Bringing CUPID Indoor Positioning System to Practice

Souvik Sen†, Dongho Kim†, Stephane Laroche‡, Kyu-Han Kim†, Jeongkeun Lee†
†HP Labs, ‡HP Networking
{souvik.sen, dongho.kim, stephane.laroche, kyu-han.kim, jklee}@hp.com

ABSTRACT

WiFi based indoor positioning has recently gained more attention due to the advent of the IEEE 802.11v standard, requirements by the FCC for E911 calls, and increased interest in location-based services. While there exist several indoor localization techniques, we find that these techniques tradeoff either accuracy, scalability, pervasiveness or cost – all of which are important requirements for a truly deployable positioning solution. Wireless signal-strength based approaches suffer from location errors, whereas time-of-flight (ToF) based solutions provide good accuracy but are not scalable. Recent solutions address these issues by augmenting WiFi with either smartphone sensing or mobile crowdsourcing. However, they require tight coupling between WiFi infrastructure and a client device, or they can determine the client's location only if it is mobile. In this paper, we present CUPID2.0 which improved our previously proposed CUPID indoor positioning system to overcome these limitations. We achieve this by addressing the fundamental limitations in Time-of-Flight based localization and combining ToF with signal strength to address scalability. Experiments from 6 cities using 40 different mobile devices, comprising of more than 2.5 million location fixes demonstrate feasibility. CUPID2.0 is currently under production, and we expect CUPID2.0 to ignite the wide adoption of WLAN-based positioning systems and their services.

Categories and Subject Descriptors

H.3.4 [**Information Systems and Retrieval**]: Systems and Software

Keywords

Wireless; Localization; Application; Indoor positioning

1. INTRODUCTION

Indoor positioning based services are recently attracting extensive interests from enterprises and consumers alike. While the primary goal of indoor positioning, notably in hospitals, malls and stadiums, is to provide navigation aid [1], others use indoor positioning to better market to customers [2], provide just-in-time information via audio for tours [3], connect people of interest in proximity to one another [4]; and ensure asset tracking and security in the enterprises [5]. The U.S. Federal Communications Commission (FCC) aims to use indoor positioning to provide timelier and more effective emergency services such as 911 calls [6]. The advent of IEEE802.11v standard, which allows location related information to be shared with the client from the WLAN network, has further propelled interest in the industry [7]. Several other standards activities involving access point location databases, navigation route representation, Geo-location APIs and accuracy testing, together can fuel pervasive adoption of indoor localization techniques [8]. While there exists numerous innovative solutions to improve the accuracy of indoor localization [9], we still face steep challenges to apply those techniques from research to practical scenarios due to various critical constraints such as accuracy, scalability, cost, energy and pervasiveness.

Accurate yet scalable and pervasive positioning is challenging. Manual WiFi fingerprinting [10–13] or additional infrastructure deployments [14, 15] achieve accuracy but are known to be costly propositions. Crowdsourcing solutions reduce the cost of fingerprinting [16, 17] but are slow to adapt to changes in the environment. Innovative triangulation and trilateration based schemes either require a high density of access points (APs) or sophisticated multi-antenna systems [18–21]. Sensing based solutions are limited to smartphones and tablets and can only locate the client after it has moved significantly [22–24]. In this paper, we rely on network-based positioning using only the WLAN infrastructure because it offers advantages in cost-efficiency (i.e., positioning uses the already available communication infrastructure), device-openness (i.e., any device compliant with the communication standards can be localized), energy-efficiency (i.e., client device need not continuously run any positioning software), and pervasiveness (i.e., requires only WiFi, a dominant technology across all end devices).

We present CUPID2.0 which achieves a mean localization error of $1.8m$, without requiring any fingerprinting or calibration, client coordination, software or mobility, and can locate hundreds of devices, every second with reasonable network overhead. We have implemented, deployed and analyzed CUPID2.0's performance at 6 cities across 2 different continents for more than 14 months. Extensive evaluation results comprising of 40 different mobile devices and more than 2.5 million location fixes demonstrate that CUPID2.0

WWW 2015, May 18–22, 2015, Florence, Italy.
ACM 978-1-4503-3469-3/15/05.
http://dx.doi.org/10.1145/2736277.2741686.

can enable large scale adoption and deployments. Consequently, CUPID2.0 has been productized and announced by HP Networking, as *HP Location Aware* [25, 26].

To design CUPID2.0, we revisit on our previous proposals [24,27] with significantly different and realistic constraints. In our previous proposals, we observed that WiFi-based indoor positioning is primarily affected by multipath. Whenever the direct path [1] is relatively weak due to a blockage (figure 1), the signal strength or propagation time of the wireless signal is biased by the stronger reflected components that traverse longer distances than the direct path. Therefore, to eliminate the effect of multipath on distance estimation, we devised mechanisms to extract the *Energy of the Direct Path (EDP)* [27] and the *Time-of-Flight of the Direct Path (TFDP)* [24]. However, EDP is fundamentally susceptible to shadowing caused by blockages, and TFDP is not scalable as it requires several data-ACK exchanges between the AP and the client. In our earlier work we sidestepped these limitations by augmenting them with mobile sensing. However, this limited us to locate the client only after it has moved significantly. To eliminate the dependency on client mobility, in this paper we build CUPID2.0 as a WLAN-based solution that only uses AP-based trilateration techniques.

We find that unlike EDP, TFDP is largely unaffected by environmental variations. Apart from multipath, TFDP is also not susceptible to wireless shadowing. This is because, while any occlusions can absorb and reduce the energy of the direct path significantly, the propagation time of the direct path is affected only while it traverses through the blocking object, the dimensions of which are significantly lower than the distances traversed by the direct path through the air. Based on these observations, we focus primarily on TFDP to design CUPID2.0 and further investigate its challenges.

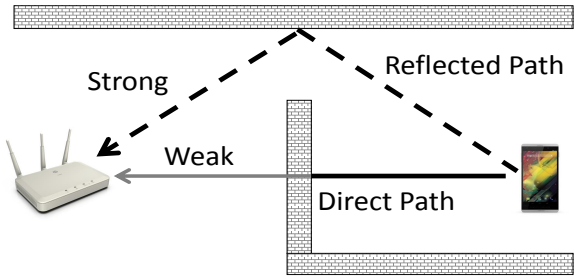

Figure 1: Wireless signal traverses through multiple paths, one direct, and a few reflected paths.

Although TFDP-based distance estimation is feasible, determining the client's location based on TFDP results into unique set of practical challenges:

- **Hardware noise:** Trilateration uses multiple APs for location estimation, thus often requiring distance computation from far-away APs. We find that if the received signal at the AP from the client is weak, the physical layer sometimes adds a fixed value to the correct TFDP, making distance estimation from distant APs difficult.
- **Client diversity:** TFDP estimation requires timely response to the AP's data packet from the client. We find that different client WiFi chipsets wait for different duration before responding to the AP's packet, making it difficult to correctly estimate the TFDP and therefore adversely affecting the pervasiveness of our solution.
- **Scalability:** TFDP measurements are inherently noisy because of the limited bandwidth of WiFi. The effect of random noise can be alleviated by aggregating the TFDP measurements from several data-ACK exchanges between the client and the AP [27]. However, this causes a large network overhead, particularly with multiple APs. Further, neighboring APs are typically configured on different channels, requiring channel switching for location estimation, adding to the network overhead.
- **Energy efficiency:** Apart from the AP to which the client is associated with, it may not respond to the other APs' data packets as its WiFi radio will mostly be off due to the use of power-save mode (PSM). Forcing the client to be awake will reduce its energy efficiency.

In this paper we systematically address the challenges identified above to bring accurate WiFi-based indoor positioning to practice. Briefly, our solution combines complementary characteristics of TFDP and EDP to enable accurate yet scalable positioning (Section 3.5). We introduce smart packet overhearing and scheduling algorithms that reduce the network overhead and avoid the client's PSM problem (Section 3.2). We address the client heterogeneity and hardware noise issues via intelligent AP selection algorithms (Section 3.3,3.4). To the best of our knowledge, this paper is the first to present a year-long real-world deployment of a WiFi-based positioning system, and address the thus-learnt practical challenges. We believe that the implementation, deployment, testing and measurements of CUPID2.0 will help researchers and practitioners further develop indoor location-based systems, services, and applications.

2. BACKGROUND AND PROBLEM IDENTIFICATION

Trilateration based positioning using WiFi APs requires accurate distance estimation. We review two distance estimation techniques (signal strength and time-of-flight) and explain challenges and opportunities in using them.

2.1 Signal strength-based approach

Signal strength-based distance estimation approach [18, 27–31] leverages the fact that the wireless signal suffers attenuation proportional to the distance between the transmitter and the receiver. If P_0 is the received signal strength (RSSI) at a distance of $1m$ from the client and P_R is the RSSI at the AP, then the distance between them can be calculated using the path-loss equation:

$$P_R = P_0 - 10\gamma log(d) \quad (1)$$

γ is a parameter called the path loss exponent that depends on the wireless multipath characteristics as well as any shadowing or occlusions created by walls, windows or even the human body. Since γ is susceptible to environmental variations, signal strength-based distance estimation using RSSI is known to be inaccurate [18]. To address the adverse effect of multipath on distance estimation, our previous work [27] primarily uses only the Energy of the Direct Path (EDP), ignoring the multipath reflections between the client and the AP. EDP improves performance over RSSI because RSSI includes the energy carried by multipath reflections which

[1]Direct path signal is the signal component that traverses along the straight line joining the client and the AP.

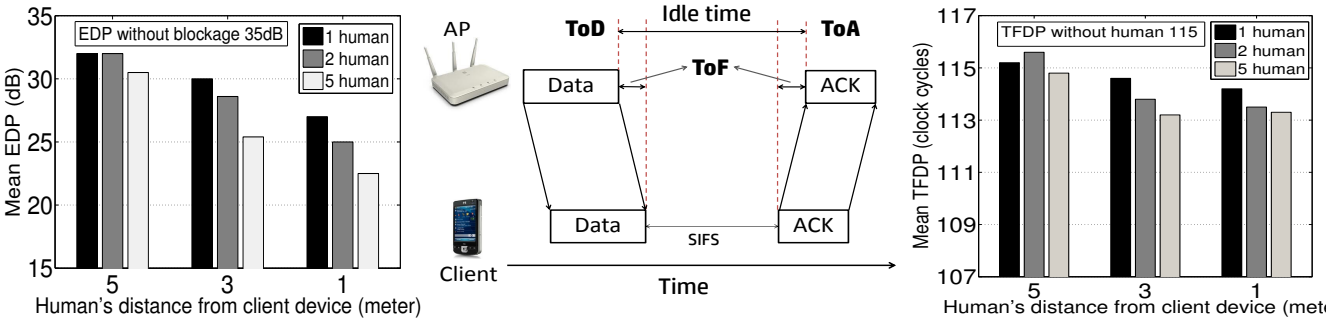

Figure 2: (a) EDP recorded at the AP for different shadowing scenarios. Client is $10m$ away. (b) ToF computation based on data-ACK exchange. (c) TFDP recorded at the AP for different shadowing scenarios.

Orientation	East	West	North	South
Vertical	42.3	43.8	39.7	39.3
Horizontal	36.2	36.5	33.8	34.9

Table 1: Mean EDP (dB) for different phone orientations at $5m$ distance. East implies facing the AP.

travel longer distances than the actual distance between the client and the AP (figure 1). However EDP is susceptible to shadowing. Figure 2(a) plots the average EDP values when the client-AP link is occluded by human bodies. The shadowing effect due to human bodies is hard to model as it depends not only on the number of human beings but also the proximity of the human to the client device. The modeling of the shadowing effect is even more complicated if such occlusions happen along with walls, pillars, windows and other building structures. More importantly, the effect of shadowing changes frequently due to environmental changes, making it difficult to estimate and track.

Signal strength based ranging also relies on the knowledge of the transmit power of the client. However, different mobile devices may use different transmit power values, making it difficult to apriori estimate the same [18] (table 5). P_0 in equation 1 also depends on the gain and polarization of the client's antenna. Therefore, even for the same device, different device orientations result into different P_0 values because orientation affects the antenna polarization as observed by the AP (table 1). It may be possible to adjust the value of P_0 depending on the orientation estimated from the client's gyroscope or compass [32]. However, such approaches are not client agnostic as they require application-level feedback at the client. Therefore, due to uncertainties in wireless shadowing, transmission power and antenna polarization, it is difficult to solely use the EDP to accurately estimate the distance between the client and the AP [24].

2.2 Time-of-Flight based approach

Time-of-Flight (ToF) captures the propagation time of a wireless signal from the client to the AP. It is proportional to the distance between the client and the AP because longer the distance greater will be the wireless propagation time. In the context of WiFi, ToF is measured using a round-trip data-ACK exchange [33]. The AP can precisely calculate the *Time-of-Departure* (ToD) of the data packet when it is sent out into the air by the physical (PHY) layer (figure 2(b)). It can also record the *Time-of-Arrival* (ToA) of the ACK packet based on preamble detection. According to the 802.11 standard, the client is expected to wait for a fixed time duration called Short Inter Frame Space (SIFS) between the data and the ACK packet. Therefore the AP can calculate the ToF and translate it to a distance estimate (d_{ToF}) as follows:

$$ToF = \frac{(ToA - ToD) - SIFS}{2} \quad (2)$$

$$d_{ToF} = c \times ToF \quad (3)$$

where c is the speed of light ($3.0 \times 10^8 m/s$).

As demonstrated in our earlier work [24], the ToF, obtained from the ToA and the ToD, actually captures the distance traversed by the strongest wireless propagation path between the client and the AP. Therefore, in order to calculate an accurate distance, we use the ToF of only the Direct Path (TFDP) [2]. Use of a stronger reflected signal otherwise causes significant error. The TFDP is generally less affected by shadowing and occlusions than the energy of the direct path (EDP). This is because any object blocking the direct path can significantly reduce its energy due to absorption. However, the propagation time of the direct path is affected only while it traverses through the blocking object, the dimensions of which are significantly lower than to the distance traversed by the direct path through the air. Our experiments show that TFDP is not affected by occlusions or proximity of human body (figure 2(c)) and the orientation of the client device (table 2). Even if the direct path is weak due to occlusions or blockage, as long as it is detectable at the AP, the TFDP can indicate the client's distance accurately.

Orientation	East	West	North	South
Vertical	115.1	114.7	115.4	114.2
Horizontal	114.2	114.9	115.8	115.2

Table 2: Mean TFDP (clock cycles) for different phone orientations. East implies facing the AP.

2.3 Challenges in TFDP based location estimation

Although TFDP-based distance estimation is more robust than signal strength based approach, location computation requires distance estimation from multiple APs. Since time-of-flight requires an active data-ACK exchange, this results into unique set of challenges as follows:

Overhead of determining TFDP: Time of Flight of Direct Path (TFDP) obtained from the PHY layer is inherently noisy due to the randomness of preamble detection time, which affects the Time-of-Arrival (ToA) timestamps. Our previous work [24] addresses this problem by leveraging the

[2]Mechanism of TFDP computation is elaborated in [24].

observation that the noise in TFDP values follows a Gaussian distribution and hence can be addressed by employing a Kalman filter [34]. However, as shown in figure 3 Kalman requires more than 10 data-ACK exchanges to estimate the correct TFDP value. This induces network overhead because each AP needs to send several data packets to estimate the client's distance. To solve the overhead problem, we exploit short NULL data packets. We further reduce channel airtime usage by scheduling the NULL packets back-to-back. If t_{NULL} and t_{ACK} are the time duration of the NULL and ACK packet respectively, t_{MAC} is the MAC layer contention time due to backoff and n is the number of NULL packets required to correctly obtain the client's distance from a single AP, then the total channel airtime usage (i.e., overhead) due to positioning is:

$$Overhead = N_c * N_{AP} * (n * (t_{NULL} + t_{SIFS} + t_{ACK}) + t_{MAC}) \quad (4)$$

where N_{AP} is the number of APs used to locate N_c number of clients. E.g., using only 4 APs to track 50 clients, every second, occupies 25% of the wireless airtime, which is prohibitive. Of course the number of NULL packets (n) can be smaller if the AP can leverage existing download traffic to the client. However, location tracking still needs to employ additional packets because the client may not have a continuous stream of traffic. Therefore, the TFDP overheads significantly limit the number of clients that can be tracked.

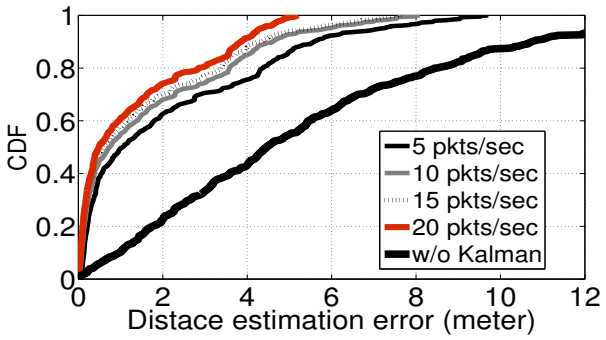

Figure 3: CDF of distance estimation error for different number of packets used by the Kalman filter.

Distance estimation complexity of neighboring APs: Location computation requires distance estimation from neighboring APs, apart from the AP to which the client is associated with. It is possible to obtain an ACK from the client if the neighboring AP impersonates the associated AP using MAC-address spoofing. However, the client may use the power-save mode (PSM) in which it may turn off its WiFi radio and hence it may not even receive the packets sent by the neighboring APs (figure 4). This is because a PSM client only follows the associated AP's beacons to determine when it should turn on its radio to receive the download packets. It ignores the beacons from all other APs. It is not possible to keep the client's WiFi radio always on without adversely affecting its battery life. Thus, to be energy-efficient, we need a solution that respects the client's PSM, yet enables distance estimation from neighboring APs.

Another challenge is that the neighboring APs may be on different channels than the client and its associated AP, requiring channel switching for location estimation. Such channel switchings further adds network overhead because

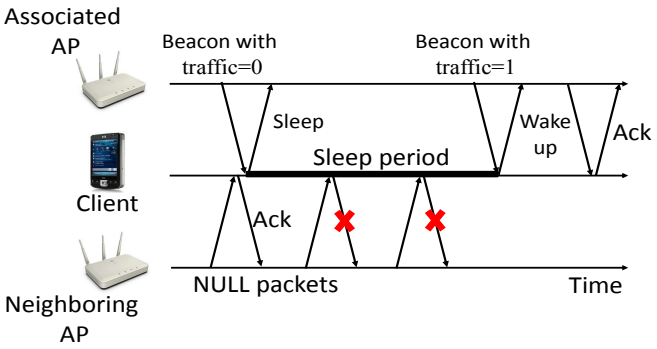

Figure 4: PSM clients will not respond to neighboring AP's packets during the sleep period.

it may consume upto 1ms [3]. We can amortize the channel switching overhead by measuring all the clients in the same channel back-to-back. However, this requires careful scheduling of the NULL packets among all APs. Therefore, TFDP based positioning needs a light-weight scheduling algorithm that ensures low channel-switching overheads.

Device heterogeneity: TFDP is calculated by measuring the time duration between the data-ACK exchange between the AP and the client. Ideally, the duration between the data and the ACK packet should capture only the ToF and an SIFS duration (equation 2). However, chipset manufacturers inculcate a random but fixed time duration to the SIFS [33]. This results into an additional time offset, which we call the *SIFS offset*. We find that the SIFS offset is a fixed number for each WiFi chipset type [33]. Caesar [33] addresses this problem by manually creating a database of per-chipset type and SIFS offset pair, and thereafter accounting for the SIFS offset in TFDP computation by using a lookup based on the client's MAC address. This approach is not scalable, as new chipsets are continuously introduced. More importantly, the MAC address of the client either indicates the mobile device or chipset manufacturer, and not the specific chipset type. Therefore, we need a mechanism that can automatically compute and account for the SIFS offset value, without requiring any manual intervention or relying on MAC-address based lookup.

Error due to signal normalization: Apart from client heterogeneity, another source of error in TFDP is the signal normalization operation in the PHY layer. The PHY layer in the AP can receive the client's ACK over a large range of signal strength values. It prefers to decode the wireless signal within a preferred range (PR) of signal strength. If the incoming signal is weaker or stronger than the PR, it is internally normalized. If the signal is too strong, the RF front-end is set in a low gain mode and the amplifier gain is decreased to avoid Analog to Digital Conversion (ADC) saturation. Weak signals are amplified to ascertain that the quantization noise of the ADC does not hamper correct detection. In our experiments we find that if the incoming ACK signal triggers signal normalization, the TFDP obtained from the PHY layer of Atheros 9390 chipsets is overestimated by a fixed value of 30 clock cycles.

The challenge is that we cannot determine the overestimation, based solely on the client's RSSI. In our experiments we find that the normalization of weak signals can occur between $0-16$dB, PR can occur between $10-48$dB and strong

[3]We have verified that Atheros' software-based Fast Channel Switch (FCS) feature switches channels within 1ms [35].

signal normalization can happen above 40dB. This implies that if the ACK packet's RSSI is between $10-16$dB or between $40-48$dB, additional signal normalization may or may not be invoked. RSSI is not a good indicator of signal normalization operation in the PHY layer because it depends on the incoming signal's peak-to-average power ratio [33]. Therefore, we need a robust scheme that can accurately determine the client's TFDP irrespective of its RSSI value.

3. SOLUTIONS

This section presents our solutions, called CUPID2.0, that address the limitations identified in the previous section. We begin with CUPID2.0's overall architecture.

3.1 Architecture

Figure 5 illustrates the architecture and operation of CUPID2.0. As shown in the figure, CUPID2.0 consists of a location server and multiple APs, each of which talks to a client. Given the architecture, CUPID2.0 employs trilateration to determine the client's location in the following steps. First, the location server instructs a set of selected APs to estimate and share the client's distance estimates. Let d_i is the estimated distance between a client and an AP i. Next, the location server estimates the location $< L_x, L_y >$ of the client by solving a set of trilateration equations using the Gauss-Newton [36] method:

$$d_i = \| < AP_{i,x}, AP_{i,y} > - < L_x, L_y > \| \quad (5)$$

where $< AP_{i,x}, AP_{i,y} >$ is the location coordinate of AP i, recorded apriori. Of course the above architecture entails selection of a minimal set of APs that can yield the client's location accurately and light-weight and robust distance estimation between the APs and the client. Our distance estimation approach is primarily based on the Time-of-Flight of the Direct Path (TFDP), as calculated from the data-ACK exchange between the AP and the client. However, we also combine TFDP with Energy of the Direct Path (EDP) to ensure scalability. In the following subsections we elaborate on the algorithms of CUPID2.0.

3.2 Distance estimation from multiple APs

CUPID2.0 requires multiple APs to conduct data-ACK exchanges adding to its wireless overhead. To address this overhead and to ascertain that neighboring APs can reliably obtain the TFDP information for PSM clients, we propose an overhearing-based scheme which enables neighboring APs to calculate the TFDP of the client without any active data-ACK exchange. Figure 5 shows a high level overview of our overhearing scheme. Whenever the location server needs to compute the client's location, it instructs the AP to which the client is associated with (called the *associated AP*) to begin multiple data-ACK exchanges. The associated AP (AP_2 in figure 5) transmits data packets right after its beacon to ensure that the client is awake. All other neighboring APs (AP_1 and AP_3 in figure 5) do not transmit any probe packets but overhears to compute the Time Of Arrival ($ToA_{2,i}$) of data packet's direct path from AP_2, as well as the ToA of ACK packet's direct path from the client C ($ToA_{C,i}$). The location server can determine the distance between the associated AP and the client using the TFDP between the two ($TFDP_{2,C}$ in figure 5). The location server can also mathematically compute the TFDP between the the i^{th} neighboring AP and the client ($ToF_{i,C}$) as:

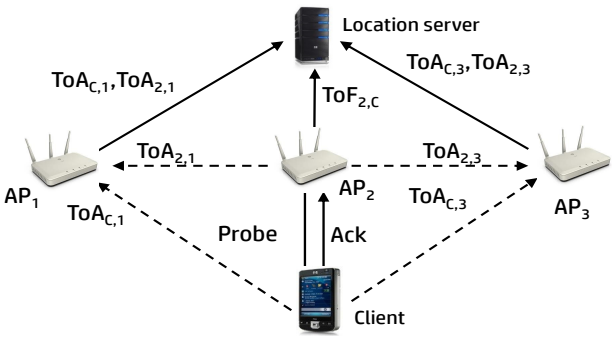

Figure 5: Neighboring APs can estimate the client's distance based on overhearing and recording the ToAs of the associated AP and the client's packets.

$$TFDP_{i,C} = ToA_{C,i} + d_{i,2}/c - ToA_{2,i} - TFDP_{2,C} - SIFS \quad (6)$$

Where $d_{i,2}$ is the distance between the associated AP AP_2 and AP_i and c is the speed of light.

The above overhearing technique does not require any clock synchronization between the APs. The ToA and TFDP values reported by a particular AP is according to its own clock. Equation 6 uses the known distance value between a neighboring AP and the associated AP ($d_{i,2}$), to align their clock values in the same frame of reference. However, the overhearing scheme does require the neighboring APs to switch channels and listen to the associated AP's data transmission. To achieve this, the location server sends distance measurement requests simultaneously to all the selected APs. On reception of the request, the associated AP transmits the NULL data packets right after its next beacon packet (figure 6). This ensures that the client is awake because it reads the beacon's Traffic Indication Map (TIM) and learns that the AP has traffic for itself. Neighboring APs switch its channel to coincide with the associated AP's beacon. In our scheme all APs know each other's beacon transmission times. This is accomplished by using the periodic background scanning operation of the RRM (radio-resource management) functions. During background scanning, an AP records the beacon arrival times of all other APs. Because beacons are transmitted periodically (every $100ms$), the same arrival time can be used to determine when the associated AP will transmit its next beacon.

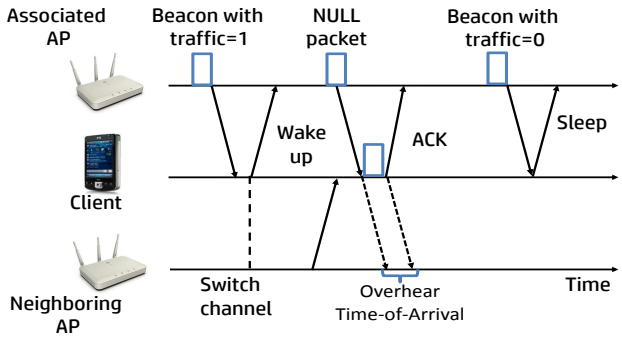

Figure 6: Scheduling NULL data packets after the associated AP's beacon.

3.3 SIFS offset estimation

Each wireless chipset type may have different SIFS offsets (δ_{sifs}) which needs to be accounted for correct TFDP estimation. Due to the δ_{sifs}, the estimated distance by the i^{th} AP (m_i) will differ from the actual distance (d_i) by ϵ meters:

$$\epsilon = \frac{c \times \delta_{sifs}}{2} \tag{7}$$

$$m_i = d_i + \epsilon \tag{8}$$

Observe that the error in distance estimation (ϵ) is a constant and does not vary accross different APs. Therefore, it is possible to numerically estimate ϵ by including equation 8 with our original trilateration equation (5):

$$m_i - \epsilon = \| < AP_{i,x}, AP_{i,y} > - < L_x, L_y > \| \tag{9}$$

The above equation can estimate the location of the client as well as the constant distance error (ϵ), which can thereafter be used to determine the SIFS offset (δ_{sifs}) using equation 7. Because of the additional unknown variable (ϵ), unlike equation 5, the revised trilateration equations require measurements from at least 4 APs. However, once the SIFS offset is estimated, thereafter we use only 3 APs with equation 5 to locate the client.

Access Point Selection: Due to the SIFS offset and its estimation errors, CUPID2.0 needs to carefully choose the set of APs that can participate in trilateration. E.g., in figure 7, if the chosen APs (AP_1, AP_2, AP_3) are collinear with the client, the distance error due to the SIFS offset may add up, injecting errors in location estimation. However, if the chosen APs surround the client (AP_2, AP_3, AP_4), the individual distance errors will often cancel out resulting in a correct estimation of the client's location. To address this, we predetermine a set of triangles that can be formed by the neighboring APs. We begin tracking the client by using all the APs within the client's range. Once the client's location and SIFS offset values are known, we determine a set of 3 APs that surrounds the client. Thereafter, we track the client using the same APs with equation 5.

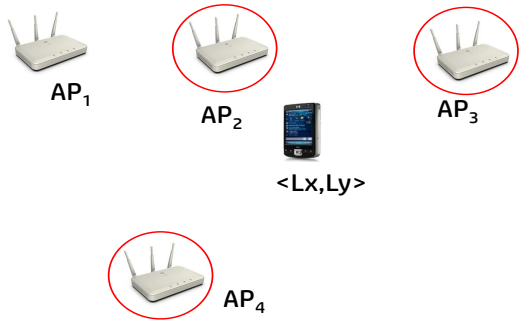

Figure 7: To address the error due to the SIFS offset, AP_2, AP_3 and AP_4 which surrounds the client, should be used for trilateration.

3.4 Dealing with signal normalization

As discussed in section 2, unless the RSSI of an ACK packet falls within $16-40$ dB, the TFDP obtained from the PHY layer of the Atheros chipset can be biased by upto 30 clock cycles due to the additional time required for signal normalization. If the incoming ACK signal is received outside the $16-40$ dB range, the TFDP values may or may not include a timing offset (figure 8(a)). Due to this uncertainty, initially when the client's previous location is not available, we locate the client by using only the APs who receive the client's ACK within the $16-40$ dB range. However, once the client's location is known, we estimate its distance to the other weak or strong APs, denoted as the expected distance. The expected distance value can be used to determine if the TFDP reported by an AP is biased due to signal normalization. To elaborate further, we leverage the observation that the TFDP values recorded at a particular weak or strong AP may follow two distinct distributions(figure 8(a)). Therefore, we employ k-means clustering to distinguish them. If k-means suggest two distinct clusters, then the cluster with the smaller mean value captures the correct TFDP readings. However, if k-means suggests a single cluster, then all the TFDP readings reported by the AP can either be correct or be biased due to signal normalization. In such scenarios, we utilize the AP only if the expected distance of the client to the AP is close to the distance calculated from its TFDP values. Otherwise, we ignore the TFDP readings from the AP as they may be erroneous due to random noise.

3.5 Achieving scalability by combining TFDP with EDP

While our overhearing scheme (section 3.2) reduces overhead by eliminating the data-ACK exchanges from the neighboring APs, we further explore opportunities to improve scalability by reducing the number of packets from the associated AP, required for location estimation. We leverage the fact that EDP and TFDP has complimentary characteristics, which when combined can enable accurate yet scalable positioning. We observe that, unlike TFDP, EDP-based distance estimation is scalable because it mostly requires only a single response from the client. On there other hand, while TFDP is accurate, EDP-based localization is vulnerable to environmental changes. EDP-based distance estimation can be accurate if it is possible to correctly estimate the environment-related path-loss exponent (γ in equation 1) parameter. Since TFDP yields the accurate distance between the client an an AP, we can use it to determine the γ that the same AP should use to estimate the client's distance based only on EDP. However, it is important to periodically use the TFDP to recalibrate the path-loss exponent used by the AP, because the γ may change over time due to environmental variations, resulting into increase in distance estimation error (figure 8(b)).

Figure 8(c) illustrates our scalable distance estimation approach by combining TFDP with EDP. Our scheme operates in rounds of a few seconds, the length of which depends on the airtime overhead acceptable to the administrator. In the beginning of each round, we calculate the client's location using strictly the TFDP estimated from 10 data-ACK exchanges between each AP and the client. Thereafter, we use the TFDP based distance estimates to calculate the per-AP γ value. We calculate P_0 in equation 1 as twice the EDP observed by the strongest AP. This heuristic works reasonably well because the estimated γ is a normalized value based on the P_0 value used. For the remaining duration of each round, we use the previously estimated γ value to determine the client's distance and location strictly based on the EDP. The TFDP-based location estimation overhead (equation 4) therefore reduces to:

$$\begin{aligned} Overhead = N_c * (k * (n * (t_{NULL} + t_{SIFS} + t_{ACK}) + t_{MAC}) \\ +(1-k) * ((t_{NULL} + t_{SIFS} + t_{ACK}) + t_{MAC})) \end{aligned} \tag{10}$$

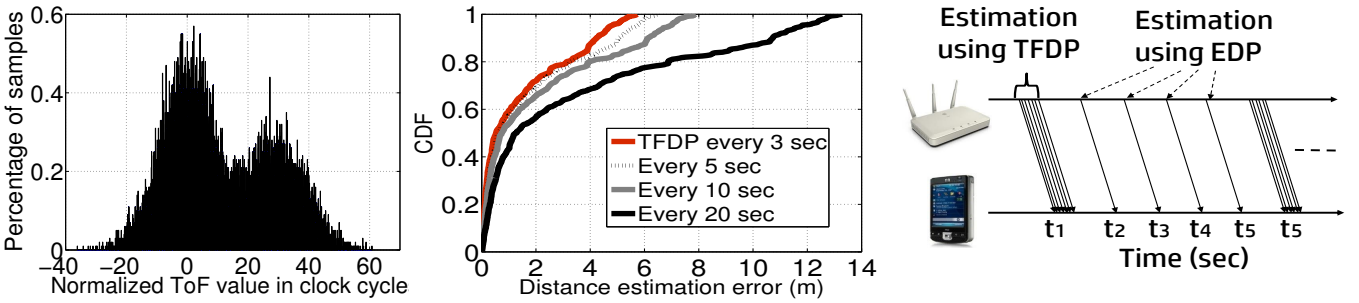

Figure 8: (a) Due to signal normalization, the TFDP recorded at weak or strong APs may fall in two distinct distributions. (b) Distance estimation error using EDP and path-loss exponent calculated using TFDP, at different intervals. (c) Scalable distance estimation using TFDP and EDP.

where k is a fraction indicating the frequency with which the TFDP is used to calculate the client's location. Observe that due to use of overhearing, CUPID2.0's overhead does not depend on the number of APs. As shown numerically in table 3, by using the TFDP every 5 sec, it is possible to locate upto 100 clients with less than 5% airtime overhead.

	No. of clients						
	10	20	30	50	100	150	200
TFDP only	5	10	15	25	50	75	100
CUPID2.0	0.43	0.87	1.31	2.2	4.4	6.6	8.8

Table 3: Percentage of wireless overhead of CUPID2.0 (TFDP every 5 sec) and TFDP-only scheme.

3.6 Scheduling for multiple clients

The scheduling algorithm in CUPID2.0 schedules NULL packet transmissions to each client every second under the wireless overhead constraints (e.g., airtime) given by a network administrator. We define CUPID2.0's airtime overhead as the fraction of airtime that the NULL packets occupy the wireless channel added with the neighboring AP's channel switching time, if applicable. To create an efficient schedule, CUPID2.0 divides the site into non-overlapping triangles formed by joining adjacent APs (figure 9). To locate the clients situated within a particular triangle, CUPID2.0 employs the APs forming the vertices of the same triangle. E.g., in figure 9, clients in $\triangle ABE$ are tracked by APs A, B and E, clients in $\triangle BEF$ are located by APs B, E,f and so on. To improve efficiency, CUPID2.0 needs to simultaneously locate as many clients as possible. This is analogous to scheduling as many triangles in figure 9 in parallel as possible. Two nearby triangles cannot be scheduled simultaneously because they may either share an AP or may be too close to each other, causing interference. But, clients belonging to nearby physical triangles, who do not share any APs, can be located simultaneously as long as they are on different channels. To accommodate this in our schedule, we further divide each physical triangle into upto 3 virtual triangles. Clients within a physical triangle who are all on the same channel belongs to the same virtual triangle. CUPID2.0 needs to create a schedule which will serve all the virtual triangles within the minimum possible time.

To solve the above scheduling problem, we treat each virtual triangle as a node in a conflict graph. An edge between two nodes means that they cannot be scheduled simultaneously. Finding the most efficient schedule of the virtual triangles is analogous to coloring the conflict graph using a minimum

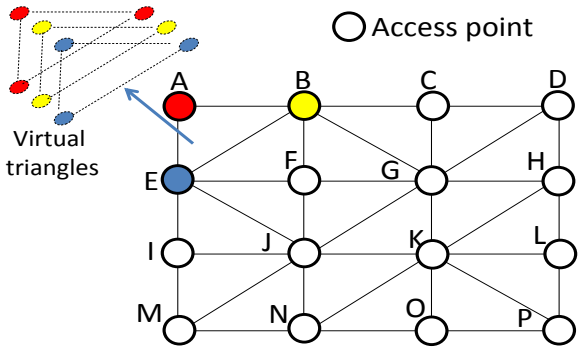

Figure 9: CUPID2.0's scheduling: Each physical triangle can be split into 3 virtual triangles. Virtual triangles who are not in conflict with each other can be scheduled simultaneously.

number of colors, which is known to be a NP-hard problem [37]. Therefore we use a greedy approximation called DSATUR [38] to color the conflict graph. The colors of the conflict graph determines the sequence with which each virtual triangle should be scheduled. Virtual triangles with the same color are scheduled in the same time slot, whereas those with different colors are scheduled in different time slots. The length of each time slot (t_s) depends on the overhead constraint given by the administrator:

$$t_s = \frac{Total\ available\ time}{Total\ no.\ of\ channels} < Overhead \qquad (11)$$

Using equations 10 and 11, we can determine the value of k, which yields the frequency with which the TFDP can be used to determine the location of all the clients in a particular slot. Using the TFDP frequency and the schedule based on the colors of the conflict graph, we can locate all clients, every second, without surpassing the overhead constraint. If there exists a large number of clients, we might not be able to find the value of k that satisfies the overhead constraint. In such scenarios, we use a default of $k = 0.05$, i.e, we use the TFDP every 20 seconds to find each client's location, and we use EDP otherwise.

4. IMPLEMENTATION AND EVALUATION

We evaluate CUPID2.0 using HP MSM 460 APs containing Atheros 9390 chipset tuned at different frequencies using a 40MHz bandwidth. The Atheros 9390 chipset uses a 88MHz clock and can export the Channel State Information (CSI) of any received packet. The reported CSI is a matrix containing one complex number per subcarrier and per receive antenna at the AP. Since the CSI captures the wireless

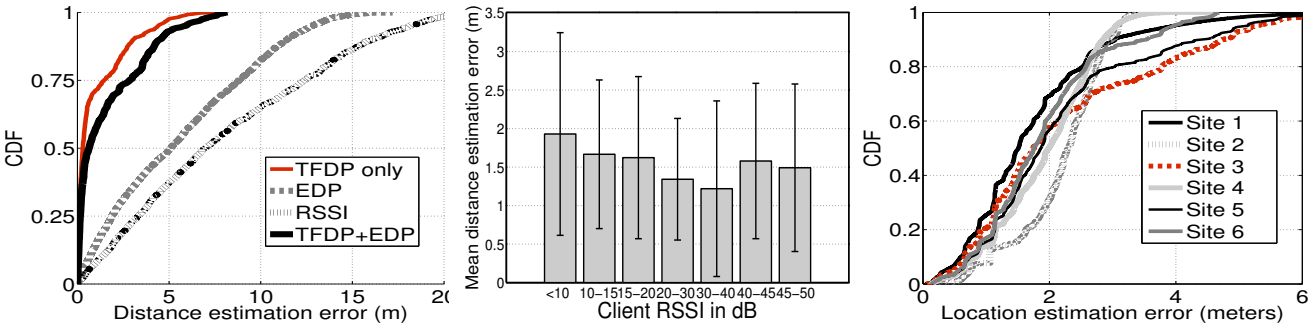

Figure 10: (a) Distance estimation error using different metrics. (b) CUPID2.0's distance estimation error does not vary significantly with signal strength. (a) CUPID2.0's location estimation error at different sites.

channel in the frequency domain, an Inverse Fast Fourier Transform (IFFT) provides the multipath components in the time domain. We determine the Energy of Direct Path (EDP) and Time-of-Flight of Direct Path (TFDP) using the time-domain multipath characteristics [24, 27]. The chipset also reports the Time-of-Departure (ToD) value, in terms of clock cycles, in the transmit-complete descriptor of any data packet. Similarly, it exports the Time-of-Arrival (ToA) and RSSI value of the ACK packet in its receive descriptor. For each data-ACK exchange, we estimate the ToF by using the ToA and ToD, as well as the chipset's clock frequency.

We implement several functions within the Atheros driver to enable packet scheduling, overhearing, channel switching, client admission, time-of-flight, RSSI and CSI reporting. A userspace module at the AP communicates with the AP's driver via netlink and with the location server using TCP sockets. The scheduling, distance and location estimation algorithms are implemented at the location server. It uses the client list from the APs, along with the target network overhead constraint from the administrator, to create a schedule which can report each client's location, every second. Depending on the schedule, the server uses 10 NULL data packets for TFDP or 1 packet for EDP computation.

Methodology

We have deployed the CUPID2.0 positioning system at a total of 6 sites (details in table 4). To evaluate CUPID2.0's performance we design several real-life experiments at each of the sites with APs installed at known locations. At each site, we evaluated CUPID2.0's performance at several locations whose ground truth coordinates were already known. To evaluate mobile scenarios, we asked the users to walk around arbitrarily in the building for an hour during normal office hours, with the smartphone in their pant pocket, or in hand. As each user walks, the AP estimates and stores her location coordinates. To collect ground truth, we pasted numbered markers at known locations. Whenever the user walked through a marker, she recorded its number and the current time. Since the locations of the markers are known, we did interpolation between the markers to obtain the ground truth. The distance between the ground truth and the estimated location is CUPID2.0's instantaneous localization error. CUPID2.0 has been tested by more than 200 users using more than 40 different mobile devices.

4.1 Distance estimation accuracy

CUPID2.0 computes the distance of the client by combining the time-of-flight and energy of her direct path, called TFDP and EDP respectively. CUPID2.0's distance estimation error depends on how frequently it uses the TFDP to recalibrate the EDP's path-loss exponent. Even with a TFDP computation every 5 seconds, mean distance estimation error is less than $1.5m$ (figure 10(a)). RSSI or EDP individually don't perform as well due to susceptibility to environmental variations and causes a mean error of $5.3m$ and $7.2m$ respectively. CUPID2.0's distance estimation accuracy may also depend on the signal strength of the client because of the effect of signal normalization on TFDP values (section 2.2). However by detecting and accounting for the signal normalization offset, CUPID2.0 maintains low errors for all clients, irrespective of signal strength (figure 10(b)).

Site	Site size sq. ft.	No. of deployed APs	No. of unique users	No. of location fixes	Mean error (m)
Site 1	21000	7	>100	∼1.1M	1.7
Site 2	2420	4	>10	∼50000	2.2
Site 3	4300	4	2	∼50000	2.3
Site 4	10200	6	>20	∼400000	1.7
Site 5	37000	14	>10	∼110000	1.9
Site 6	12000	6	>100	∼1.25M	2.0

Table 4: Description of sites where CUPID2.0 was tested, primarily using Galaxy S4 and iPhone 5.

4.2 Location estimation performance

We present CUPID2.0's location estimation performance using more than 500000 location fixes obtained from 6 different sites of varying sizes (details in table 4). Figure 10(c) plots the distribution of location estimation error, per site, when the clients are tracked by using TFDP estimates every 5 seconds and EDP values otherwise. Figure 11(a) compares CUPID2.0's positioning performance with RSSI, EDP, TFDP only techniques. CUPID2.0 demonstrates similar performance as TFDP, but reduces the wireless overhead by using TFDP less frequently. Figure 11(b) shows that CUPID2.0's positioning error is inversely proportional to the TFDP computation frequency. The frequency of TFDP estimation depends on the user specified wireless overhead. To understand the effect of overhead on location accuracy, we performed experiments using a 4300 square feet area at Site 1. Figure 11(c) shows the average location estimation error for different overhead constraints. The average positioning error under high density (40 clients) and low overhead (1% airtime usage) is approximately $3.2m$. The error will be lower if a higher overhead is acceptable ($1.7m$ with 5% overhead) because CUPID2.0 can utilize the TFDP more frequently to deal with the errors in EDP-based location estimation.

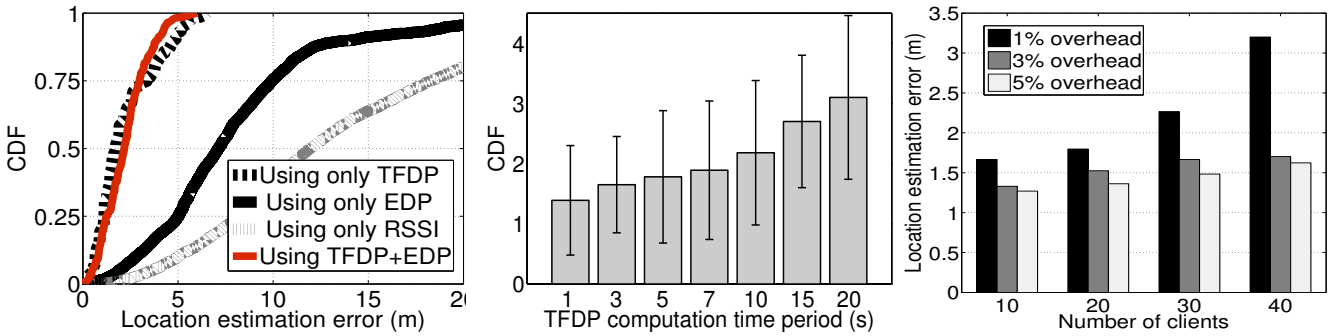

Figure 11: CUPID2.0's location estimation error (a) is similar to that of using only the TFDP and is significantly lower than EDP and RSSI; (b) depends on how frequently TFDP is used for positioning; (c) for different wireless overheads restrictions.

4.3 Effect of device heterogeneity

Device heterogeneity affects CUPID2.0's performance because different client chipsets add an unknown offset to the SIFS duration. CUPID2.0 accounts for the unknown offset by using intelligent access point selection. As a consequence it achieves the same performance as prior schemes which assumes that the SIFS offset is known apriori (figure 12(a)). Table 5 shows CUPID2.0's location estimation error for different mobile devices, as measured at Site 1, using the TFDP every 5 seconds and EDP otherwise. We find that if two mobile devices have the same chipset type, they demonstrate very similar error characteristics. Different chipsets have slightly different timing characteristics and may also use different clock speeds. E.g., Galaxy S3 has a 802.11n chipset containing a 88MHz clock where has Galaxy S4 is 802.11ac enabled with a 160MHz clock. Thus, different chipsets experience different location estimation errors, with the highest error recorded for Galaxy S3 (mean 2.2m).

4.4 Large-scale simulation

To understand CUPID2.0's performance in large venues (e.g., entire shopping mall, airport), we perform trace-driven simulation. In the simulation, we use a floor plan of 200,000 square meter, covered by 625 APs. We vary the number of users per AP as well as the wireless overhead constraint. For each client configuration and overhead constraint, using the scheduling algorithm, we measure how frequently a user can be located using TFDP. Thereafter we utilize our traces to determine her average localization error accordingly (using figure 11(b)). Figure 12(b) shows our simulation results. The mean localization error increases with increasing number of users or with tighter airtime constraint due to infrequent TFDP estimation. However, the localization error is still reasonable in all scenarios demonstrating that CUPID2.0 can achieve scale without sacrificing accuracy.

Device, device count Default count: 1	Chipset	P_0 (dB) (eqn. 1)	Mean error
Iphone 5, **total 8**	BCM4334	39.39	2.1m
Iphone 5S	BCM4334	39.39	2.1m
Ipad Mini	BCM4334	58.19	2.0m
Ipad Air	BCM4334	51.81	1.93m
Ipad 4	BCM4334	50.92	1.91m
Ipad Mini2	—	55.83	1.85m
Galaxy S3	BCM4330	40.29	2.23m
Galaxy S4, **total 7**	BCM4335	43.91	1.65m
Galaxy Note 3	BCM4339	55.83	1.67m
Galaxy Mega	WCN3680	48.89	1.51m
Galaxy Tab 3	Marvell 88W8787	55.54	1.86m
Galaxy Note 10.1	—	45.80	1.92m
HTC One M7	BCM4335	45.15	1.63m
Moto X	WCN3680	53.70	1.45m
Moto Droid Maxx	WL1285C	47.98	2.0m
LG G2	BCM4335	45.17	1.65m
Xperia Z1	—	52.77	1.78m
Kindle Fire	WG7310	53.61	2.1m
Lumia 1020	—	56.49	1.97m
Xperia ZL	—	40.04	1.81m
Kindle Fire HD	BCM43239	54.19	1.64m
Nexus 4	WCN3660	45.65	1.75
Blackberry Z30	—	34.24	2.12m
HTC One M8	—	44.24	1.65m
Lumia Icon	—	40.63	1.84m
Lumia 1520	—	43.81	1.81m
Xperia tab Z	—	48.53	1.78m

Table 5: Performance using different devices.

5. RELATED WORK

The literature about indoor localization is vast [9,39]. Given limited space, in this section, we choose only works relevant to our system to differentiate them from our work.

Signal strength based approach: Most WiFi based indoor positioning systems utilize the RSSI to determine the client's location. Common techniques include pattern matching where the client's location is determined by using a prior established RSSI fingerprint database [10–13] or trilateration based solutions where the client's distance is calculated from multiple APs and later combined to yield the client's location. However, RSSI is known to be a poor estimator of location [40] because it is vulnerable to environmental variations. EZ [18] deals with location errors using GPS fixes which is not guaranteed to be available all the time. Our previous work [27] and FILA [28] attempts to improve signal-strength based solutions using PHY layer solution. But they are still susceptible to wireless shadowing, errors due to client diversity and random phone placements.

Time-based approach: Early time-of-flight based localization schemes required clock synchronization [41] or additional hardware modules [15,42]. In the context of WiFi, the feasibility of computing time-of-flight based on regular data-ACK exchanges were first proposed in [43] and refined in [33]. Our previous work SAIL [24] demonstrated that it is important to eliminate the adverse effect of multipath in

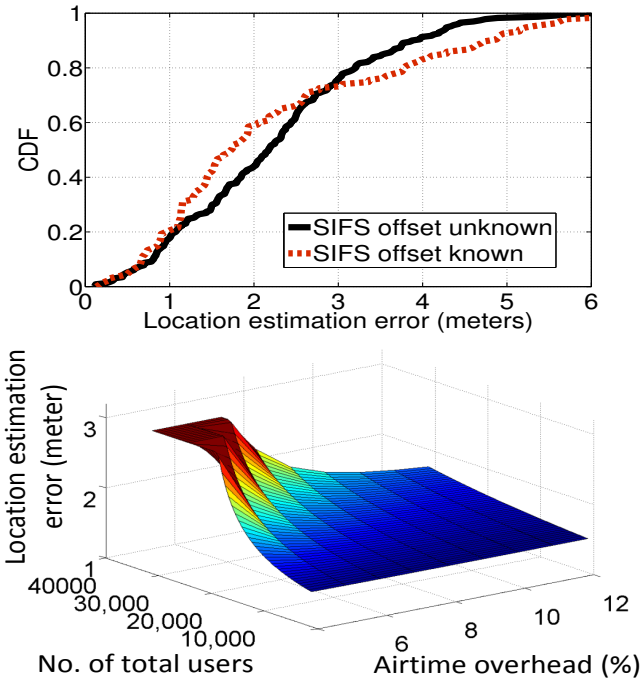

Figure 12: CUPID2.0's location estimation error (a) does not require knowledge of the SIFS offset; (b) for varying number of clients and overhead constraints.

ToF based distance estimation. However SAIL uses smartphone sensing to locate only mobile clients; and it does not address the overhead and deployment issues with ToF-based trilateration. In this paper we systematically identify and address challenges towards a practical and production-ready indoor positioning system based on WiFi.

Angle-of-Arrival (AoA)-based approach: Triangulation based approaches use estimation of the client's angle at multiple APs to localize the client [21,44–49]. However, most of these approaches are not readily deployable as they are accurate only if implemented using sophisticated antenna arrays. Angle estimation on commodity APs results into an average angle estimation error of 20^o [27, 49] making them difficult to use for accurate positioning. Borealis [47], SpinLoc [50] and Ubicarse [48] calculates angles from commodity chipsets, but requires specific user actions and orientation information to compute the location of the mobile device.

Smartphone sensing based: Smartphone-based dead- reckoning techniques has often been used to improve the performance of WiFi-based localization [51]. However these approaches are vulnerable to random fluctuations in sensing data due to drift, compass errors, random phone placements etc. Recent approaches try to address these issues by using crowdsourcing techniques [16, 17, 22]. But, crowdsourcing either requires a strong user incentive model, or leads to coarse grained fingerprint information which reduces their accuracy. Crowdsourcing further cannot adapt to frequent, and automatic configuration changes as adopted by recent WLAN APs; the default update time settings found in [52] are 10mins and 4mins. Map based particle filtering [23, 53] also addresses noise in sensing information, but they rely on unique indoor walking paths such as long corridors, making them ineffective in open areas such as airport, auditorium etc.

6. DISCUSSION & ONGOING WORK

Energy efficiency: CUPID2.0 imposes minimal energy overhead on client devices. Unlike mobile sensing based solutions, it does not require any process to run at the client. Further, CUPID2.0's NULL data packet exchange involves only the PHY layer processing, and thus no other energy cost exists at the client.

Impact of wireless interference: In our year-long experiments, we did not witness any noticeable effect of interference on time-of-flight estimation. We believe this happens because at the AP, the time-of-arrival of the client's ACK packet is detected through sophisticated preamble detection ċitelee2010improved. Further, the client does not carrier sense the channel before transmitting the ACK packet, resulting into no timing issues due to interfering signals.

Multi-floor positioning: CUPID2.0 identifies each floor by simply looking at the top three APs with respect to the received signal strength from the client's ACK. In our evaluation in Site 1, we find that the floor of the strongest 3 APs is always the same as the client's floor. However, this may not be true in atriums. We are currently investigating schemes to enable accurate 3D positioning, based on time-of-flight.

Device-based positioning: Device-based positioning will be possible by transposing CUPID2.0 from the infrastructure to the device. We are currently investigating its feasibility. However, there are advantages in computing the time-of-flight at the infrastructure. First, we may not be able to replicate our overhearing mechanism that improves scalability, at the client device. Second, mobile devices typically have a single antenna resulting into coarser multipath resolution and increased distance estimation error [24]. Third, to perform trilateration, mobile devices may need to scan multiple channels resulting to reduced battery life.

Improving accuracy with 802.11ac: The granularity of time-of-flight reported by the PHY layer depends on the chipset's clock frequency. Our current results use a 802.11n chipset that has a 88MHz clock, and hence provides a distance estimation granularity of 1.7m. We note that CUPID2.0's accuracy will further improve with upcoming 802.11ac based systems that uses a clock speed of 160MHz.

7. CONCLUSION

In this paper, we have presented CUPID2.0, a system that solves many practical challenges for WiFi-based indoor localization. As we explained, CUPID2.0 is the first system that demonstrates accuracy, scalability, low-cost, and energy efficiency in practical settings. Rather than using expensive fingerprinting or energy-hungry mobile sensing, CUPID2.0 uses Time-of-Flight (ToF) based trilateration to determine the user's location with less than 2m error. Furthermore, it combines ToF with signal strength and uses novel overhearing mechanisms to make location estimation scalable and practical. Large scale experimentation at 6 different cities, comprising of more than 240 users, 2.5M location fixes and 40 different mobile devices confirm the performance of CUPID2.0. We believe that our upcoming product called HP Location Aware, which is based on CUPID2.0, will be able to drive pervasive adoption of indoor positioning systems and extend the territory of location-based services and applications.

8. REFERENCES

[1] Techcrunch. How the 49ers are using beacons to help you find hot dogs. *URL http://techcrunch.com/2014/11/04/how-the-49ers-are-using-beacons-to-help-you-find-hot-dogs-and-beer/*, 2014.

[2] Shopkick. Shopkick retail platform. *URL http://www.shopkick.com/*.

[3] Techcrunch. Wifarer brings indoor navigation to the royal bc museum. *URL http://techcrunch.com/2012/08/01/wifarer-brings-indoor-navigation-to-the-royal-bc-museum/*, 2012.

[4] Lokast. Lokast mobile app platform. *URL http://www.lokast.com/*.

[5] ZoneDefense. Zonedefense location-based mobile security platform. *URL http://airpatrolcorp.com/products/zonedefense/*.

[6] FCC. Fcc proposes new indoor requirements and revisions to existing e911 rules. *URL http://www.fcc.gov/document/proposes-new-indoor-requirements-and-revisions-existing-e911-rules*.

[7] 802.11v.

[8] Qualcomm. Discussion of indoor location standards. *URL http://www.cwins.wpi.edu/workshop12/presentation/Standardization_panel/kirk.pdf*.

[9] Y. Gu et al. A survey of indoor positioning systems. *IEEE Communications Surveys & Tutorials*, 2009.

[10] V. Bahl et al. RADAR: An in-building rf-based user location and tracking system. In *INFOCOM*, 2000.

[11] S. Sen et al. Spot localization using phy layer information. In *MobiSys*, 2012.

[12] Yu-Chung et al. Accuracy characterization for metropolitan-scale wi-fi localization. In *MobiSys*, 2005.

[13] M. Youssef et al. The horus WLAN location determination system. In *MobiSys*, 2005.

[14] G. Borriello et al. Walrus: wireless acoustic location with room-level resolution using ultrasound. In *MobiCom*, 2005.

[15] N. Priyantha et al. The cricket location-support system. In *MobiCom*, 2000.

[16] Z. Yang et al. Locating in fingerprint space: wireless indoor localization with little human intervention. In *MobiCom*, 2012.

[17] A. Rai et al. Zee: zero-effort crowdsourcing for indoor localization. In *MobiCom*, 2012.

[18] K. Chintalapudi et al. Indoor localization without the pain. In *MobiCom*, 2010.

[19] J. Xiong et al. Arraytrack: A fine-grained indoor location system. In *NSDI*, 2013.

[20] R. Nandakumar et al. Centaur: locating devices in an office environment. In *MobiCom*, 2012.

[21] D. Niculescu et al. Ad hoc positioning system (APS) using AoA. In *INFOCOM*, 2003.

[22] H. Wang et al. Unsupervised indoor localization. MobiSys, 2012.

[23] F. Li et al. A reliable and accurate indoor localization method using phone inertial sensors. 2012.

[24] A. T. Mariakakis et al. Sail: single access point-based indoor localization. In *MobiSys*, 2014.

[25] HP. HP Location Aware - featuring HP Labs indoor location technology. *URL http://h30507.www3.hp.com/t5/Innovation-HP-Labs/HP-Location-Aware-featuring-HP-Labs-indoor-location-technology/ba-p/157850#.VF59uovF_Kc*.

[26] HP Networking. Sdn-enabled location technology delivers new revenue generation opportunities. *URL http://www8.hp.com/us/en/hp-news/press-release.html?id=1608266#.VF5-RIvF_Kc*.

[27] S. Sen et al. Avoiding multipath to revive inbuilding wifi localization. In *MobiSys*, 2013.

[28] K. Wu et al. Fila: Fine-grained indoor localization. In *INFOCOM*, 2012.

[29] N. Banerjee et al. Virtual compass: relative positioning to sense mobile social interactions. *Pervasive Computing*, 2010.

[30] K. Whitehouse et al. A practical evaluation of radio signal strength for ranging-based localization. *MC2R*, 2007.

[31] K.K. Chintalapudi et al. Ad-hoc localization using ranging and sectoring. In *INFOCOM*, 2004.

[32] Y. Chapre et al. Received signal strength indicator and its analysis in a typical wlan system. In *Local Computer Networks (LCN)*, 2013.

[33] D. Giustiniano et al. Caesar: carrier sense-based ranging in off-the-shelf 802.11 wireless lan. In *CoNEXT*, 2011.

[34] G. Welch et al. An introduction to the kalman filter, 1995.

[35] Qualcomm Atheros. Qualcomm atheros technology overview. *URL http://www.qca.qualcomm.com/wp-content/uploads/2013/11/AR9462.pdf*.

[36] J. Nocedal et al. *Numerical Optimizationn*. Springer series in operations research and financial engineering. Springer, 1999.

[37] M. R. Garey et al. Some simplified np-complete problems. In *STOC*, 1974.

[38] D. Brélaz. New methods to color the vertices of a graph. *Communications of the ACM*, 22(4):251–256, 1979.

[39] J. Hightower et al. Location systems for ubiquitous computing. *Computer*, 2001.

[40] A. J. Khan et al. Experiences with performance tradeoffs in practical, continuous indoor localization. In *WoWMoM*, 2013.

[41] M. Youssef et al. Pinpoint: An asynchronous time-based location determination system. In *Mobisys*, 2006.

[42] M. Ciurana et al. A ranging system with ieee 802.11 data frames. In *RWS*, 2007.

[43] D. McCrady et al. Mobile ranging using low-accuracy clocks. *IEEE Transactions on Microwave Theory and Techniques*, 38(6), 2000.

[44] J. Xiong et al. SecureAngle: improving wireless security using angle-of-arrival information. In *HotNets*, 2010.

[45] L. Cong et al. Hybrid tdoa/aoa mobile user location for wideband cdma cellular systems. *Wireless Communications, IEEE Transactions on*, 1(3):439–447, 2002.

[46] A. Tarighat et al. Improved wireless location accuracy using antenna arrays and interference cancellation. In *ICASSP*, 2003.

[47] Z. Zhang et al. I am the antenna: accurate outdoor ap location using smartphones. In *MobiCom*, 2011.

[48] S. Kumar et al. Accurate indoor localization with zero start-up cost. In *MobiCom*, 2014.

[49] J. Gjengset et al. Phaser: enabling phased array signal processing on commodity wifi access points. In *MobiCom*, 2014.

[50] S. Sen et al. Spinloc: Spin once to know your location. In *HotMobile*, 2012.

[51] R. Harle. A survey of indoor inertial positioning systems for pedestrians. 2013.

[52] Aruba Networks. Configuring adaptive radio management (arm) profiles and settings. Aruba Whitepaper.

[53] J. Hightower et al. Particle filters for location estimation in ubiquitous computing: A case study. *UbiComp*, 2004.