CAESAR: Carrier Sense-Based Ranging in Off-The-Shelf 802.11 Wireless LAN

Domenico Giustiniano
Disney Research Zurich, Switzerland
domenico@disneyresearch.com

Stefan Mangold
Disney Research Zurich, Switzerland
stefan@disneyresearch.com

ABSTRACT

Wireless local area networks have been designed for wireless communication. Frames are acknowledged (ACKed) after a short and predefined MAC idle time. The MAC idle time varies with i) the physical distance between stations, caused by the delay of wireless signal propagation, and ii) the time to detect the ACK at the local station, which varies with the signal strength of the incoming ACK. We present CAESAR, CArriEr Sense-bAsed Ranging, that combines time of flight and signal-to-noise ratio measurements to calculate the distance between two stations. CAESAR measures the distance by estimating the MAC idle time in a data/ACK communication at a 44 MHz clock resolution and the ACK detection time on a per-frame basis. CAESAR is a software-based solution that is entirely implemented at the transmitter and it requires no protocol modifications and only a limited calibration in links with multi-path propagation. We implement CAESAR on commodity hardware and conduct extensive experiments both in controlled network conditions and dynamic radio environments. Our measurements confirm the accuracy of the solution and show the capability to track the distance to WLAN smartphones at pedestrian speeds.

1. INTRODUCTION

Today's IEEE 802.11 Wireless LANs (WLANs) are largely deployed in indoor environments. WLANs have also started to assist satellite navigation systems, such as Global Positioning System (GPS), to mitigate the unfavorable satellite propagation in locations with a limited view of the sky, such as indoor areas and urban canyons. An assisting WLAN localization tool has the competitive advantages of operating without additional infrastructure cost and being available in most of today's mobile devices, such as smartphones. These factors make WLAN localization the preferred assisting technology over other short range communication technologies like infrared [1], RFID [2], ultrasonic [3] and ultra-wideband [4], as confirmed by today's commercial services [5–8].

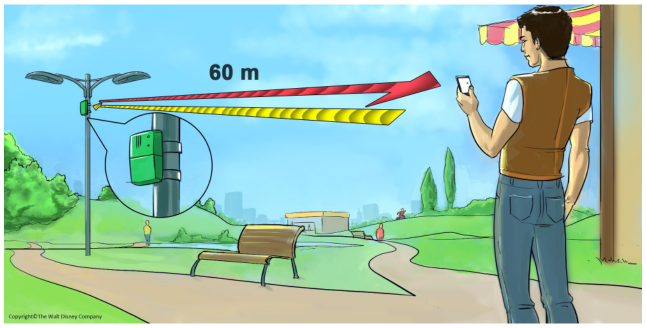

Figure 1: A CAESAR Access Point estimates the distance to a remote station, such as a WLAN smartphone.

Most of the work on WLAN localization has focused on inferring the distance (*ranging*) from a WLAN station to several reference points using *Signal-To-Noise Ratio (SNR)*-based measurements. Despite the low precision and the need for theoretical or empirical models, SNR-based localization is widely used in commercial solutions because it only requires software support in off-the-shelf WLAN devices. As an alternative, *Time-Of-Flight (TOF)* ranging techniques calculate the distance based on the time of propagation between two wireless devices. While TOF is widely used in technologies like GPS, it has received less attention in 802.11 WLAN. The problem is that the 802.11 WLAN technology was not designed for localization but for communication. Hence, WLAN TOF ranging techniques are affected by noise in the measurement, due to clock drift, interaction with system tasks, and protocol overhead. This noise causes low precision, low convergence, and low tracking capability of the estimated distance [9, 10].

In this contribution, we present a WLAN ranging technique based on TOF and SNR measurements to estimate the distance between two WLAN stations. We call it CAESAR, CArriEr Sense-bAsed Ranging, because it takes advantages of the native WLAN carrier-sense. CAESAR makes the following contributions:

- CAESAR uses a WLAN TOF echo technique (i.e. the round-trip time of a signal transmitted to a remote device) that minimizes the error due to noise like clock drift: We devise an echo technique based on the MAC idle time (Short Interframe Space, SIFS) between the data and the subsequent Acknowledgement (ACK). This MAC idle time is the shortest 802.11 MAC timing. It is used to reserve the communication channel, and its duration is predefined and expected to be constant. However, the MAC idle time in a data/ACK communication varies with the physical distance between the two stations because of the delay of wireless signal propagation. CAESAR exploits the variation and the short duration of the MAC idle time for ranging measurements (Section 2).
- CAESAR measures the MAC idle time using carrier sense samples at the resolution of the main WLAN clock, 44 MHz in 802.11b/g. This results in a distance estimated with a quantization error of ≈ 3 m, and it also minimizes the noise due to interactions with system tasks. The MAC idle time varies not only due to the signal propagation time but also as a function of the time to detect the ACK, which is influenced by the SNR of the ACK. CAESAR uses SNR measurements to characterize the dispersion generated by the ACK detection time, thus increasing the accuracy. It does not require, but might gain from calibration in links with strong multipath component (Section 3).
- A low-cost and widely deployed ranging solution does not require any hardware changes, but it just uses what is needed for the correct operations of the 802.11 protocol, as in SNR-based ranging methods. We exploit the reconfigurability of the open-source driver of the Atheros WLAN chipsets [11] to implement CAESAR on commodity hardware. We present a new method for accessing the carrier-sense information in today's chipsets (Section 4).

Our measurements in indoor and outdoor radio environments confirm the accuracy of the solution. For indoor links, CAESAR shows a distance error of less than 1 m in 8 links out of 10 of our testbed, converges after fewer than 25 data frames sent to the remote station, and obtains stable results across different network workloads. This work also demonstrates that TOF can be applied to track the distance to off-the-shelf WLAN smartphones at pedestrian speeds (Section 5).

2. CARRIER SENSE-BASED RANGING

The focus of this paper is the design and implementation of an 802.11 WLAN ranging technique. Ranging is used by a local station (L-STA) to separately calculate the distance d to remote stations (R-STAs) in its coverage area. The question we aim to answer is whether it is possible to design a precise and real-time 802.11 WLAN ranging technique, only exploiting the 802.11 MAC protocol and without requiring either a new PHY layer or adding low-level PHY processing. That goal has multiple conflicting requirements:

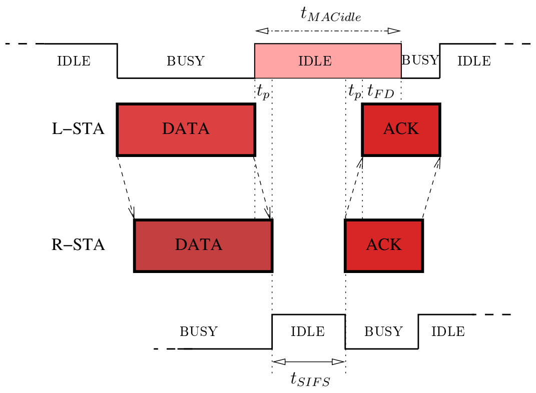

Figure 2: Busy and idle time at the MAC layer differ between the local station (L-STA) and the remote station (R-STA). While a constant idle time is expected at R-STA, according to the 802.11 protocol (SIFS), the idle time at L-STA varies, as a function of the propagation time (t_p) and the ACK frame detection time (t_{FD}).

- **High precision**: L-STA runs a fine-grained geolocation algorithm based on the ranging to R-STAs.
- **Fast convergence**: Any geolocation tracking algorithm requires real-time distance estimation.
- **802.11-compliant ranging**: No changes needed in the 802.11 protocol, to exploit its cost-effectiveness.
- **No additional hardware**: An implementation on commodity hardware, to ease the adoptation of the ranging technique for research and commercial use.
- **Low network usage**: The primary goal of a WLAN network is network communication. Other stations must not be affected by the ranging traffic.
- **Workload independent**: The estimation must be independent of L-STA and network workload.
- **Minimal calibration**: It must be immediately ready to be used by geolocation services.

In this work we present an 802.11 WLAN ranging technique that targets the above requirements, combining a TOF echo technique with SNR measurements. First, we introduce the technique.

2.1 MAC Idle Time

Due to the lack of clock synchronization, WLAN TOF ranging techniques is usually based on echo techniques, where an L-STA measures the round trip time of a signal transmitted to an R-STA [12]. WLAN echo techniques are affected by noise in the measurement, caused by clock drift, interaction with system tasks, and protocol overhead (due to the estimation of the remote processing time). This noise causes low precision, low convergence, and low tracking capability of the estimated distance [9, 10].

To tackle these problems, we devise a TOF measurement technique based on the MAC idle time between the MAC data frame and the subsequent MAC ACK frame. Busy and idle channel states are used to sense the presence (busy channel) or absence (idle channel) of a frame on the medium, according to the 802.11 Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA) protocol. Between the data and ACK communication, the channel is idle for a short time. This is used by an 802.11 station to reserve the channel, so that no other transmissions are possible. This time is the shortest 802.11 MAC idle time, and we exploit it for devising a ranging technique.

As part of the 802.11 CSMA/CA protocol, the L-STA transmits a data frame and expects to receive an ACK from R-STA, signaling the successful reception of the frame (see Fig. 2). The MAC idle time at the R-STA is denoted Short Interframe Space (SIFS), and it is fixed by the 802.11 standard, e.g. $t_{SIFS} = 10\,\mu s$ in 802.11b/g [13]. The MAC idle time at the L-STA, denoted as $t_{MACidle}$, varies with the propagation time of the data $t_{p,data}$ from the L-STA to the R-STA and the ACK $t_{p,ACK}$ from the R-STA to the L-STA. For a speed of light of $c = 300\,\mathrm{m}/\mu\mathrm{s}$, a value of less than $t_{MACidle} = 12\,\mu s$ is expected for typical coverage ranges.

We measure $t_{MACidle}$ using the main 44 MHz WLAN clock. Since the signal travels from L-STA to R-STA and back, the distance resolution for the single sample (i.e., a data/ACK exchange) is $300/(2 * 44) = 3.4\,\mathrm{m}$[1].

2.1.1 Clock drift and interaction with system tasks

Due to tolerances, there is a relative clock drift between L-STA and R-STA. This clock drift causes significant measurement errors with echo techniques [9]. The impact of drift is negligible using an echo technique based on $t_{MACidle}$. Indeed $t_{MACidle}$ is measured over a short period of time and uses the main 44 MHz WLAN clock. For example, a drift of 25 ppm between the local and remote clock [13] results in a drift of 25 s in a million seconds, or 0.13 % of one WLAN cycle in $12\,\mu$s. Measuring $t_{MACidle}$ with the main WLAN clock minimizes also the interaction with the system tasks, because the WLAN clock operations are mostly not influenced by the tasks of the main CPU.

2.2 Channel state in a Data/ACK exchange

To determine what parameters affect $t_{MACidle}$, we identify what factors affect the busy-to-idle and idle-to-busy transitions, referring to the channel state changes illustrated in Fig. 2. Correct identification of frame boundaries, i.e., the start and the end of a new frame, is one of the most critical operations in WLAN networks.

2.2.1 L-STA estimates locally the ACK detection time

The $t_{MACidle}$ starts with the busy-to-idle transition at the end of the data transmission and ends with the idle-to-busy transition at the start of the ACK reception. While the busy-to-idle transition at the end of the data transmission is readily determined by the end-of-frame transmission, the start of the ACK reception requires more discussion. The default operation of an 802.11 WLAN station is reception, wherein it continuously senses the medium via the Clear Channel Assessment (CCA) module and reports the busy and idle channel state at the WLAN clock resolution. The CCA declares the medium as busy when it detects an 802.11-modulated signal (*frame detection*, FD). This operation causes a delay at the receiver from the start of the 802.11 preamble to the idle-to-busy transition, that we call the frame detection time t_{FD}.

2.2.2 Not estimating the remote processing time

Most of the $t_{MACidle}$ at R-STA is the result of the t_{SIFS}. The start and end of the SIFS are determined by PHY operations at the resolution of the main WLAN clock: i) The SIFS starts when the signal strength of the data at R-STA drops below a certain threshold (called the coarse low threshold) at the end of the frame [14]. ii) R-STA starts to transmit the ACK when the SIFS expires. iii) The turnaround time from reception to transmission of the start of the ACK (which is known and constant) occurs during the SIFS [15]. This results in a stable t_{SIFS} and allows us to design a WLAN echo technique without needing additional message passing per-remote station to determine the remote processing time per-frame.

2.2.3 Carrier sense-based ranging

The $t_{MACidle}$ is calculated as (see Fig. 2):

$$t_{MACidle} = t_{p,data} + t_{SIFS} + t_{p,ACK} + t_{FD} \qquad (1)$$

We assume a similar radio-channel for the propagation of the data and the ACK, so $t_p = t_{p,data} = t_{p,ACK}$, and we estimate the distance $\hat{d}$ at L-STA as:

$$\hat{d} = c \cdot (t_{MACidle} - t_{SIFS} - t_{FD})/2, \qquad (2)$$

CAESAR only needs local information, such as the t_{FD}, $t_{MACidle}$, and the SNR, and it does not require any information from the R-STA.

We devise solutions to estimate t_{FD} and $t_{MACidle}$. In Section 3, we present a novel estimator to characterize the dispersion in the measurements due to t_{FD}. The estimator increases the accuracy, filtering a limited set

[1]A reduction of this quantization error occurs with the 802.11n WLAN clock, driven by a clock of at least 88 MHz.

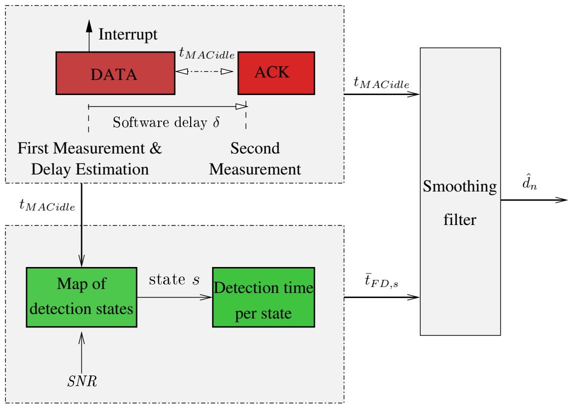

Figure 3: CAESAR block diagram. On the top left, we show the block estimating $t_{MACidle}$: L-STA sends an 802.11 data and locally measures $t_{MACidle}$. On the bottom left, we have the block estimating t_{FD}: L-STA uses the pair $(t_{MACidle}, SNR)$ to infer t_{FD}. $t_{MACidle}$ and t_{FD} samples determine the distance between L-STA and R-STA, when filtered over multiple samples.

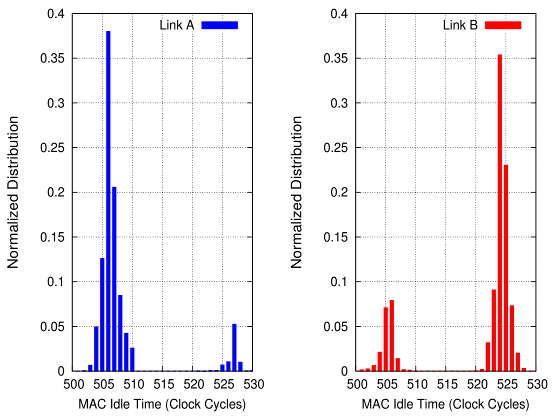

Figure 4: Histogram of $t_{MACidle}$ using commodity WLAN hardware. There are two spikes per-link, that would cause an unacceptable high variability in distance estimation.

of samples. In Section 4, we show how increased software programmability in off-the-shelf wireless devices allows us to estimate $t_{MACidle}$ at the driver-level. This is non-trivial because $t_{MACidle}$ occurs in a short period of time during a data/ACK communication.

3. FRAME DETECTION TIME

We implement CAESAR on commodity Atheros chipsets using the Ath9k driver [11]. Atheros-based drivers are considered a "de-facto" standard for the experimental validation of 802.11 WLAN designs. Fig. 3 shows the block diagram of our implementation. On the top left, we show the block estimating $t_{MACidle}$: L-STA sends an 802.11 data and measures $t_{MACidle}$ by extracting and processing 44 MHz hardware timing counters. On the bottom left, we show the block estimating t_{FD}: L-STA uses the pair $(t_{MACidle}, SNR)$ to infer t_{FD}. Finally, $t_{MACidle}$ and t_{FD} samples we used determine the distance between L-STA and R-STA, when filtered over multiple samples.

3.1 Correlating detection time and SNR

The IEEE 802.11 standard mandates that at least 90% of $\{t_{FD}\}$ samples have to be in an observation time window of 4 μs from the start of the preamble[2] [13]. Although the frame detection implementation is manufacturer dependent, it always occurs during the 802.11 preamble, correlating the received sequence samples with one or more delayed copies of the sequence, with the delay being equivalent to the length of one symbol [16]. We conduct a test with two indoor links, denoted as link A and link B, to study the variability of t_{FD}, and we report the results in Fig. 4. In each test, a L-STA sends constant ping traffic for several hours to a R-STA closer than 15 m. The link distance is fixed, but we observe a high variability of ACK SNR (up to 15 dB) over the day.

[2]This time corresponds to the first 4 symbols in the 802.11b DSSS/CCK preamble and the first 5 short symbols (over the 10 short ones) of the 802.11a/g OFDM preamble.

$t_{MACidle}$ increases with $t_{p,data}$ and $t_{p,ack}$, but a distance of less than 15 m results in a delay of less than $(15/3.4)/44 = 0.1$ μs. The observed $t_{MACidle}$ in Fig. 4 is in the range of $500 - 530$ clocks, i.e., $\approx 11.3 - 12\,\mu s$, higher than $t_{SIFS} + 0.1 = 10.1\,\mu s$. The resulting difference is due to the t_{FD} as expressed in Eq.(1).

A closer look to Fig. 4 shows a bimodal distribution of $t_{MACidle}$, with two main spikes for each link. Link A presents a first spike centered at $\approx 506-507$ clock cycles and a second spike centered at $\approx$ 527 clock cycles. We find a similar distribution in link B. The difference between the two spikes of ≈ 20 clock cycles of $t_{MACidle}$ would cause an unacceptable uncertainty in distance estimation. Referring to Eq.(2), 20 cycles correspond to a difference in the distance estimate of $3.4 \cdot 20 = 68$ m.

This finding can be explained noticing that, in link A, the samples of the second spike are on average received at a lower ACK SNR compared with the ones of the first spike. In contrast, in link B, the second spike is associated with frames at a higher SNR compared with those of the first spike. Since $t_{MACidle}$ in Eq.(1) is a function of t_{FD}, and since t_{SIFS} is expected to be constant, we infer that t_{FD} is a function of the ACK SNR, that is $t_{FD} = f(SNR)$. In particular, the frame detection requires that the ACK preamble is detected and that the dynamic range of the signal is the desired one according to the Automatic Gain Control (AGC) block [17]. The operation for optimizing the dynamic range per frame is a function of the SNR, which causes the spikes in Fig. 4, and L-STA indicates that *the medium is busy at MAC* *layer only after that the signal gain is adjusted.* This method helps to minimize the incidence of false frame detections and of undetected frames, which would reduce the MAC throughput [14]. Next, we clarify how the AGC impacts t_{FD}.

3.2 Impact of gain control

WLAN receivers decode frames over a large range of signal strengths, that may vary from 0 to 70 dB above the background noise level. In each WLAN chipset (and independently of the manufacturer), these signals are internally normalized into a fixed range.

Whenever the incoming signal r is in the *preferred range* (PR), it is already in the ideal range of reception and can be decoded immediately. When instead r is out of the PR range, the gain control of the AGC has to be tuned so that it falls in the desired range. For cost and complexity reasons, the AGC implementation is fairly simple: Two detection mechanisms are possible, which are performed concurrently [14, 17, 18].

A *strong signal detection* (SSD) may occur when r is received at a high strength, that exceed a certain threshold (denoted as coarse high threshold). Upon detection of this condition, the circuit places the RF front-end in a low gain mode and decreases the amplifiers gain to avoid ADC saturation. For example, in PRISM, Intel and Atheros chipsets, the AGC subtracts a coarse 30 dB gain from a strong signal r. This AGC gain is kept constant during the entire frame reception. Once the reception is completed, the AGC block reacts to this event by increasing the amplifier gain to the default value.

If the signal strength of r is too low, the quantization noise of the ADC converter may render the SNR too low for correct detection. Here, a *weak signal detection* (WSD) may be needed to adjust the signal in the preferred range. This process uses statistical analysis based on the correlation of the received 802.11 preamble.

A higher t_{FD} is expected (thus, according to Eq.(2), a higher $t_{MACidle}$ at a given distance d) when the AGC must adjust the dynamic range of r, that is, for SSD and WSD frames. We then introduce the concept of *detection state*. For an ACK frame, the AGC at L-STA operates in one of the following detection states:

- **SSD state**: signal r has high SNR and is detected via strong signal detection.
- **PR state**: signal r is the preferred range.
- **WSD state**: signal r has low SNR and is detected via weak signal detection.

As an example, consider Fig. 4. We expect that the first spike on link A corresponds to frames in the PR state, because no gain control is required, while the second one is for the WSD state, because of the higher time needed to declare a medium busy and the low SNR of ACKs. A different distribution occurs on link B, where ACK frames are received at high SNR, and thus the second spike is due to frames in SSD state.

Table 1: State of frame detection: $t_{MACidle}$ range, SNR range, and $\bar{t}_{FD,s}$ per state

	$t_{MACidle}$(cycles)	SNR(dB)	$\bar{t}_{FD,s}$(cycles)
SSD state	521 − 600	42 − 70	81.1
PR state	500 − 519	15 − 54	63.3
WSD state	521 − 600	0 − 28	84.0

3.3 Revealing the detection state per frame

In order to determine the detection state for all the set of $\{t_{MACidle}, \text{SNR}\}$, we conduct a variety of experiments, placing two stations in locations that differ in terms of distance, SNR, walls/obstacles and 802.11g modulation. For each test, we collect $t_{MACidle}$ and SNR samples as in the tests of Fig. 4. The tests are valid for each 802.11g PHY rate and frame length, because the 802.11g preamble is independent of these parameters. We use the following methodology to infer the state:

- Sample in the PR or the SSD/WSD state: We use $t_{MACidle}$ to distinguish between the two states. From Fig. 4, the standard deviation of the samples in one state (in the order of few cycles) is sufficiently smaller than the difference in cycles between the maximum value of adjacent states (of about 20 cycles). Then, $t_{MACidle}$ samples in PR states do not overlap with the ones in SSD/WSD state.
- Sample in the SSD or the WSD state: We use the ACK SNR to distinguish between the two states. In fact, samples in SSD or WSD state may lie in similar ranges of $t_{MACidle}$, but in different range of ACK SNR. We add an upper bound of 600 cycles, to limit the impact of the small number of outliers in the implementation and frames which are not ACKed due to low signal quality, collisions, etc..

Table 1 allows us to infer the frame detection state per sample. For example, if CAESAR measures $t_{MACidle} = 530$ cycles and $SNR = 45$ dB, it maps the sample to the SSD state.

From Table 1, the observed SNRs of our samples are between 0 and 70 dB, and the SNR ranges for different detection states are partially overlapping. For example, frames with SNR between 15 and 28 dB could be received in either PR or WSD state. In particular, the signal amplitude above the noise is not the only factor that determines what AGC gain is used before the ADC conversion. The SNR is only a measurement of the average signal strength above the noise, but does not give any information about the signal peak-to-average power ratio (which is particularly critical in 802.11g/a OFDM mode) and the presence of interfering signals [19]. As a result, a detection state cannot be unequivocally determined based on the SNR of the ACK, but SNR must be combined with the information about $t_{MACidle}$.

3.4 Detection state and multipath

Once a sample n is mapped to a state, the distance $\hat{d}_{n,s}$ can be expressed as:

$$\hat{d}_{n,s} = c\cdot[(t_{MACidle} - \gamma_s) - t_{SIFS} - \bar{t}_{FD,s}]/2 \quad s \in \mathcal{S}, \tag{3}$$

where $\mathcal{S}$ is the set of detection states, $\bar{t}_{FD,s}$ is the average frame detection time for state s, that depends on the chipset manufacturer implementation, and γ_s is the correction factor for multi-path effect. We also indicate σ_s as the standard deviation of samples in state s. Next we describe how $\bar{t}_{FD,s}$ and γ_s are derived.

3.4.1 Average frame detection time

We measure the set of $\{\bar{t}_{FD,s}\}$ with L-STA and R-STA in line-of-sight (LOS) and known distance using AR9220 Atheros chipsets. For each detection state, we send ping traffic and collect samples of $t_{MACidle}$. From Eq.(2), we then calculate $\bar{t}_{FD,s}$. We summarize the results in Table 1. We measure an average detection time up to 84.0 clock cycles, i.e., $\bar{t}_{FD,s} = 1.84\,\mu$s for the WSD state. Then, $\bar{t}_{FD,s}$ is within an observation time window of 4 μs, as required by the IEEE 802.11 standard. Moreover, since the average value of the PR state can be approximated to the time of two OFDM short symbols (equal to $0.8 \cdot 2 = 1.6\,\mu$s) of the 802.11 preamble, we can conclude that the frame detection in 802.11g mainly works over the first two short symbols, while the AGC block requires an additional time of $\approx 0.4\,\mu$s.

We also measure the standard deviation (not shown in the Table 1) for the three states and find higher values for the SSD and the WSD state ($\sigma_s = 0.83$ and 0.9 clock cycles, respectively) compared with the PR state (a fraction 0.45 of a clock cycle). This extra noise is likely added by the gain control in the AGC block.

3.4.2 Indoor multipath correction factor

As every ranging mechanism, CAESAR is subjected to indoor multipath. Since the multipath error causes a positive bias in the $\{t_{MACidle}\}$ samples, it overestimates the distance, due to the extra distance traveled by the signal with respect to the direct path. Efficiently tackling the multipath requires the ability to discriminate the direct path at lower signal strength (if any) with respect to the reflected paths. However, under multipath effect, a MAC-based solution like CAESAR that relies only on MAC busy and idle times synchronizes on the strongest (reflected) path. While this is an inherent limitation of not using advanced PHY processing, we can alleviate the multipath assuming that it causes a spreading of the $\{t_{MACidle}\}$ samples. In particular, some of the radio signals (e.g. the WLAN frames) arrive to destination via the direct path, others via a reflected path. As a result, the standard deviation σ_s tends to increase, respect to a link with only a direct path.

Let us introduce the multipath threshold t_s. We consider that the direct path is predominant in links where $\sigma_s < t_s$ and no correction factor is needed ($\gamma_s = 0$), and, under multipath, links with $\sigma_s \geq t_s$, where we subtract a correction factor $\gamma_s = \sigma_s/2$. This factor reduces the estimate of a sample that is assumed to be received via a reflected path. We tested small variations of these parameters and observed robustness of the results. This method is not effective in presence of severe multipath effect, i.e., there is only a non-direct path. The result in this case is an overestimation of the distance and a small standard deviation ($\sigma_s < t_s$). Only a PHY layer solution can alleviate severe multipath effects.

3.4.3 Smoothing filter

The dispersion of the measurements can be reduced by filtering multiple samples and exploiting the detection state of the frame. Indeed, samples in different states are subjected to independent sources of noise due to the preamble detection and the gain control adjustment. The noise components can be assumed Gaussian and can then be filtered out via an Exponentially Weighted Moving Average (EWMA) filter:

$$\bar{d}_n = (1 - \alpha_s)\bar{d}_{n-1} + \alpha_s \hat{d}_{n,s}, \tag{4}$$
$$s.t. \quad \bar{d}_1 = \hat{d}_{1,s}$$

where $\bar{d}_n$ is the current distance estimate, $\hat{d}_{n,s}$ is the last measured sample according to Eq.(3), detected in state s, and α_s is the filter weight.

4. IMPLEMENTATION

CAESAR is entirely implemented at L-STA. The implementation is modular: Time-sensitive and chipset-dependent features are coded, respectively, in the `ath` and `ath9k` modules (that implement the low-level software functionality). In order to guarantee real-time estimation, we implement the smoothing filter in the driver. The filter is implemented at the `mac80211` layer (and thus at a higher level of the stack) with general-purpose code.

In order to avoid the introduction of any hardware modules, we devise a method to measure $t_{MACidle}$ via software, accessing what current WLAN hardware can provide to the driver, as explained in the next section.

4.1 Low-level measurements

Our objective is to estimate $t_{MACidle}$ via software. Ideally, we would need to know the timestamps of frame boundaries, e.g., the end of a data transmission. However the WLAN hardware does not need to provide timestamps at the main 44 MHz resolution for 802.11

operations. The idle time can instead be estimated by using the absolute time and the busy time as measured by CSMA/CA operations. We exploit 32-bit register counters maintained by Atheros WLAN chipsets, and updated in hardware at the resolution of a clock tick [20]: 1) `R_RCCNT` counts the time at the 44 MHz built-in quartz resolution. 2) `AR_RCCNT` counts the number of clock ticks where the transceiver is busy at the MAC layer, for transmission or reception.

Let us denote $I = (\texttt{R\_RCCNT} - \texttt{AR\_RCCNT})$ as the entire time in clock units during which the card is idle. From Fig. 3, we can estimate $t_{MACidle}$ at the L-STA by measuring I in two instants of time, i) when the 802.11 data is on the air and the channel is busy (i.e., the data transmission is ongoing), denoted as I_1, and ii) when the ACK is on the air and the channel is busy (i.e., the ACK reception is ongoing), denoted as I_2. As a result:

$$t_{MACidle} = I_2 - I_1 \qquad \text{[clock cycles]}. \tag{5}$$

This is challenging to implement because, as shown in Fig. 2, the state of the channel is also idle before the data transmission and after the ACK reception. For example, if the second reading would occur after that the ACK has been received, $t_{MACidle}$ would be higher than expected, and we would overestimate the distance according to Eq.(3). Further, $t_{MACidle}$ occurs in a very short time and the ACK duration is in the order of tens of μs. Next we describe how CAESAR implements fine-grained detection of the time of an ongoing data transmission and ongoing ACK reception.

4.2 Detecting an ongoing data transmission

We are interested in performing the first reading during the data transmission. For that goal we can use the WLAN hardware interrupts. Note that the interrupts cannot be used to directly and efficiently estimate the TOF in a data-ACK exchange. This is due to unpredictable delay between the time of an IEEE 802.11 hardware event and the driver notification, caused by system tasks and the operating system (OS) power saving state [21, 22].

The WLAN hardware interrupts can instead be used to inform the driver that an event, e.g. the data transmission, started at some time *in the past*. The Atheros baseband releases an interrupt `ATH9K_INT_TX`[3], when the data starts to be transmitted. CAESAR measures I_1 when the interrupt is handled by the driver.

Interrupts cannot be exploited for estimating when the ACK reception is ongoing. Only when the ACK reception is completed, another interrupt is triggered by the hardware and then handled by the driver for upper-layer notification of the successful transmission. However, the delay occurring between the ACK completion and the interrupt handling is sensitive to the CPU usage [21, 22]. Measuring I_2 after this interrupt would erroneously add some (unpredictable) idle time after the busy-to-idle transition at the end of ACK reception and cause errors in the measurements. We describe next how I_2 is measured by CAESAR.

4.3 Detecting an ongoing ACK reception

CAESAR estimates the time δ from the first measurement until ACK reception is ongoing and measures I_2 when this delay elapses. In detail, at the time of measuring I_1, CAESAR reads also the `AR_TFCNT(1)` register counter, that counts the number of clock ticks during which the 802.11 transceiver transmits a frame. `AR_TFCNT(1)` can be reset after each transmission and then used to provide the exact time at L-STA from the start of the transmission until the data transmission has been detected at the driver as explained in section 4.2.

CAESAR uses the busy-waiting `udelay(`δ`)` function of the kernel for scheduling a delay δ as:

$$\delta = t_{Data} - Cr \cdot \texttt{AR\_TFCNT(1)} + t_{SIFS} + t_{Preamble} \; [\mu s], \tag{6}$$

where t_{Data} is the data duration, a function of local PHY information such as the frame length and the rate, Cr is the WLAN clock rate in MHz (44 in 802.11b/g) and $t_{Preamble}$ is the 802.11 preamble length. Since no other time-sensitive software tasks are executed in the WLAN driver between I_1 and I_2 (but only hardware processing), this delay does not introduce any side-effect on the throughput[4]. When the `udelay` function elapses, the ACK reception is ongoing and the driver measures I_2, to estimate $t_{MACidle}$ as in Eq.(5).

4.4 Evaluation of low-level implementation

High accuracy of the values reported by the registers is necessary to guarantee that the readings occur during an ongoing data transmission and ACK reception, respectively. In this section we evaluate whether this goal is obtained by CAESAR. We place two stations in proximity and send ping traffic of 500 bytes from one station for 1000 s. We consider two network loads: 1) one frame per second, which is an ideal condition because of low OS computational load, and 2) flooding ping traffic, which is a case for intensive OS load.

[3]The interrupt `ATH9K_INT_TX` is released for two different events: When i) a data begins to be transmitted and ii) an ACK has been completely received. In order to distinguish between the two events, a status flag `EINPROGRESS` indicates whether `ATH9K_INT_TX` is triggered for the first or second cause. We read the counters when the interrupt `ATH9K_INT_TX` with status flag `EINPROGRESS` is handled by the driver.

[4]We verified experimentally that CAESAR and the original driver obtain very similar TCP throughput of $\approx$ 21 Mb/s via *iperf* tests originated from one Soerkis machine [23] to a second one, with machines at a CPU speed of 500 MHz. The `udelay` function gives the CPU to other processes. Hence the CPU time is not wasted.

4.4.1 *First reading*

The accuracy of the first reading depends of the delay associated with the interrupt `ATH9K_INT_TX`. Our methodology requires that this interrupt is handled before the completion of the data transmission. For that goal, we use the `AR_TFCNT(1)` counter. This value indicates the time passed from the start of data transmission to the time that the interrupt is handled by the driver, i.e., the interrupt delay. Thus, `AR_TFCNT(1)` indicates the set of minimum frame lengths and maximum transmission rates that can be used in CAESAR for ranging calculation.

Results are plotted in Fig. 5 as a function of time. We find an average interrupt delay of $42.23\,\mu$s for the low traffic rate, with a minimum delay of $\approx 35\,\mu$s, and $71.55\,\mu$s for the flooding rate, with standard deviations of $5.86\,\mu$s and $26.78\,\mu$s respectively. These results confirm that the interrupt handler is less predictable for a high system workload. For flooding traffic and $99\,\%$ of the frames, an interrupt delay of at most $\approx 150\,\mu$s occurs from the start of the data transmission until the interrupt is handled by the driver. As a result, only data traffic with transmission time greater than $150\,\mu$s can be reused for ranging calculation with no network overhead. Dedicated traffic must be sent instead when the 802.11 data traffic results in transmission times smaller than $150\,\mu$s.

4.4.2 *Second reading*

The accuracy of the second reading depends on two factors, the reliability of the `udelay` function (which might depend on other CPU processes) and the delay in the driver code to read the registers. For the evaluation, we tune the delay δ to detect an ongoing SIFS at L-STA[5]. In fact, our goal is to analyze the variation of the idle time due to the low-level implementation. Using δ as shown in Eq.(6) would add the variation due to the propagation delay and the ACK detection time. Since we aim to detect the event of an ongoing SIFS, we expect an idle time constantly lower than $10 \cdot 44 = 440$ cycles, where $t_{SIFS} = 10\,\mu$s. This method also allows us to calculate the standard deviation of the event that triggers the I_2 reading.

Idle time samples are plotted in Fig. 5 as a function of time, for both low traffic rate (on top) and flooding rate (on the bottom). For better representation, we show the first 10 s of the test. We find an average idle time of $4.17\,\mu$s for the low traffic rate and $4.68\,\mu$s for the flooding rate, with standard deviations of $0.31\,\mu$s and $0.75\,\mu$s, respectively. This calculation does not consider the samples with idle time greater than $10\,\mu$s. They are not present for the low traffic rate and are only 0.026% of the flooding traffic. The result shows the

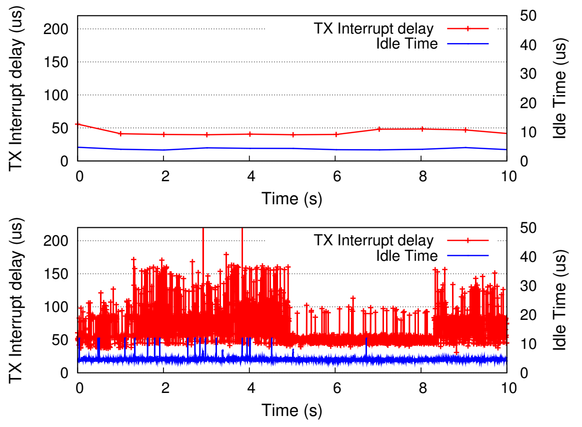

Figure 5: The interrupt that notifies the start of a data transmission must be handled before the end of the data transmission. For flooding traffic, only the traffic with transmission time greater than $150\,\mu$s can be reused by CAESAR. Moreover, the second reading presents a low standard deviation as shown by tuning the delay δ for detecting the event of an ongoing SIFS (that is, detecting idle time smaller than $10\,\mu$s).

very high probability of the event detection and the small standard deviation compared with the duration of the 802.11g ACKs ($\approx$ tens of μs). This permits our methodology to detect the event of an ongoing ACK reception, using δ as in Eq.(6). Hence, the second reading can detect the ongoing ACK reception and retrieve the register counters `R_RCCNT(2)` and `AR_RCCNT(2)` during that time.

4.5 Discussion

In this section, we briefly summarize what is needed to implement CAESAR in today's chipsets.

4.5.1 *Busy and idle time*

Our implementation of CAESAR is based on Atheros chipsets, and their flexibility to provide registers that we exploit for a ranging technique. Other chipset manufacturers already expose similar information. For example, [24] uses the busy and idle time available in the microcode of Intel chipsets.

4.5.2 *Average frame detection time*

The dependency of frame detection time versus the SNR are general insights, because SSD, PR and WSD states are needed in all chipsets. However, the frame detection technique is proprietary and manufacturer dependent. Thus $\{\bar{t}_{FD,s}\}$ has to be calculated for different chipset brands, similar to what we discussed in Section 3. This characterization is feasible because it is required only once per chipset brand. For example, all our AR9220 chipsets have the same performance.

[5]That is, $\delta = t_{Data} - 44 \cdot \texttt{AR\_TFCNT(1)} + K$, where $K < t_{SIFS}$ is a constant factor.

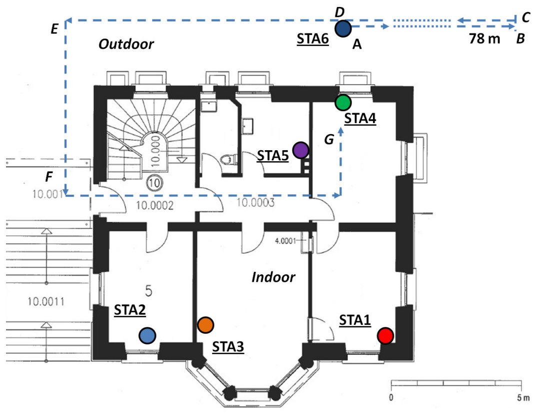

Figure 6: Map of testbed. STA1, STA2, STA3, STA4, STA5 are placed in the locations shown in the map, in different rooms of the testbed. Concrete walls separate the rooms. STA6 is initially placed in the location A and then moved along the path in the map until it reaches position G.

4.5.3 *SIFS time*

The R-STA must add a constant idle time. Some chipsets add a constant offset to the SIFS time, as the *HTC magic* and the *iPad*, that have a constant positive offset of 49.9 cycles in comparison to Atheros chipsets, thus resulting in a $t_{SIFS} = 10 + 49.9/44 = 11.3\ \mu s$. The constant offset is manufacturer dependent, and it might be detected by L-STA based on the first bits of the ACK MAC address (that describe the chipset manufacturer) and a table of SIFS offset per manufacturer.

5. EVALUATION

In this section we evaluate the implementation of CAESAR via several experiments. We show that:

- Average errors of less than 1 m are obtained in 8 links out of 10. A lower accuracy is obtained in links with severe multipath obstruction.
- The error drops below 2 m after fewer than 25 samples in 9 links out of 10.
- It is stable across different frame rates at the L-STA.
- WLAN interference has a limit impact on the estimate, when 3 other stations send a high traffic rate.
- It can track the distance to a WLAN smartphone at pedestrian speeds both outdoors and indoors [25].

5.1 Scenario and deployment

Five stations are at fixed locations according to the map in Fig. 6, and at a relative distances as shown in Table 2. There is no direct LOS between any station pair. These stations (STA1, STA2, STA3, STA4, STA5) are embedded platforms, based on Soekris net5501-70 [23] and they implement CAESAR. We use the AR9220 chipset from Atheros as the WLAN hardware interface, operating in 802.11g mode and with auto-fallback rate activated. In the tests, the filter weight is $\alpha_s = 1/20$, and the multi-path threshold is $t_s = 0.6$ WLAN clock tick in PR state and $t_s = 1$ WLAN clock tick in SSD and WSD state. The sixth station (STA6) is a commercial *HTC magic* smartphone, used to evaluate the tracking capability of CAESAR outdoors and indoors.

Each link is obstructed by walls and obstacles. Rooms are separated by 10 to 20 cm thick concrete walls. Metal furniture is also present. Measurements are subjected to variable channel conditions, also affected by people walking and WLAN interference from other networks. The testbed is controlled via a powerline network to seperate control data from the wireless channel.

Table 2: Indoor evaluation. Link x-y is the link between STAx and STAy. Links are at different signal strengths, from an average of 16.7 dB up to 55.8 dB. Average errors of less than 1 m are obtained in 8 links out of 10.

	Distance			
Link	Expected	Measured	Std	SNR
$1-2$	9 m	8.05 m	0.84 m	38.7 dB
$1-3$	7 m	7.22 m	0.60 m	46.2 dB
$1-4$	9 m	8.29 m	1.20 m	41.9 dB
$1-5$	8.5 m	10.10 m	1.01 m	32.4 dB
$2-3$	2 m	2.57 m	0.71 m	55.8 dB
$2-4$	12 m	15.29 m	2.76 m	16.7 dB
$2-5$	9 m	8.91 m	0.76 m	37.2 dB
$3-4$	9.5 m	10.20 m	1.59 m	29.1 dB
$3-5$	7.5 m	7.24 m	1.41 m	52.0 dB
$4-5$	2 m	1.73 m	1.61 m	53.2 dB

5.2 Indoor evaluation

We measure the distance for each link of the testbed. We indicate link x-y as the link between STAx and STAy. For each link, L-STA sends one unicast ping frame per second (fps) for 400 seconds to an R-STA. Results are summarized in Table 2, calculated over the entire time of measurement. Links have a wide range of signal quality, as measured from the ACK SNR, varying from an average of 16.7 dB up to 55.8 dB. Table 2 shows that average errors of less than 1 m in 8 links out of 10, and the standard deviation (std) is smaller than 1.6 m. A higher error is for the links $1-5$ and $2-4$, likely due to multi-path. Particularly, several obstructions are present in the link $2-4$, resulting in a low average SNR of 16.7 dB.

5.2.1 *Distance error over time*

We plot in Fig. 7 the absolute error distance per link versus the number of samples, for the first 200 samples. The error drops below 2 m after fewer than 25 samples in 9 links out of 10. On link $1-2$, the first sample $\hat{d}_{1,s}$ erroneously overestimates the distance and a few

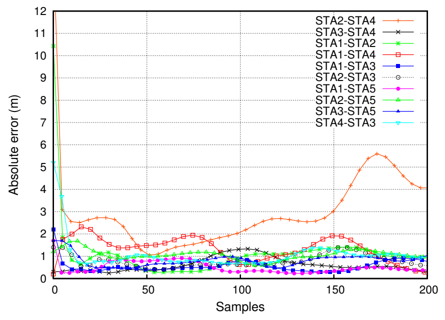

Figure 7: The absolute error drops below 2 m after fewer than 25 samples in 9 links out of 10.

samples are needed before its effect is smoothed out. On link $2-4$ under severe multi-path, there is an error of up to 5.5 m. This link is subjected to variability in the estimate due to channel variations. As shown in Table 2, this causes a higher standard deviation (of 2.76 m) compared with the other links in the testbed.

5.2.2 *Stability*

We perform tests on the link $1-3$ at 7 m to verify the robustness of the implementation with respect to L-STA traffic rate. In the test, we send traffic for 60 s at various frame rate, from 1 up to 1000 fps. Higher frame rates imply higher workload of the L-STA main CPU due to network communication (such as ping packet generation and processing in the kernel), interrupt handling, etc. Particularly, 1000 fps results in a CPU usage of $50-70\%$ on our embedded platforms. The average distance for each test is shown at the top of Fig. 8. There is only a slight impact of the frame rate, less than 0.5 m between 1 fps and 1000 fps. As a consequence CAESAR is stable across different L-STA workloads, and results of Fig. 7 can be mostly considered independent of the traffic rate.

5.2.3 *Impact of WLAN interference*

Ranging traffic must coexist with communication traffic and the estimate distance should not be affected by WLAN interference. Since CAESAR operates on the MAC idle time between a data/ACK communication, WLAN interference does not directly affect the estimate as long as the ACK is successfully received. However, impact on the estimate can still be observed. To demonstrate that, we consider link $1-3$, and we let 3 other stations transmit ping traffic at various rate, from 1 up to 1000 fps, to R-STA, and we let R-STA reply with the ping response. We send 1 fps in link $1-3$, so that it is highly subjected to network interference.

The average distance for each test is shown at the bottom of Fig. 8, as a function of the fps. The estimated distance slightly decreases when the network traffic increases. We observe a difference of ≈ 1 m for 1000 fps with respect to 1 fps, resulting in a difference of $t_{MACidle}$ of $\approx 1/3.4 = 0.29$ WLAN clock tick. A possi-

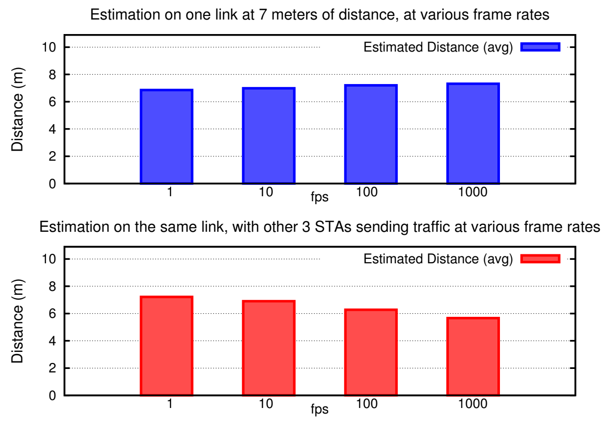

Figure 8: Stability (top) and impact of interference (bottom) as a function of the frame rate.

ble explanations for this result is that the high workload of R-STA causes a drift of its WLAN clock, and thus of t_{SIFS}, contrary to what expected in Section 2.2.2.

5.3 Distance tracking of a WLAN smartphone

We evaluate the scenario of estimating the distance in real time from STA4 to a smartphone (STA6). CAESAR STA4 sends traffic at a low rate of 10 fps to the smartphone. We show the scenario in Fig. 6. In the first part of the test, the smartphone is in an outdoor position. The outdoor link to STA4 has a clear direct path, but reflections are likely present from the neighboring houses, cars on the street and other people walking in the same area. The initial position of the smartphone is 4 m from STA4 (position A), and the mobile user walks away, reaching position B at 78 m of distance from STA4. After being stationary for a few seconds, he walks back to the original position D and then continues through the path shown in Fig. 6 until he goes indoors and reaches the final position G, 1 m away from the AP. The time at which the mobile user reaches positions $\{A, \ldots, G\}$ are calculated by a chronometer App running on the smartphone, while intermediate positions are calculated via linear interpolation.

Fig. 9 shows the distance estimated by CAESAR over time. CAESAR can track the smartphone in real-time. In most locations, the error is a few meters, and we find a higher error only at around 30 s, 120 s (likely due to γ_s, that may overestimate the correction factor due to multipath), and 160 s. Particularly, at $\approx$ 160 s the smartphone is only ≈ 12 m away, but the severe building obstructions cause an 802.11 disassociation of the smartphone from the AP (and no data is sent to it). Fig. 9 also shows the SNR during the test (in dB). Whereas there is a small change in the SNR (with a minimum SNR of 8 dB) between 40 and 100 s, the distance changes rapidly. Moreover, the SNR between 150 and 165 s is *similar* to the SNR reported between 40

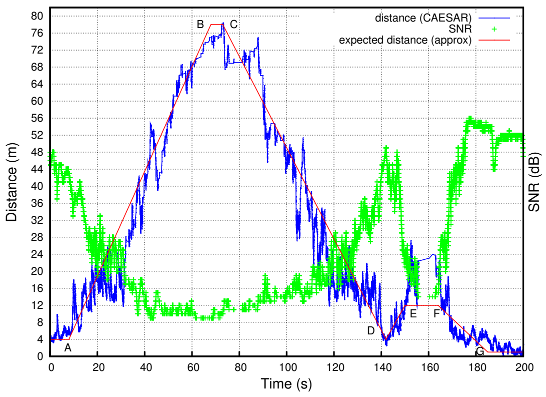

Figure 9: Test with CAESAR AP and *HTC magic* smartphone client (STA6) at a pedestrian speed. The distance is estimated in real-time. The smartphone is in outdoor environment until point F (at ≈ 164 s) is reached and indoor from point F until the end of the test.

and 100 s, while the distance is much smaller, due to a severe variation of the channel path loss. This result indicates that a SNR-only ranging technique would not be effective in this environment. CAESAR shows a high robustness during these path loss variations.

6. RELATED WORK

A survey of related work on estimating the location of a wireless station is provided in [26]. To apply geo-location algorithms, a mobile station has to be in wireless range of a sufficient number of WLAN APs, which is a common case in dense WLAN deployments like urban environments [27]. Next we discuss previous work related to the key features of our solution.

6.1 Time Of Flight

In WLAN ranging techniques, Time-Of-Flight measurements have received little attention. In TOF measurements, a linear relation holds between the propagation time t_p of frames and the distance d, $d = c \cdot t_p$, with c indicated as the speed of light. Multi-path propagation makes the problem non-linear, due to the overestimate as a result of reflected paths. TOF ranging requires either i) synchronized clocks, like in GPS, or ii) an echo technique, that eliminates the need of clock synchronization. Among the works that require synchronized clocks, [28], synchronized the clock for network synchronization. This required dedicated hardware modules at both front-ends and protocol modifications for exchanging frame timestamps. Thus, it cannot estimate the distance to off-the-shelf smartphones, like CAESAR does. To by-pass the lack of clock synchronization, [12] first proposed a two-way echo technique for ranging calculation. Echo techniques measure the round-trip time of a signal transmitted to a remote station. The WLAN data-ACK was then investigated in [9, 10, 21, 22, 29]. Measurements were collected over a long time (hundreds of μs or ms), subjected to variable jitter, required significant post-processing and/or extra PHY implementation. CAESAR minimizes the temporal duration of the measurement, reduces the impact of external sources of noise, and operates on commodity WLAN hardware.

[30] used probe messages as a method to estimate the round-trip time, which added jitters of up to 5 μs and required dedicated chipsets at both local and remote station. [9] calculated the 802.11 TOF based on a low 1 MHz clock. Higher clock resolutions have been investigated in [10, 31]. [10] introduced a dedicated counter module to measure 44 MHz MAC signals. [31] calculated the TOF based on the correlation between the received signal and a Barker signal and required a software-defined radio platform on a monitoring station. CAESAR does not introduce any protocol overhead, and uses the 44 MHz clock without additional hardware.

Hardware interrupts have been exploited in [21, 22]. High-jitter was present in the measurement due to the interrupt handling delay, as a consequence of the power saving state of the machine. CAESAR uses the interrupts in a novel way that reduces the dependency on the main CPU workload. Finally, the kernel timers have also not been used in previous work to detect an ongoing ACK reception.

6.2 Signal Strength

WLAN ranging techniques are mainly based on the SNR of frames from the remote station. There is a non-linear relation between the power level RSS and the distance d: $RSS \propto 10n \log d$, where $\mathrm{RSS} = (\mathrm{SNR} + \mathrm{N})$, N is the noise level and n is the (environment dependent) path loss coefficient. Errors in the estimation are also caused by multi-path and shadowing effects. Theoretical and empirical models are used to translate SNR into a distance. Despite these disadvantages, SNR-based localization is widely used in commercial solutions because they typically only require software changes in off-the-shelf WLAN devices (as long as the SNR per frame is provided at driver level). Rather than using the SNR for inferring the distance, CAESAR uses the SNR to determine the state of the frame detection and hence to infer the time to detect the ACK.

When no ranging measurement is available, signal pattern matching methods are applied. A calibration to the radio environment is performed periodically for determining offline the signal strength signatures at known locations. This negatively impacts the overall maintenance cost [27, 32–37]. These systems have not been practically deployed because it is very time consuming to train a signature-based WLAN positioning system. Over-provisioning of APs and inter-APs measurements are used to reduce the maintenance cost, at the price of higher deployment cost and increased traffic overhead [38]. No offline phase is required by CAESAR.

7. CONCLUSION

The calculation of distances between stations in network coverage is a critical component of a WLAN navigation system. Current solutions do not meet a set of conflicting requirement, such as high precision, fast convergence, and minimal environmental calibration. This limits the potential of WLAN to assist existing navigation systems such as satellite-based systems. We introduced CAESAR, a carrier sense-based ranging technique for measuring the distance between WLAN stations. Our evaluation shows both high accuracy and fast convergence in controlled network conditions and dynamic radio environments. CAESAR operates in real-time and can track the distance to WLAN smartphones at pedestrian speeds. WLAN hardware and communications protocols are not changed. Our method provides an attractive technology for augmenting location-aware applications. As future work, we aim to investigate CAESAR in large scale networks and crowded environments.

8. ACKNOWLEDGEMENT

The very helpful comments from Maria Gorlatova and Vladimir Vukadinovic and the artwork from Maurizio Nitti are appreciated. We thank Peter Steenkiste for insightful and constructive feedbacks.

9. REFERENCES

[1] R. Want, A. Hopper, V. Falcão, and J. Gibbons, "The active badge location system," *ACM Trans. Inf. Syst.*, vol. 10, pp. 91–102, Jan. 1992.
[2] D. Haehnel, W. Burgard, D. Fox, K. Fishkin, and M. Philipose, "Mapping and localization with RFID technology," in *Proc. IEEE ICRA'05*, Apr. 2004.
[3] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proc. ACM MobiCom'00*, Aug. 2000.
[4] S. Gezici, Z. Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu, "Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks," *IEEE Signal Process. Mag.*, vol. 22, pp. 70–84, July 2005.
[5] Skyhook. http://www.skyhookwireless.com/, viewed 31-10-2011.
[6] Google, "Google My Location." http://googlemobile.blogspot.com/2007/11/new-magical-blue-circle-on-your-map.html, viewed 31-10-2011.
[7] Navizon. http://www.navizon.com, viewed 31-10-2011.
[8] Apple, "Apple location services." http://support.apple.com/kb/HT1975, viewed 31-10-2011.
[9] A. Günther and C. Hoene, "Measuring round trip times to determine the distance between WLAN nodes," in *Proc. NETWORKING 2005*, May 2005.
[10] M. Ciurana, F. Barcelo, and F. Izquierdo, "A ranging system with IEEE 802.11 data frames," in *Proc. IEEE RWS'07*, Jan. 2007.
[11] Atheros, "ath9k driver." http://linuxwireless.org/en/users/Drivers/ath9k, viewed 31-10-2011.
[12] D. McCrady, L. Doyle, H. Forstrom, T. Dempsey, and M. Martorana, "Mobile ranging using low-accuracy clocks," *IEEE Trans. Microw. Theory Tec.*, vol. 38, no. 6, 2000.
[13] "IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. (2007 revision)," June 2007.
[14] I. Tinnirello, D. Giustiniano, L. Scalia, and G. Bianchi, "On the side-effects of proprietary solutions for fading and interference mitigation in IEEE 802.11b/g outdoor links," *Comput. Netw.*, vol. 53, pp. 141–152, Feb. 2009.
[15] "Practical manufacturing testing of 802.11 OFDM wireless devices." Available at http://www2.litepoint.com/node/4984, viewed 31-10-2011.
[16] J. Heiskala and J. Terry, *OFDM Wireless LANs: A Theoretical and Practical Guide*. Sams Publishing, 2001.
[17] "Method and system for noise floor calibration and receive signal strength detection." Available at http://www.patentstorm.us/patents/7245893/description.html, viewed 31-10-2011.
[18] R. Gummadi, D. Wetherall, B. Greenstein, and S. Seshan, "Understanding and mitigating the impact of rf interference on 802.11 networks," in *In Proc. of ACM Sigcomm '07*, pp. 385–396, 2007.
[19] "Automatic gain control for a wireless receiver." Available at http://www.freepatentsonline.com/y2006/0222118.html, viewed 31-10-2011.
[20] P. Acharya, A. Sharma, E. M. Belding, K. C. Almeroth, and K. Papagiannaki, "Congestion-aware rate adaptation in wireless networks: A measurement-driven approach," in *Proc. IEEE SECON'08*, June 2008.
[21] M. Ciurana, D. Giustiniano, A. Neira, F. Barcelo-Arroyo, and I. Martin-Escalona, "Performance stability of software TOA-based ranging in WLAN," in *Proc. IPIN'10*, Sept. 2010.
[22] M. Ciurana, D. López, and F. Barceló-Arroyo, "SofTOA: Software ranging for TOA-based positioning of WLAN terminals," in *In Proc. of LoCA '09*, pp. 207–221, Springer-Verlag, 2009.
[23] S. Engineering. http://www.soekris.com, viewed 31-10-2011.
[24] Y. Grunenberger, M. Heusse, F. Rousseau, and A. Duda, "Experience with an implementation of the idle sense wireless access method," in *In Proc. of ACM CoNEXT '07*, pp. 24:1–24:12, 2007.
[25] D. Giustiniano and S. Mangold, "Demo: distance tracking using WLAN time of flight," in *In Proc. of ACM Mobisys '11*, pp. 349–350, 2011.
[26] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 37, pp. 1067–1080, Nov. 2007.
[27] D. Han, D. G. Andersen, M. Kaminsky, K. Papagiannaki, and S. Seshan, "Access point localization using local signal strength gradient," in *Proc. PAM'09*, April 2009.
[28] M. Youssef, A. Youssef, C. Rieger, U. Shankar, and A. Agrawala, "Pinpoint: An asynchronous time-based location determination system," in *In Proc. of ACM MobiSys '06*, pp. 165–176, 2006.
[29] C. Hoene and J. Willmann, "Four-way TOA and software-based trilateration of IEEE 802.11 devices," in *Proc. IEEE PIMRC'08*, Sept. 2008.
[30] S. A. Golden and S. S. Bateman, "Sensor measurements for Wi-Fi location with emphasis on time-of-arrival ranging," *IEEE Trans. Mobile Comput.*, vol. 6, pp. 1185–1198, Oct. 2007.
[31] S. König, M. Schmidt, and C. Hoene, "Precise time of flight measurements in IEEE 802.11 networks by cross-correlating the sampled signal with a continuous Barker code," in *MASS*, pp. 642–649, 2010.
[32] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan, "Indoor localization without the pain," in *In Proc. of ACM MobiCom '10*, pp. 173–184, 2010.
[33] P. Bahl and V. Padmanabhan, "RADAR: an in-building RF-based user location and tracking system," in *Proc. IEEE INFOCOM'00*, Mar. 2000.
[34] Y. Ji, S. Biaz, S. Pandey, and P. Agrawal, "ARIADNE: a dynamic indoor signal map construction and localization system," in *Proc. MobiSys'06*, June 2006.
[35] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Practical robust localization over large-scale 802.11 wireless networks," in *Proc. ACM MobiCom'04*, Sept. 2004.
[36] N. Bulusu, J. Heidemann, and D. Estrin, "GPS-less low cost outdoor localization for very small devices," *IEEE Personal Communications Magazine*, vol. 7, pp. 28–34, October 2000.
[37] M. Youssef and A. Agrawala, "The Horus WLAN location determination system," in *In Proc. of ACM Mobisys '05*, pp. 205–218, 2005.
[38] H. Lim and L. Kung and J. Hou, and H. Luo, "Zero-configuration, robust indoor localization: Theory and experimentation," in *Proc. IEEE INFOCOM'06*, Apr. 2006.