement in an open area is not supported. Based on a series of detections of a phone, our goal is to produce an accurate second-by-second estimate of the phone's location. This task is made difficult by several factors, as listed below.

Spatial Sparsity of Detections Densely instrumenting the entire area of interest with Wi-Fi monitors may be neither feasible nor desirable. This makes standard RF localization methods [6, 10] poorly suited for this problem, as they typically rely one some variant of triangulation. In the case of Wi-Fi detections, the number of monitors simultaneously detecting a phone typically varies between 0 and 1, and may occasionally reach 2.

Temporal Sparsity of Detections Detections rely on phones actively transmitting Wi-Fi packets, which may not happen regularly or reliably. When a phone does transmit, it may transmit only a few packets, over a very short duration. Often times, a phone may pass by a monitor without being detected at all.

Overlap The measured Wi-Fi packet reception range in our experiments is 250-300 meters. Thus, if a pair of monitors are spaced more closely than 600 meters, a transmission may be detected by two monitors simultaneously.

Unpredictable path loss Differences in phone placement, immediate surroundings and transmission power contribute to significant differences in received power between devices at the same range from a monitor.

Fading Mobility and a highly dynamic environment lead to significant temporal variations in received signal strength and highly stochastic behavior. In cases with overlap, this can lead to situations where, for two monitors A and B, a phone is closer to monitor A than monitor B, but is detected by monitor B one second, and by monitor A the next.

From the list above, it is clear that no passive Wi-Fi tracking system can guarantee accurate tracking performance. In particular, the significant possibility of missed detections may introduce positional ambiguity that cannot be completely resolved, even if the car travels only on instrumented streets. Below, we first discuss a simple straw-man solution, and then go on to describe our proposed trajectory estimation algorithm.

4.1 A Straw-Man Algorithm

For the sake of argument, consider the following naïve solution.

> *Whenever a detection is made, take the detecting monitor's location as the phone's location. Interpolate phone locations between detections.*

This is a poor solution for several reasons. First, the long range and temporal sparsity of detections means the phone may be as much as 300 meters away from the monitor location when a detection is made. Received signal strength (RSS) is a distance indicator, but no useful one-to-one mapping between distance and RSS can be found: while a high RSS is very unlikely at long distances, a low RSS may arrive from a wide range of distances.

Second, overlap and variations in signal strength means a phone that is stationary between two monitors may well be detected intermittently by either or both. This straw-man algorithm would have the phone's location jumping rapidly back and forth between the two monitor locations. Imposing restrictions on acceleration or velocity would help, but would not solve the fundamental problem.

Third, straight-line interpolation between distant locations produces very low-quality trajectories. Imposing a spatial network, such as a road map, can significantly improve this quality, but introduces a route ambiguity which must be resolved. Using the shortest path between monitor locations is not feasible due to the high location uncertainty.

As this example suggests, this problem does not lend itself well to standard deterministic algorithms, due to the highly stochastic nature of our detections.

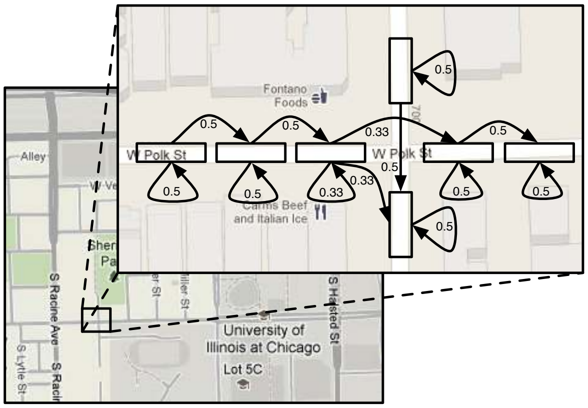

Figure 2. Hidden Markov model of two intersecting one-way roads. Road segments are subdivided to increase granularity. (Figure not to scale.)

4.2 Basic Trajectory Estimation Algorithm

Below, we discuss a probabilistic trajectory estimation method. In essence, rather than directly assign the phone to a single location at each instance in time, we model the distribution of possible locations over time, and extract the maximum probability trajectory from this model.

To reduce the possible set of locations, restrict movements and so help produce an accurate trajectory, we assume that phones travel along a known spatial network such as a road or trail map. A model of this spatial network, consisting of spatially located vertices connected by directional edges, is taken as input.[2]

Given such a spatial network, the trajectory estimation problem bears a measure of similarity to map-matching, in that a network path is produced from a series of potentially low-accuracy location estimates. Indeed, our trajectory estimation method is inspired by Viterbi map-matching [26]. We thus formulate the trajectory estimation problem using a hidden Markov model of states, transition probabilities and emission probabilities, and use Viterbi's algorithm to find the maximum probability path traversed, represented by a sequence of hidden states visited in the Markov model.

When used in conjunction with the Viterbi algorithm, a hidden Markov model expects periodic sensor input, and produces one state transition $i \rightarrow j$ per period. Transitions are governed by transition probabilities $p(i \rightarrow j)$, and emission probabilities $p(obs|i)$, where obs is the current observation. Self-transitions (transitions s.t. $i = j$) are allowed. Below we describe the HMM formulation in more detail, in terms of hidden states, transition probabilities and emission probabilities.

4.3 Hidden States: Cut-Up Road Segments

A simple road map typically consists of vertices, representing intersections, edges representing road segments between intersections, and (mostly for vehicles) turn-restrictions determining allowable transitions between edges. In prior work on Viterbi map-matching [26] each road is represented as a state in the Hidden Markov Model. Over time, a tracked phone travels from one road segment to another road segment via their adjacent intersections. This has been shown to work well for both GPS and WiFi localization (with an active receiver in the vehicle). Here, the location of the device is relatively well known, and the objective is simply to determine which road segment is being traversed.

In the case of passive Wi-Fi tracking however, our objective is to determine the second-by-second location. The output from the Viterbi algorithm is a maximum-probability sequence of states. Thus street-segments, which may be hundreds of meters in length or more, are not of a sufficient granularity for trajectory estimation. Moreover, we would like our model to capture the stochastic relationship between signal strength and distance. To do so, we also require states with a significantly smaller spatial extent.

Figure 2 illustrates our simple solution. The zoomed in section of the map contains two one-way roads going south and east. Here, the two east-bound street segments on either side of the intersection are divided into multiple sub-states. Each sub-state represents a rectangular area in which a phone may be located, and a road segment may only be traversed by passing through each of its sub-states in sequence. Turn restrictions are maintained for the final substate of any segment.

This accomplishes two things. First, it increases the granularity of the HMM such that an approximate location along the street can be produced. Second, it enforces a speed-limit of sorts. In the HMM, transitions happen in lock-step with incoming sensor readings, at regular intervals. Thus, given one sensor reading per second (which may sometimes show "no detection"), only one sub-state per second can be traversed, limiting travel speed to one sub-state length per second. Enforcing a speed-limit helps significantly in suppressing the output of spurious routes.

4.4 Transition Probabilities

Transition probabilities primarily model the behavior at intersections: continuing straight, or turning. In this work, we do not make any assumptions about driver behavior, and use uniform transition probabilities between adjacent segments. Thus, for each segment i with adjacent segments $n \in N$, $p(i \rightarrow i) = p(i \rightarrow n) = \frac{1}{|N|+1}$. Any turn restrictions, including allowed or disallowed u-turns, are accounted for in the adjacent segments.

4.5 Emission Probabilities

Emission probabilities describe the probability $p(obs|s)$ of making observation obs if the current state is s. In our case, the observation is the set of observations of the phone in question, by all monitors, at the present time. As mentioned above, the state s is a part of a road segment. Thus, we are asking "if the phone was here, what is the probability that we would have made the detection we just made?"

Correctly modeling the emission probability is central to our trajectory estimator. The emission probability model defines the probability distribution of the phone's location across the entire spatial network, for each second.

[2] If no spatial network is available, this is a spatial network consisting of all the monitors, and a bi-directional edge between each pair of monitors.

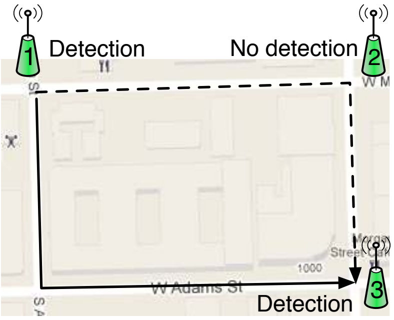

Figure 3. Motivating example for our emission probability design.

Several factors influence the emission probability of an observation. Distance, of course, plays a big role. Intuitively, the probability of making a detection decreases with distance, reaching zero at the maximum reception range. However, variability in channel quality (fading) is a significant factor as well, influenced primarily by placement and (potentially moving) obstacles. This implies that a transmission may not be detected, even when sent at close range.

Also, a detection can only be made if a transmission actually took place: phones typically transmit WiFi packets at rather long intervals, and for short durations of time. We discuss this, and how the frequency and duration can be increased, in more detail in §5. Unfortunately, we cannot predict when a transmission will happen. Thus, a phone may pass by a monitor without transmitting anything, and remain undetected.

An observation *obs* consists of a set of events $e_m \in obs$, one from each monitor m, such that

$$p(obs|s) = p_{tx} \prod_{m \in obs} p(e_m|s,tx), \quad (1)$$

where tx indicates that a transmission happened, and p_{tx} is the probability of a transmission happening. As we do not have a good model for predicting when a phone will transmit next, we estimate a constant $p_{tx} = \frac{1}{300}$ from measurements under controlled conditions. Note that the algorithm is not very sensitive to this value.

Each event e_m can take the values $detection_m$ or $nondetection_m$. Here, $nondetection_m$ simply indicates that the phone was not detected at this monitor. While this may seem like an insignificant event, it does in fact carry some valuable information. Of course, non-detections need not be physically reported to the server: the absence of a detection report indicates a non-detection. Each $detection_m$ is accompanied by a signal strength annotation. Naturally, a $nondetection_m$ has no equivalent.

For all states s, it is true that $p(nondetection_m|s) = 1 - p(detection_m|s)$. For states outside of the maximum detection range, $p(detection_m|s) = 0$.

Figure 3 illustrates a motivating example for this design. The phone enters the area at monitor 1, where it is detected. It then travels to monitor 3 along an unknown path, and is not detected until it reaches monitor 3. Here, the fact that no detection is made between monitors 1 and 3 carries some information. All other things being equal, it is more likely that the phone traveled the route with no intermediate detector.

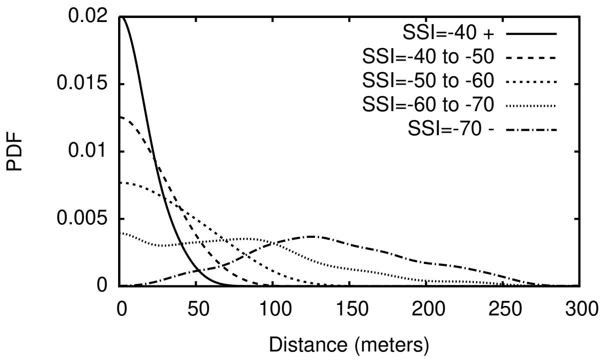

Figure 4. Distance probability density function for several RSS ranges.

Finally, to determine $p(detection_m|s,tx)$ we rely on experimental measurements of the RF channel. Figure 4 illustrates, for a received packet at a given RSS, the probability density of the transmitter being at a certain distance, $p(dist)$. To create this plot, we repeatedly drove and walked past several monitors at approximately constant velocity, carrying several smartphones configured to transmit packets periodically. We then compute, over each RSS bin, a Gaussian kernel density estimate over the distance of all received packets, or

$$p(dist|RSS) = \sum_{dist \in received\_packets} N(dist, \sigma^2), \quad (2)$$

where σ is the kernel bandwidth, which we picked visually to be 15 meters, to produce a smooth curve. Interestingly, the -70dBm curve has close to zero probability density near the monitor, whereas all others have a high density at that point.

Intuitively, a low signal strength packet may have come from a wide range of distances, whereas a high signal strength detection almost certainly originated near the monitor. This allows us to compute $p(detection_m|s,tx)$ for a packet of a given signal strength over the area covered by s, or,

$$p(detection_m|s,tx) = \int_x \int_y p(dist(x,y,m)|RSS)dxdy, \quad (3)$$

where $dist(x,y,m)$ is the Euclidean distance between the coordinate (x,y) and the monitor. Due to this measurements-driven approach, our passive Wi-Fi tracking system is easily adapted to different channel conditions: calibration can be done by simply traversing the instrumented area with a selection of phones and a GPS receiver.

4.6 Viterbi Processing

Every second, each monitor reports the MAC addresses it observed during that second. Given aggregate detections or non-detections of a given phone from all monitors, we compute the emission probability of the combined multi-monitor observation as described above. We then use Viterbi's algorithm to turn a series of observations and corresponding emission probabilities into a location trace.

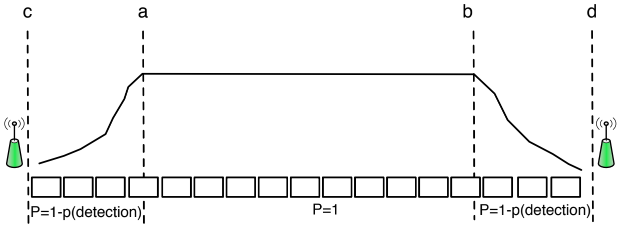

Figure 5. When multiple maximum-probability estimates exist, choose the trajectory that minimizes the expected error over all estimates.

Viterbi's algorithm produces the maximum probability series of states, given a finite series of observations. However, due to our handling of non-observations, the observation series does not end. Rather, as a vehicle leaves the reception range of the monitors, the lack of packet receptions are regarded as non-observations, which are then fed into the algorithm.

Since our goal is to reproduce the path traveled, rather than estimate a probability distribution we must at some point choose a single path. In general, the longer the wait, the more information is assimilated by the algorithm, resulting in more accurate paths. However, the information value of repeated non-observations decreases quickly. Thus, our algorithm by default makes a "hard" choice, which implies that a location trace is produced, after an inactivity period T.

If real-time tracking is desired, an alternative solution is to produce the trajectory in a streaming fashion. That is, produce output as soon as part of a trajectory can be estimated with high confidence. Clearly, non-detections do not admit high-confidence estimates: we essentially have no idea where the phone went, until we hear from it again. Thus, additional trace portions are not produced during periods of non-detection. In our implementation, output is delayed further: given observations at monitors m_1, m_2, m_3 in that order, the first trace portion, between m_1 and m_2, is produced only when the first observation at m_3 is made.

4.7 Error-Minimizing Estimates for Periods of Non-Detection

The output of the Viterbi algorithm is the maximum probability state sequence for the given time series of observations. During periods of detection, the variations in emission probability induced by mobility typically keeps ambiguity relatively low, and a unique maximum probability path often exists.

However, during periods of non-detection, such as when a phone travels between two well-separated monitors, or simply does not transmit any Wi-Fi packets for a while, ambiguity in the maximum probability path rises quickly. Figure 5 illustrates an example situation. Here, during the period of non-detection, any number of maximum-probability paths exist.

In Figure 5, consider the case where no detections are made between points (a) and (b). The probability of non-detection in all states between (a) and (b) is one. Hence, as long as the transition probabilities are equal for all states between (a) and (b), any allowable sequence of states between these two points is a maximum probability path. Without further guidance, an arbitrary path will be selected by the Viterbi algorithm.

Note that in this example, and indeed in most scenarios of this type, all maximum-probability paths follow a single spatial path between (a) and (b). The difference lies only in the duration of time spent in each state. Thus, rather than let the Viterbi algorithm choose an arbitrary distribution of time, we want to distribute the time spent in each state in such a way that the expected error is minimized over the distribution of all possible trajectories. For the (a)–(b) case above, it can be shown that straight interpolation achieves this goal. Thus, given a total duration T, during which a spatial path consisting of $|S|$ equal-sized segments $s \in S$, the optimal time spent in each segment during a period of non-detection is $t(s) = \frac{T}{|S|}$.

In general, and this is illustrated by the case (c)–(d) in Figure 5, we believe (but have not proven) that the optimal distribution is

$$t(s) = \frac{T p(obs|s)}{\sum_{u \in S} p(obs|u)},$$

where *obs* is the observation that shows non-detection in all monitors. In §8.4, we compare this error-minimizing method against standard Viterbi processing, and find that it achieves a significant reduction in mean error as well as root mean-squared error.

5 Prompting Additional Transmissions

The performance of passive Wi-Fi tracking can be defined in terms of coverage (the number or percentage of phones tracked), or in terms of accuracy (e.g. mean location error over time). In this section, we describe three mechanisms that address these dual objectives: maximizing the number of devices detected, and maximizing the number of packet receptions from each detected device. As discussed in §4, increasing the number of packets received from each device is important for determining accurate trajectories of moving devices.

To satisfy these goals we employ three techniques in the Wi-Fi monitor in addition to passive monitoring. First, we advertise two popular access point SSID's (attwifi, tmobile)[3]. This increases both the number of phones detected, and the number of packets received from each phone.

Second, we emulate APs with SSID for which a directed probe-request is made by a phone. This does not increase the number of phones detected, but does increase the number of packets received from each phone.

Finally, we periodically send RTS packets to detected phones, causing them to respond with a CTS. This increases the number of packets received from each phone. In §6, we discuss how detections are stored and delivered to the central tracking server.

5.1 Popular SSID AP Emulation

Cellular data network of mobile operators are struggling to keep up with the increasing data demands of smartphones

[3] While anybody can advertise any SSID, advertising a "pretend" attwifi access point where a real AP exists could create confusion. We disable such advertisements whenever such confusion may occur.

and tablets. One mechanism mobile operators use to alleviate this problem is Wi-Fi data offloading. For example, AT&T provides free open Wi-Fi access at Starbucks and other public places. These not only provide convenient Internet access to people but also allow offload data demands from the cellular network.

Mobile operators also try to make sure smartphones automatically connect to their Wi-Fi hotspot. For example, AT&T's free Wi-Fi hotspots use attwifi as SSID. Once an iPhone has been associated with an attwifi AP, the iPhone automatically associates with any future attwifi APs it finds nearby.

When a phone scans for nearby access points, two things may cause our monitor to not detect the phone. First, a phone may enable its Wi-Fi receiver, yet not transmit any probe messages, or second, the phone may transmit a message which is not received correctly by the AP. In both cases, we can use the auto-associating features of typical smartphones to recover. By transmitting beacons advertising popular SSID's such as "attwifi" and "tmobile", any phone with an active receiver tuned to the correct channel becomes aware of these two "virtual" APs. Many times, the phone will then send an association request, increasing chances of a phone being detected by a monitor.

Perhaps more importantly, we also want to increase the number of packets received by each passing phone. For this, we use `hostapd`, to emulate the full operation of an open Wi-Fi access point (though in our case without providing end-to-end Internet connectivity). Thanks to `hostapd`, the energy saving functions of IEEE 802.11 are handled automatically, making sure that packets are only sent to the phone when its receiver is active. The power-saving functionality in the phone typically results in the transmission of null-frames to the AP, which we can use for tracking purposes. This process continues as long as a station is associated with our AP.

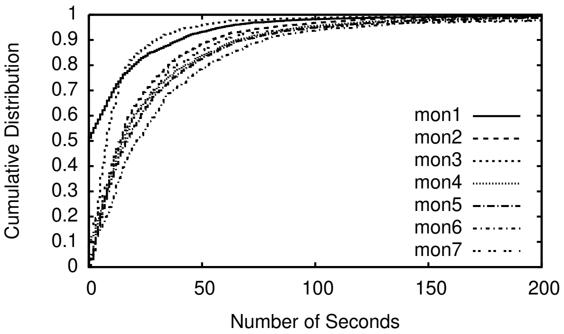

Figure 6. Distribution of time duration of null packet reception for associated devices in fixed APs.

We receive additional packets from most associated stations using this technique. Figure 6 shows the CDF of the number of seconds (not necessarily consecutive) during which null packets are received from associated devices, across 7 different monitors and several thousand phones. For most monitors, packets are received for a median of 10-15 additional seconds, typically well spaced through the time of association. Some associations result in significantly higher numbers, likely due to differences in mobility: a phone that moves slower stays associated for longer, and has more opportunities to transmit. One point should be noted here that we record only one packet every second for the purpose of compact logging and hence the actual number of packets received can be higher.

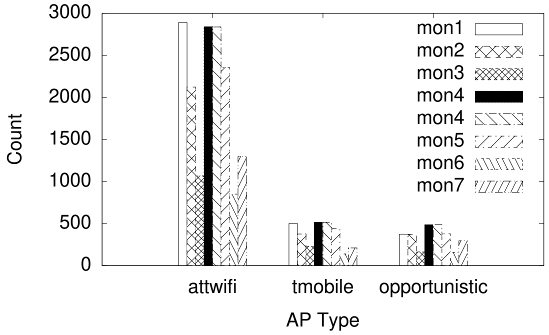

Figure 7. Unique devices associated with different types of APs during 12 hour deployment.

Figure 7 shows the number of unique devices associated with our emulated APs (attwifi, tmobile), for a 12 hour deployment using 7 monitors. Here, attwifi dominates due to its significant iPhone presence, but tmobile produces a measurable number of associations as well. The third set of bars illustrates the number of associations using "opportunistic" AP emulation, discussed below.

5.2 Opportunistic AP Emulation

A station that wants to get associated with an AP scans for APs nearby either passively by observing AP beacons or actively by sending probe requests. While scanning actively for APs, most phones will transmit an anonymous, broadcast probe request. While these provide a detection, we have not found a way to leverage such broadcasts into multiple transmissions. However, phones frequently also transmit directed probe-requests for a particular SSID that it has been associated to in the past. Such probe requests reveal the association desires of the phone in question, which we can take advantage of.

By emulating the requested SSID, we can encourage the device to attempt to associate with us. However, the association process initiates at the station, which will only attempt association if it recognizes the emulated AP as one that it has connected to in the past. One major factor in this recognition process is the security protocol used, which is not specified in the probe request.

For open authentication, there is no security protocol, and the station gets associated with the AP as in the popular SSID case above. But for secured Wi-Fi networks, the smartphone will not try to associate if the security protocol does not match with the devices' remembered security protocol. Hence we emulate multiple APs with same SSID but different security protocols for each observed SSID so that one of the emulated APs matches with the station's remembered APs and the device associates with the emulated AP.

The overall association process in shown in Figure 8. In the case of open association, the association process com-

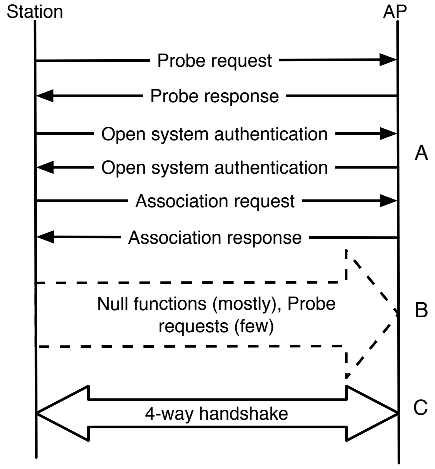
Station
AP
Probe request
Probe response
Open system authentication
Open system authentication
A
Association request
Association response
Null functions (mostly), Probe requests (few)
B
4-way handshake
C

Figure 8. Association process of a station with an AP. In the case of open association, the process completes at A. However, for WPA/WPA2 the subsequent secure association 4-way handshake (C) is required. If this is not initiated by the AP, the station repeatedly sends null packets and probe-request packets.

Platform	Behavior
IOS 4.3.3	Sends null-function packets continuously and probe request packets intermittently for around 10 minutes.
IOS 5.0	Send a single null packet and disassociates after 5 seconds. Then repeats association attempt.
Android 2.3.4	Sends 10 null packets every second for 10 seconds, then disassociates. Then repeats association attempt.

Table 1. Smartphone secure Wi-Fi association behavior after open system association.

pletes at time A (labels on right side of figure). However, for WPA/WPA2 secure association, a 4-way handshake at time C is expected. As we do not have the pre-shared key for PSK or valid certificate for 802.1x, our emulated AP cannot perform this 4-way handshake, and does not attempt it. However, as shown in the Figure 8, if the AP does not start 4-way handshake, the station stays at (B), which sends null packets continuously to see if the AP is still there. This satisfies our goal of getting additional packets from the phone.

Table 1 summarizes the different behaviors of smartphones at time B in Figure 8. Essentially, once a phone associates, it will either send null and probe packets at regular intervals, or repeat the association after a timeout. Either case provides useful packet transmissions for tracking.

Our implementation supports the PSK and 802.1x authentication protocols and TKIP, CCMP ciphers, some of the more popular options for secure WPA/WPA2 authentication. Figure 9 shows the CDF of packet reception duration in opportunistic APs. A small majority of associated phones sent packets for increased duration due to this mechanism, and 30% of phones sent packets for more than 50 seconds. Again, these are spaced throughout the duration of the association, making them valuable for tracking purposes.

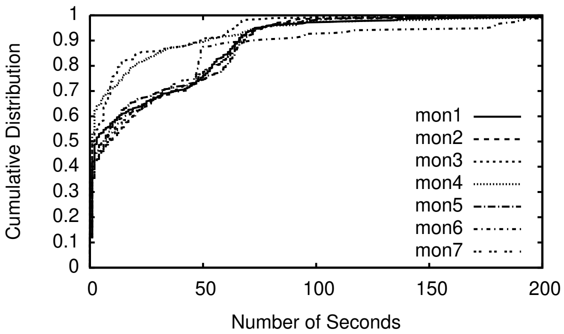

Figure 9. Distribution of time duration of packet reception for associated devices in opportunistic APs.

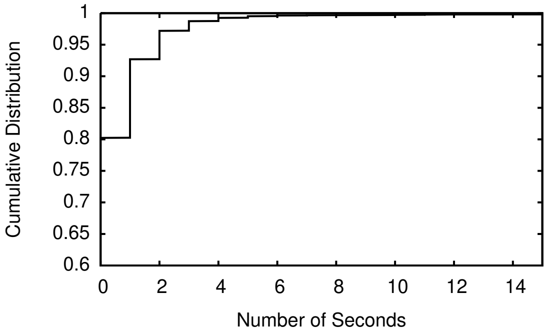

Figure 10. CDF of number of seconds during which CTS packets are received per MAC, for non-associated MACs only.

5.3 RTS Injection

Our third and final method addresses phones that do not attempt to associate with our emulated access points. According to the IEEE 802.11 standard, when a station receives an RTS frame, it should respond with a CTS if the channel is clear, its receiver is active and tuned to the correct channel. It should be noted that RTS packet contains a receiver address but no transmitter address. Hence it is not possible to determine the transmitter of a CTS packet directly from the packet.

Since the transmitter address is required for tracking, we devise a simple solution to this problem. The device transmitting the CTS uses the transmitter address in the corresponding RTS as the receiver address in the CTS packet. Hence, by using separate transmitter addresses for each intended receiver, we can distinguish between the transmitters of CTS packets. Another possible solution would be to rely on the timing of RTS and CTS frames to determine the identity of the sender. However, such precise timing information was not available on our hardware.

Figure 10 shows the results of RTS injection, for non-associated MACs only. We find that response to RTS increased the tracking duration by at least one second for 20% of observed, non-associated MACs. This is decidedly fewer phones and fewer responses than we had hoped for. The low numbers may be explained by two reasons. First, the fact

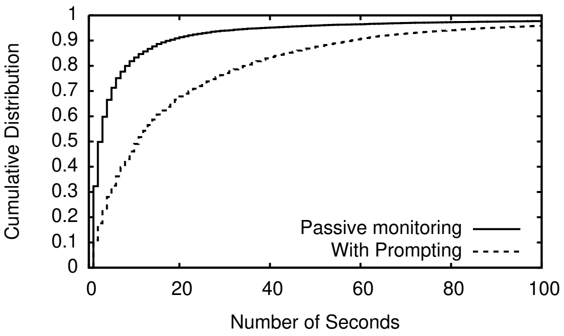

Figure 12. Impact of our three "prompting" techniques on the increment of packet reception duration per phone.

that a phone that is not associated does not keep the receiver active, or tuned to the same channel, for an extended period of time. Second, we keep only one packet every second and response to RTS can be thrown away because of this.

5.3.1 Null Frames

Recently, we were made aware of one more potential avenue of soliciting additional packets from a passing phone. Apparently, if a phone receives a null frame, it will respond with an ACK. We have not investigated this further thus far. However, it appears likely that this technique, if successfull, would produce results similar to the RTS technique described above.

5.4 Combined System

Figure 11 shows the high-level architecture of the Wi-Fi monitor subsystems discussed above. On top, each "interface" represents a virtual interface only. Each monitoring device has a single Wi-Fi radio. The left part of the figure illustrates the operation of our compact logger, which is discussed in §6. Data is delivered to the central server via one of several uplink options, discussed in §7.

To get an estimate of the combined impact of our three methods, we compare the total number of packets received per device, for a 7-monitor passive deployment, vs. a 7-monitor deployment with all three techniques activated. Figure 12 illustrates the result of this experiment. We find our techniques significantly increase the number of packets received per phone, increasing the median packet count by 5×, and the 90th percentile by 3×. Given the RTS results above, it is likely that this improvement is primarily due to the first two techniques.

6 Monitor Resource Conservation

Many of the areas of most interest for Wi-Fi tracking studies may well have a high level of background Wi-Fi activity. Capturing and retaining all of these packets, independent of whether they come from moving phones, would require significant storage and/or uplink capacity. For a practical and low-cost system, it is paramount that we reduce these requirements dramatically.

Our system supports both offline store-and-retrieve and real-time data upload modes of operation. In the case of real-time data upload, using little bandwidth is important as the uplink may be either costly (cellular), low capacity (long-range Wi-Fi mesh) or donated (nearby Wi-Fi AP). The lower the bandwidth requirements, the easier and less costly the deployment. To reduce data storage and transmission demand, we only retain second-by-second aggregates, and attempt to retain and transmit data from moving devices only.

6.1 Filtering out Stationary Devices

Stationary Wi-Fi devices are commonplace in virtually any populated area. However, we are only interested in transmissions from moving devices. To filter out packets from stationary devices, we use the following simple heuristics.

Total Observation Duration If a device d has been observed for a time greater than a blacklist threshold θ, add d to the monitoring blacklist.

However, we should not keep a device blacklisted forever as it may become mobile in future. Our second heuristic removes devices from the blacklist.

Time Since Last Observation If device d has not been observed for a time longer than the blacklist expiry threshold ε, then remove the device from blacklist and reset the total observation duration for this device.

6.2 Data Aggregation

While our algorithm operates on a second-by-second basis, it is common for Wi-Fi stations to transmit a large number of packets per second. Due to this, reporting the actual packets received, or even just the headers of these packets, is not practical due to bandwidth constraints.

Our system retains only a minimal set of information about the observations made in each second. For each observed device, we keep the MAC address and maximum received signal strength. Intuitively, and according to Figure 4, the maximum signal strength packet contains the most location information. This aggregate data is then forwarded second-by-second to the central server, or stored locally for subsequent retrieval.

For a 12-hour, 7 node deployment on a busy arterial street, during which a total of 23,000 phones were detected, the mean output rate of the monitors was 120 bytes/s (std.dev. 30 bytes), or a total of 5 MB each over the full 12-hour period.

7 Deployment Considerations

Our passive Wi-Fi tracking system consists of one or more Wi-Fi monitors placed in the area of interest, and a central server where Wi-Fi observations are aggregated and analyzed.

Wi-Fi monitors are created from standard Wi-Fi access point hardware with a custom firmware update. This provides several significant benefits. First, outdoor AP hardware is readily available, reliable and very low cost. For example, our hardware of choice, the Ubiquity PicoStation 2, retails for around $70.

The two main challenges for a large Wi-Fi monitoring deployment are power supply and connectivity. While monitors consume very little power (approx. 7W for our hardware), power may not be readily available in the area of interest, especially outdoors. In our experiments, we used two different solutions to this problem. For short-term deployments of less than 24 hours, we used a battery pack.

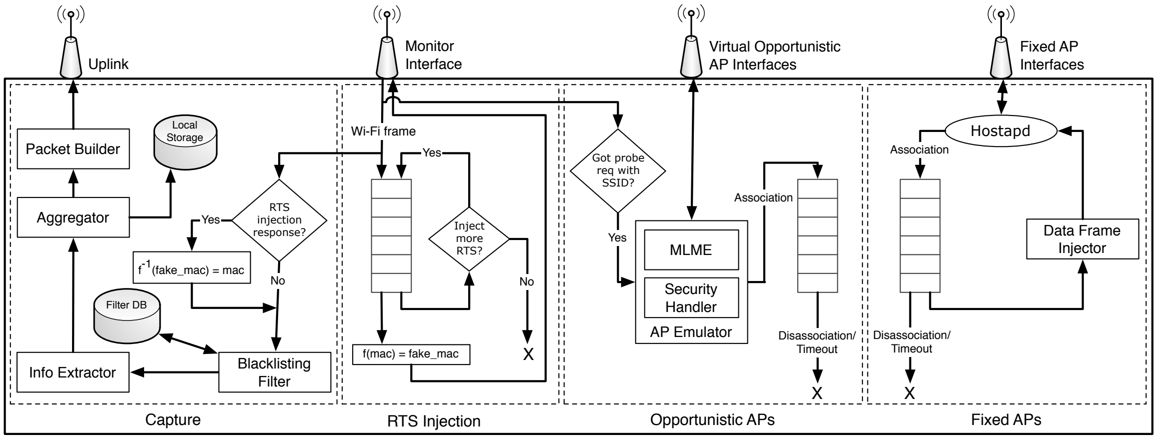

Figure 11. Functionality and processing at a Wi-Fi Monitor. Capture subsystem performs aggregation and filtration of the captured data. Fixed AP subsystem emulates few popular Wi-Fi hotspots and inject data packets to associated devices. Opportunistic APs emulate many different APs from probe-request SSIDs and perform MLME with different security protocols. Finally, RTS injection subsystem inject RTS packets for detected devices that are not associated through either fixed or opportunistic APs.

For longer-term deployments on campus and next to streets, monitors were deployed indoors near a window facing the area of interest. A large scale outdoor deployment would benefit tremendously from cooperation with the local municipal agency, which typically has power available at every intersection.

Monitors can be placed at any distance from each other depending on application requirements. However, the denser the placement, the more accurate the trajectory estimate. When Wi-Fi or Ethernet connectivity is available, each monitor connects directly to this uplink. We used this deployment mode successfully at the edge of campus, as well as in nearby restaurants and residences. We found that using the radio for both monitoring and uplink works well, and set-up was easy.

When no uplink infrastructure exists, three remaining options exist. Monitors can be equipped with a cellular modem for Internet access. Unfortunately, this incurs a significant monthly cost, which may be prohibitive for long-term or large-scale deployments. A benefit of a dense monitor deployment is that it enables monitors to form a back-haul mesh network. Given the minimal bandwidth requirements of the monitors, a single uplink somewhere in the mesh can potentially serve a large network. We were able to successfully establish mesh links across a city block distance. This suggest that a dense deployment with one monitor per intersection may be the ideal configuration.

Finally, if no Internet connectivity is available at all, we also support local storage mode. In this mode, monitors store their detections for retrieval by a “data mule”, which wirelessly downloads the log as it drives by. A data muling backhaul may be attractive where real-time tracking of phones is not necessary.

8 Results

We used three different deployments, shown in Figure 13, for the results described in this section. First, we have permanently deployed 5 nodes along the streets near our campus as shown in green. These nodes are deployed indoors, near windows overlooking the street. The permanent deployment spans 9 months, during which time 400000 unique MACs were observed.

Second, a 12-hour temporary rectangular deployment with 6 nodes on moderately busy roads, marked in red. These nodes were mounted on existing poles along the street, and performed passive monitoring only. In this trial, a total of 20000 unique MACs were observed across all monitors.

Finally, a 12-hour temporary linear 7-node deployment spanning 2.8 kilometers along a single highly trafficked road, marked in blue. Nodes were mounded on street poles. In this trial, we also enabled our optimizations to encourage phones to send additional packets. Here, a total of 23000 unique MACs were observed.

For performance evaluation purposes, we repeatedly traveled the streets covered, both walking and driving, with several different phones in our hands, pockets, or on the seat next to us in the car. For each walk or drive, we also collected a GPS trace to serve as the ground truth for location estimation.

8.1 Mystery Addresses with Unlisted OUI

The first three octets of a MAC address is the Organizationally Unique Identifier, which designates the vendor of the device. The OUI list at [4] is the authoritative list of OUIs, updated daily.

Excluded from our reported datasets, and all our other measurements, is a large fraction of MAC addresses with unlisted OUIs. Because we have not been able to verify the origin of these addresses (i.e. they may be spoofed or otherwise non-indicative of a unique device), we conservatively choose to exclude them. We have verified that these are *not* link-local addresses, nor multicast addresses: the vast majority of offending addresses have the two lower order bits of the first octet set to zero. Certain statistical properties are

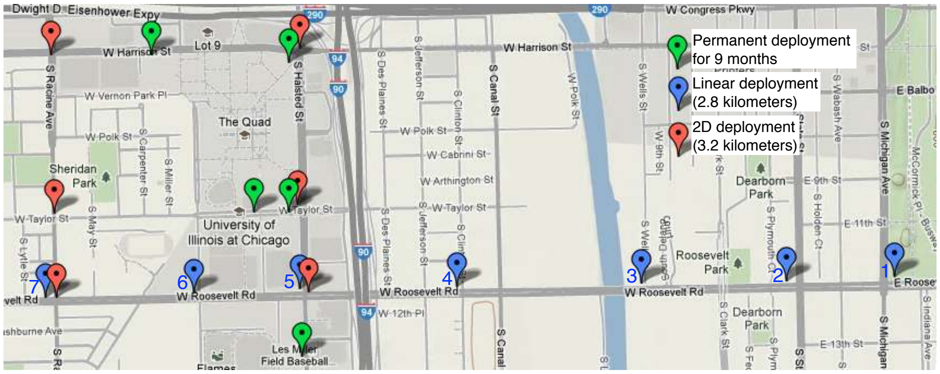
Permanent deployment for 9 months
Linear deployment (2.8 kilometers)
2D deployment (3.2 kilometers)
1
2
3
4
5
6
7
University of Illinois at Chicago
W Roosevelt Rd
S Halsted St
W Taylor St
W Harrison St
Dearborn Park
Roosevelt Park
Sheridan Park
The Quad
S Michigan Ave

Figure 13. Three deployments of monitor nodes used for evaluation.

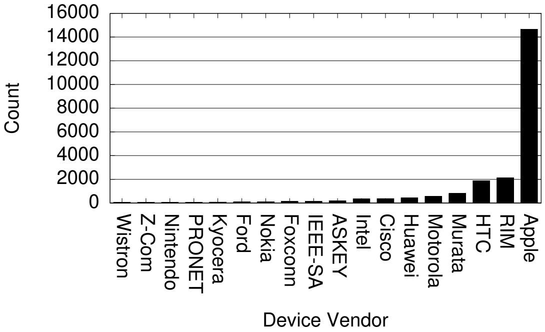

Figure 14. Distribution of devices by vendor in 7 monitors during 8AM-8PM

also the same between regular and unlisted OUI MAC addresses, see §8.2.

Across all of our deployments, we have observed well over 60,000 unique MAC addresses with unlisted OUIs. The movement of such a large number of unlisted (and hence irregular) MAC addresses on public streets is an unexplained mystery to us, and we would appreciate any input from the community. This claim is further supported by the fact that virtually no "mystery addresses" occur with the link local or multicast bits set to one.

8.2 Statistical Properties

We begin with a statistical study of the output of our monitors. For these results, we use data from our 9-month deployment unless otherwise noted.

By looking up the OUI in the official registry [4], we can compute a distribution of vendors, as shown in Figure 14. This shows Apple in a dominating position, followed by RIM and HTC. Note that these results show what was *detected* only, not the distribution of phones on the street. It is conceivable, indeed likely, that the default configuration of many Android phones precludes these from being detected. Alternatively, Android phones are the source of the mystery MACs mentioned above. Our only evidence in this direction is the fact that the number of such mystery addresses is roughly equal to Apple MACs across measurements, and that Android phones are not significantly represented elsewhere—hardly damning evidence.

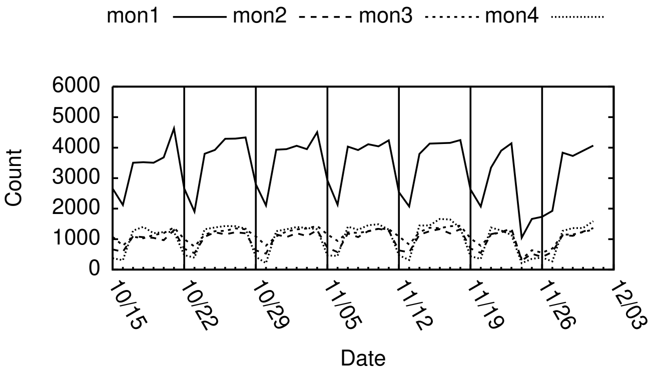

Figure 15. Unique MACs observed per day in different monitors. Lower no of MACs are observed during weekends.

One interesting application of passive Wi-Fi tracking is traffic flow or congestion monitoring. Wi-Fi tracking produces two possible measures for this type of application: speed and travel time estimates, as output by the trajectory estimator, and proportional volume. Actual volume cannot be measured directly, since not all vehicles contain smartphones and less than 100% of phones are detected by our monitors. As an indication of the significance of the volume measure, Figure 15 shows the average unique devices observed across four different monitors for each day over a two-month period. A clear weekly pattern emerges across all monitors: each Saturday is marked by a vertical bar. The astute reader will observe a large drop across all monitors on Thu Nov 24 and Fri Nov 25, corresponding to the American Thanksgiving holiday.

A diurnal pattern is also clearly detectable. Figure 16 shows the number of unique MAC addresses observed in each of our 6 permanently deployed monitors, per hour. These monitors are deployed relatively far from the street, at an indoor location near streets that are only moderately busy. This explains the somewhat low hourly numbers compared to our temporary deployments on traffic poles. However, the pattern stands out clearly across all 6 monitors, over

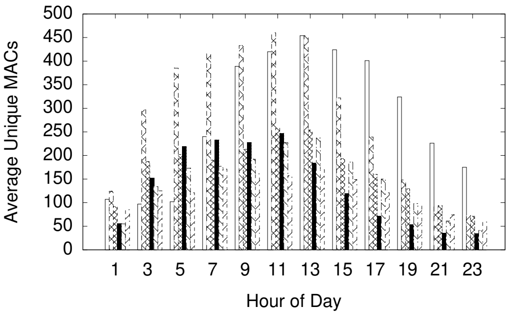

Figure 16. Average hourly unique devices observed in six monitors over 9 months.

a 9 month period. The lack of a clear "rush hour" pattern is explained by the significant presence of pedestrians on campus passing near these monitors.

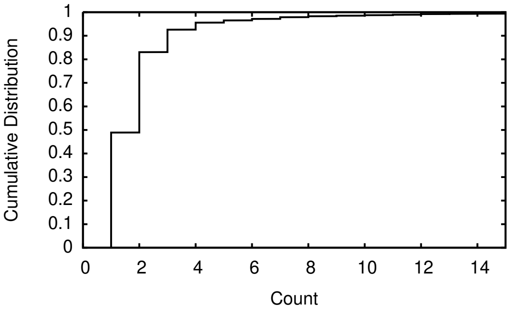

Figure 17. CDF of number of times a MAC is detected in the same monitor during a day.

Figure 17 shows CDF of how many times a MAC is observed in a particular monitor. Here the majority of MACs were observed once or twice per day, as may be expected from a typical commuting behavior.

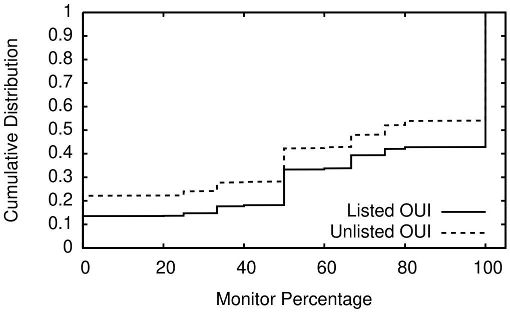

Figure 18. CDF of fraction of monitors detecting a given device, for listed and unlisted OUI.

We are also interested in determining, for phones other than our own, how often they are detected as they pass by our monitors. That is, for a randomly chosen phone and monitor, what is the probability that the monitor detects the phone as it passes by. This number cannot be accurately computed without knowing how many phones passed each monitor, a number which we cannot collect. As a proxy, we report the probability of detecting a phone, given that it can be detected.

Even computing this number can be a challenge, given that the actual trajectory of each phone is unknown. We use the following method. Let our monitors be numbered 1–7 in the east–west direction, as shown in Figure 13. For each unique phone, calculate the largest numbered and lowest numbered monitor that detected the phone. If these are separated by at least one monitor, assume that the phone took the shortest route, and therefore passed by the intermediate monitors. For each such intermediate monitor, check whether it detected the phone passing by.

Figure 18 shows CDFs of the ratio between monitors passed by, and the number of monitors detected for both listed and unlisted OUI. Roughly 50% of phones were detected by 100% of the monitors passed. Over all samples, the mean probability of detection was 68%. Note the similarity of distribution between MACs with listed and unlisted OUIs, which suggests that the unlisted MACs do represent actual smartphones.

Throughout our experimentation with passive Wi-Fi tracking, we have made a number of observations about the typical Wi-Fi transmission behaviors of iPhones in various states of activation. Table 2 lists some of our more salient observations. Overall, if a phone is in active use in some way, it tends to be responsive over Wi-Fi. We also found that some (but not all) iOS 5.0 phones scan at a higher than typical frequency, sending probe messages every minute. We were unable to find what causes an iOS 5.0 phone to exhibit this for our purposes highly beneficial behavior.

8.3 Trajectory Estimation Accuracy, Driving and Walking

As discussed in §4, our trajectory estimator outputs timestamped positions of every moving device detected by our network of Wi-Fi monitors. To evaluate the accuracy of this trajectory estimate, we calculate the distance between each timestamped position from the trajectory estimate, and the reported GPS position for our ground-truth trips collected by foot and by car.

Figure 19 shows a single trip as an illustrative example. In the picture on the bottom, monitors are marked by a yellow pin, and a selection of relative positions are shown as a green (true position) and red (estimated) car. The graph above shows the distance error of the estimate vs. the ground truth over time. As the graph clearly shows, distance error varies significantly over the course of the drive, but is limited to ≈200m. By comparison, monitors are placed on average 460 meters apart, and the maximum detection range as shown in Figure 4, is ≈250m.

Figure 20 shows the CDF of this distance error, separated by driving and walking traces. The mean error is across all traces is 69 meters, or about $\frac{1}{7}$th of the distance between monitors. The difference in distribution between walking and driving traces is primarily explained by the relatively slower speed of walking.

Intuitively, the performance of the Viterbi algorithm should be better for higher granularity observations. To quantify this effect, we remove data for one or more mon-

State	Behavior
Idle	Sends probe request every 8 minutes for 30-60 minutes. Radio is off other times.
Idle (some iOS 5.0 phones)	Sends probe requests every minute, radio is off other times.
Active from standby	Wi-Fi is on and scanning (with probe requests).
Background app (email)	Wi-Fi is on probe requests are sent when app need to data access
Associated and playing music	Wi-Fi radio is on and responding to RTS
Associated and being charged	Wi-Fi radio is on and responding to RTS

Table 2. Observations of iOS Wi-Fi behaviors. These are largely qualitative observations backed up by informal experiments. Others are advised to verify these behaviors before relying on them for their own research.

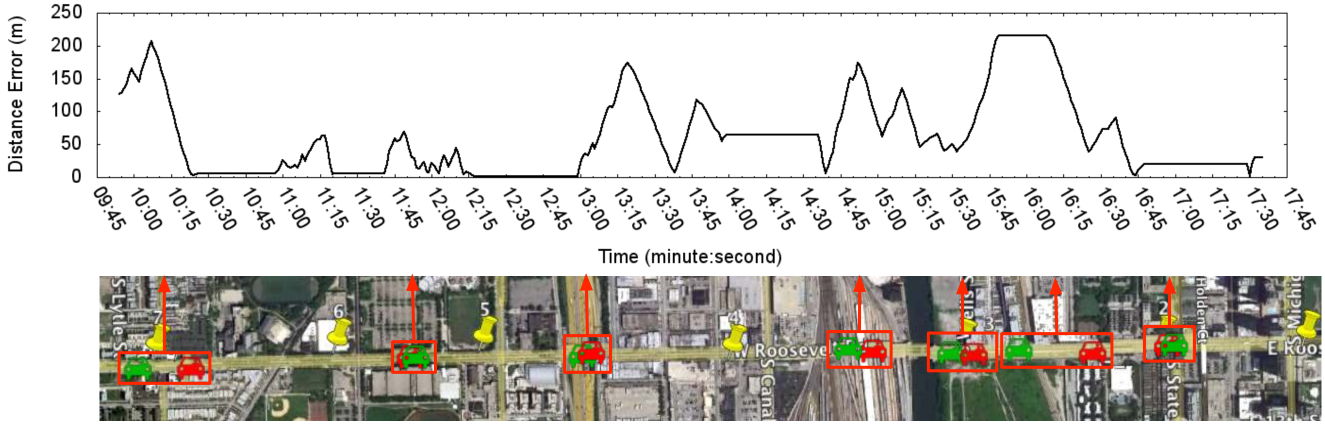

Figure 19. Distance error of position estimate, with respect to ground truth over a 2.8 km route. For a sense of scale, the red car in the map shows estimated position, and the green car is the ground-truth GPS location for a few selected times. Actual time of each relative position on the time plot indicated by the red arrow. Mean and median distance erros are 61 and 76 meters respectively.

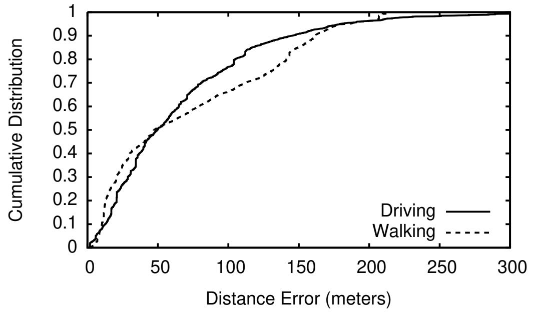

Figure 20. CDF of distance error between ground truth and Viterbi location estimation for driving and walking.

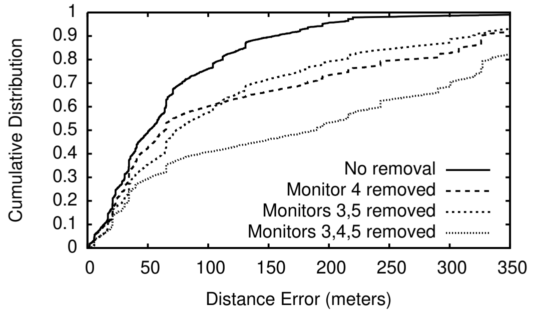

Figure 21. CDF of distance error of Viterbi algorithm as monitors are removed

Time	Dir.	Mean Speed	Std. Dev.	#
10:00-11:00	East	21.6 km/h	7.4 km/h	30
10:00-11:00	West	21.2 km/h	9.3 km/h	34
18:00-19:00	East	12.8 km/h	3.1 km/h	34
18:00-19:00	West	11.9 km/h	8.6 km/h	25

Table 3. Speed and count for two hours on a 1.5 km stretch of Roosevelt road.

itors from our traces and see how the performance of the Viterbi algorithm degrades. This is illustrated at Figure 21. Here, monitors were removed from the center of the deployment, with the worst-case scenario having a maximum inter-monitor distance of almost 2 km. We note that while results are significantly degraded, this degradation is graceful.

Finally, the table above shows the mean and standard deviation of the speed of travel on a 1.5 km stretch of Roosevelt road. Here, the 10:00 time is representative of off-peak conditions, and the 18:00 time represents rush hour traffic. The striking difference in speed between peak and non-peak driving was expected, but the similarity between east and west-bound lanes was surprising. We suspect this somewhat unusual symmetry is due to the heavy mix of business and residential buildings in the area. The column labeled # shows the number of vehicles that were detected traversing the entire stretch of road in that hour. The somewhat low number of vehicles traversing the full stretch of road can be explained

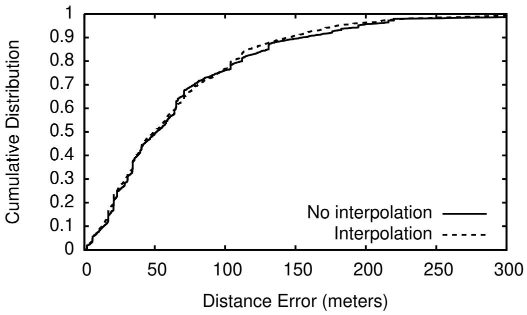

Figure 22. CDF of distance error for Viterbi position estimation with interpolation and without interpolation. Mean error for interpolated, non-interpolated are 69m, 73m respectively. RMSE was 91m, 101m respectively.

by the influence of highway entrances and exits.

8.4 Error-Minimizing Estimates for Periods of Non-Detection

In §4.7, we discuss the use of error-minimizing estimates for periods of non-detection. Figure 22 shows the impact of this post-processing step. An improvement can be seen in the large error ranges. Overall, this technique yielded a mean error reduction of 4 meters (RMSE 10 meters).

9 Conclusion

In conclusion, we have demonstrated that tracking unmodified smartphones using Wi-Fi monitors is both practical, economical and accurate. Based on measurements from several real-world deployments, we find that many smartphones are trackable using the techniques described here.

The accuracy of passive Wi-Fi tracking depends heavily on the density and geometry of deployment. However, using monitors spaced over 400 meters apart, we achieved a mean error under 70 meters as compared to GPS ground truth. Given the low equipment cost, good accuracy and high coverage of our proposed system, we believe it may be suitable for large-scale deployment in urban areas. In such areas, near real-time, high-coverage measurements of surface-street traffic flow would likely be beneficial to commuters and planners alike.

10 References

[1] http://trafficcast.com/products/view/blue-toad/.

[2] http://www.bliptrack.com/.

[3] http://www.pathintelligence.com/.

[4] IEEE OUI Registry. http://standards.ieee.org/develop/regauth/oui/oui.txt.

[5] C.-N. Anagnostopoulos, I. Anagnostopoulos, I. Psoroulas, V. Loumos, and E. Kayafas. License plate recognition from still images and video sequences: A survey. *Intelligent Transportation Systems, IEEE Transactions on*, 9(3), sept. 2008.

[6] P. Bahl and V. Padmanabhan. Radar: an in-building rf-based user location and tracking system. In *Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE (INFOCOM '00)*, volume 2, 2000.

[7] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low-cost outdoor localization for very small devices. *Personal Communications, IEEE*, 7(5), oct 2000.

[8] T. Chothia and V. Smirnov. A traceability attack against e-passports. In *In 14th International Conference on Financial Cryptography and Data Security 2010, LNCS*. Springer, 2010.

[9] B. J. Dil and P. J. M. Havinga. Stochastic radio interferometric positioning in the 2.4 ghz range. In *Proceedings of the 9th ACM Conference on Embedded Networked Sensor Systems (SenSys '11)*, pages 108–120, New York, NY, USA, 2011. ACM.

[10] I. S. et. al. Place lab: Device positioning using radio beacons in the wild. In *In Proceedings of the Third International Conference on Pervasive Computing (Pervasive '05)*. Springer, 2005.

[11] M. Hamedi, R. Fish, and A. Haghani. Data collection of freeway travel time ground truth with bluetooth sensors. In *Transportation Research Record: J. of the Transportation Research Board*, volume 2160, 2010.

[12] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom '03)*. ACM, 2003.

[13] R. Herring, A. Hofleitner, P. Abbeel, and A. Bayen. Estimating arterial traffic conditions using sparse probe data. In *Intelligent Transportation Systems, 2010 13th International IEEE Conference on (ITSC '10)*.

[14] M. Kim, D. Kotz, and S. Kim. Extracting a mobility model from real user traces. In *25th IEEE International Conference on Computer Communications. Proceedings (INFOCOM '06)*, april 2006.

[15] K. Kwong, R. Kavaler, R. Rajagopal, and P. Varaiya. Arterial travel time estimation based on vehicle re-identification using wireless magnetic sensors. *Transportation Research Part C: Emerging Technologies*, 17(6), 2009.

[16] K. Kwong, R. Kavaler, R. Rajagopal, and P. Varaiya. Real-time measurement of link vehicle count and travel time in a road network. *Intell. Transport. Sys.*, 11, Dec 2010.

[17] H. Lu, S. Zhang, X. Liu, and X. Lin. Vehicle tracking using particle filter in wi-fi network. In *Vehicular Technology Conference Fall, 2010 IEEE 72nd (VTC '10)*, sept. 2010.

[18] M.Ndoye, V. Totten, B. Carter, D. Bullock, and J. Krogmeier. Vehicle detector signature processing and vehicle reidentification for travel time estimation. In *Proceedings of 88th Transportation Research Board Annual Meeting (TRB '08)*, 2008.

[19] P. Mohan, V. N. Padmanabhan, and R. Ramjee. Nericell: rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM Conference on Embedded Networked Sensor Systems (SenSys '08)*, New York, NY, USA, 2008. ACM.

[20] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys '04)*. ACM, 2004.

[21] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall. 802.11 user fingerprinting. In *Proceedings of the 13th annual ACM international conference on Mobile computing and networking (MobiCom '07)*, pages 99–110, New York, NY, USA, 2007. ACM.

[22] I. Rose and M. Welsh. Mapping the urban wireless landscape with argos. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*, 2010.

[23] I. Rouf, R. Miller, H. Mustafa, T. Taylor, S. Oh, W. Xu, M. Gruteser, W. Trappe, and I. Seskar. Security and privacy vulnerabilities of in-car wireless networks: a tire pressure monitoring system case study. In *Proceedings of the 19th USENIX conference on Security (USENIX Security'10)*, pages 21–21, 2010.

[24] T. S. Saponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno. Devices that tell on you: privacy trends in consumer ubiquitous computing. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium (SS '07)*. USENIX Association, 2007.

[25] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson. Cooperative transit tracking using smart-phones. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*. ACM, 2010.

[26] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*, pages 85–98, New York, NY, USA, 2009. ACM.

[27] W. Xi, Y. He, Y. Liu, J. Zhao, L. Mo, Z. Yang, J. Wang, and X. Li. Locating sensors in the wild: pursuit of ranging quality. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*. ACM, 2010.