Decimeter-Level Localization with a Single WiFi Access Point

Deepak Vasisht[†], Swarun Kumar[‡], Dina Katabi[†]
[†]MIT CSAIL, [‡] CMU
deepakv@mit.edu, swarun@cmu.edu, dk@mit.edu

Abstract – We present Chronos, a system that enables a single WiFi access point to localize clients to within tens of centimeters. Such a system can bring indoor positioning to homes and small businesses which typically have a single access point.

The key enabler underlying Chronos is a novel algorithm that can compute sub-nanosecond time-of-flight using commodity WiFi cards. By multiplying the time-of-flight with the speed of light, a MIMO access point computes the distance between each of its antennas and the client, hence localizing it. Our implementation on commodity WiFi cards demonstrates that Chronos's accuracy is comparable to state-of-the-art localization systems, which use four or five access points.

1. INTRODUCTION

Recent years have seen significant advances in indoor positioning using wireless signals [48, 28]. State-of-the-art systems have achieved an accuracy of tens of centimeters, even using commodity WiFi chipsets [30, 32, 18]. Existing proposals however target enterprise networks, where multiple WiFi access points can combine their information and cooperate together to locate a user. However, the vast majority of homes and small businesses today have a single WiFi access point. Consequently, this large constituency of wireless networks has been left out of the benefits of accurate indoor positioning.

Developing a technology that can locate users and objects using a single WiFi access point would enable a range of important applications:

(i) *Smart Home Occupancy:* In particular, indoor positioning can play a crucial role in the smart home vision, where WiFi enabled home automation systems like NEST are gaining increasing popularity [37]. Accurate localization addresses a long-standing problem in home automation: reliable occupancy detection [36, 6]. With WiFi-based localization, one can track the number of users per room using their phones or wearables, and accordingly adapt heating and lighting. Knowing the identity of these occupants can then help personalize heating and lighting levels based on user preferences.

(ii) *WiFi Geo-fencing:* Beyond the home, indoor positioning can benefit small businesses that use a single access point to offer free WiFi to attract customers. But with increasingly congested networks, business owners seek to restrict WiFi connectivity to their own customers, given that 32% of users in the US admit to have accessed open WiFi networks outside the premises they serve [47]. Yet securing these networks with passwords is inconvenient, both to customers that connect to these networks and the business owners who must frequently change the passwords. Indoor positioning with a single access point provides a natural solution to this problem because it can automatically authenticate customers based on their location.

(iii) *Device-to-device Location:* More generally, enabling two WiFi nodes to localize each other without additional infrastructure support has implications in areas where WiFi networks may not exist altogether. Imagine traveling with friends or family in countries where WiFi is not as prevalent as in the US, yet still be able to find each other in a mall, museum, or train station, without the need to connect to a WiFi infrastructure.

Our goal is to design a system that enables a single WiFi node (e.g., an access point) to localize another, without support from additional infrastructure. Further, we would like a design that works on commodity WiFi NICs and does not require any additional sensors (cameras, accelerometers, etc.).

As we design for the above goal, it helps to first examine why past systems need multiple access points. The most direct approach to RF-based positioning estimates the time-of-flight (i.e., propagation time) and multiplies it by the speed of light to obtain the distance [23, 16]. However, past proposals for WiFi-based positioning cannot measure the *absolute* time-of-flight. They measure only *differences* in the time-of-flight across the receiver's antennas. Such time differences allow those systems to infer the direction of the source with respect to the receiver, known as the angle of arrival (AoA) [48]. But they don't provide the distance between the source and the receiver. Thus, past work has to intersect the direction of the source from multiple access points to localize it. In fact, past proposals typically use four or five access points to achieve tens of centimeters accuracy [30, 32, 48, 50]. Even the few recent proposals to localize using one WiFi access point [35, 53] require users to walk to multiple locations to emulate the presence of multiple access points. They then intersect signal measurements across these locations coupled with accelerometer readings to infer the user's trajectory.

There are however non-WiFi systems that can accurately measure the absolute time-of-flight, and hence localize using a single receiver. Such systems use special-

ized ultra wideband radios that span multiple GHz [5, 41]. Since time resolution is inversely related to the radio bandwidth, such devices can measure time-of-flight at sub-nanosecond accuracy, and hence localize an object to within tens of centimeters. In contrast, directly measuring time with a 20MHz or 40MHz WiFi radio results in errors of 7 to 15 meters [30].

Motivated by the above analysis, we investigated whether a WiFi radio can emulate a wideband multi-GHz radio, for the purpose of localization. Our investigation led to Chronos, an indoor positioning system that enables a pair of WiFi devices to localize each other. It runs on commodity WiFi cards, and does not require any external sensor (e.g., accelerometer, or camera). Chronos works by making a WiFi card emulate a very wideband radio. In particular, while each WiFi frequency band is only tens of Megahertz wide, there are many such bands that together span a very wide bandwidth. Chronos therefore transmits packets on multiple WiFi bands and stitches their information together to give the illusion of a wideband radio.

Yet, emulating a wideband radio using packets transmitted on different frequency bands is not easy. Stitching measurements across such packets requires Chronos to overcome three challenges:

Resolving Phase Offsets: First, to emulate a wideband radio, Chronos needs to stitch channel state information (CSI) captured by multiple packets, transmitted in different WiFi frequency bands, at different points in time. However, the very act of hopping between WiFi frequency bands introduces a random initial phase offset as the hardware resets to each new frequency (i.e., PLL locking). Chronos must therefore recover time-of-flight to perform positioning despite these random phase offsets.

Eliminating Packet Detection Delay: Second, any measurement of time-of-flight of a packet necessarily includes the delay in detecting its presence. Different packets however experience different random detection delays. To make matters worse, this packet detection delay is typically orders-of-magnitude higher than time-of-flight. For indoor WiFi environments, time-of-flight is just a few nanoseconds, while packet detection delay spans hundreds of nanoseconds [38]. Chronos must tease apart the time-of-flight from this detection delay.

Combating Multipath: Finally, in indoor environments, signals do not experience a single time-of-flight, but a time-of-flight spread. This is because RF signals in indoor environments bounce off walls and furniture, and reach the receiver along multiple paths. As a result, the receiver obtains several copies of the signal, each having experienced a different time-of-flight. To perform accurate localization, Chronos therefore must disentangle the time-of-flight of the direct path from all the remaining paths.

The body of this paper explains how Chronos overcomes these challenges, computes the absolute time-of-flight, and enables localization using a single access point.

Summary of Results: We have implemented Chronos and evaluated its performance on devices equipped with Intel 5300 WiFi cards. Our results reveal the following:

- Chronos computes the time-of-flight with a median error of 0.47 ns in line-of-sight and 0.69 ns in non-line-of-sight settings. This corresponds to a median distance error of 14.1 cm and 20.7 cm respectively.
- Chronos enables a WiFi device (e.g., an AP) to localize another with a median error of 65 cm in line-of-sight and 98 cm in non-line-of-sight settings.

To demonstrate Chronos's capabilities, we use it for three applications:

- *Smart Home Occupancy:* Chronos can be used to track the number of occupants in different rooms of a home using a single access point – a key primitive for smart homes that adapt heating and lighting. Experiments conducted in a 2-bedroom apartment with 4 occupants show that Chronos maps residents in a home to the correct room they are in with an accuracy of 94.3%.
- *WiFi Geo-fencing:* Chronos can be used by small businesses with a single access point to restrict WiFi connectivity to customers within their facility. Experiments in a coffee house reveal that Chronos achieves this to an accuracy of 97%.
- *Personal Drone:* Chronos's ability to locate a pair of user devices can directly benefit the navigation systems of personal robots such as recreational drones. Chronos enables personal drones that can maintain a safe distance from their user by tracking their owner's handheld device. Our experiments using an AscTec Quadrotor reveal that it maintains the required distance relative to a user's device with a root mean-squared error of 4.2 cm.

Contributions: To our knowledge, Chronos is the first system that enables a node with a commercial WiFi card to locate another at tens of centimeters accuracy without any third party support, be it other WiFi nodes or external sensors (e.g., accelerometers). Chronos also contributes the first algorithm for measuring the absolute time-of-flight on commercial WiFi cards at sub-nanosecond accuracy.

2. OVERVIEW

We briefly outline the organization of the rest of this paper. Chronos localizes a pair of WiFi devices without third party support by computing time of flight of signals between them. Sec. §3 describes our approach to compute time-of-flight by stitching together information across multiple WiFi frequency bands. It is followed by a description of the challenges faced by Chronos and how it addresses them. Specifically:

- **Eliminating Packet Detection Delay:** First, Chronos disentangles the time-of-flight from packet detection

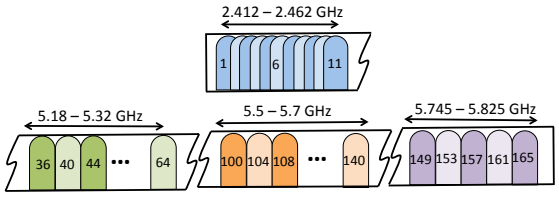


Figure 1: **WiFi Bands:** Depicts WiFi bands at 2.4 GHz and 5 GHz. Note that some of these frequencies (e.g. 5.5-5.7 GHz) are DFS bands in the U.S. that many 802.11h compatible 802.11n radios like Intel 5300 support.

delay, since the latter has no connection to the distance between transmitter and receiver (See Sec. §4).

- **Combating Multipath:** Second, Chronos separates the time-of-flight of the direct path of the wireless signal from that of all the remaining paths (See Sec. §5).
- **Resolving Phase Offsets:** Finally, Chronos removes arbitrary phase offsets that are introduced as the WiFi receiver hops between frequency bands (See Sec. §6).

3. Measuring Time of Flight

In this section, we describe how Chronos measures accurate time-of-flight of signals between a pair of WiFi devices without third party support. For clarity, the rest of this section assumes signals propagate from the transmitter to a receiver along a single path with no detection delay or phase offsets. We address challenges stemming from packet detection delay, multipath and phase offsets in §4, §5 and §6 respectively.

Chronos's approach is based on the following observation: Conceptually, if our receiver had a very wide bandwidth, it could readily measure time-of-flight from a single receiving device at a fine-grained resolution (since time and bandwidth are inversely related). Unfortunately, today's WiFi devices do not have such wide bandwidth. But there is another opportunity: WiFi devices are known to span multiple frequency bands scattered around 2.4 GHz and 5 GHz. Combined, these bands span almost one GHz of bandwidth. By making a transmitter and receiver hop between these different frequency bands, we can gather many different measurements of the wireless channel. We can then "stitch together" these measurements to compute the time-of-flight, as if we had a very wideband radio.

However, our method for stitching time measurements across WiFi frequency bands must account for the fact that many WiFi bands are non-contiguous, unequally spaced, and even multiple GHz apart (Fig. 1). Chronos overcomes these issues by exploiting the relation between the time-of-flight and the phase of wireless channels. Specifically, we know from basic electromagnetics that as a signal propagates in time, it accumulates a corresponding phase depending on its frequency. The higher the frequency of the signal, the faster the phase accumulates. To illustrate, let us consider a transmitter sending a signal to its receiver. Then we can write the wireless channel h as [42]:

$$h = ae^{-j2\pi f\tau}, \tag{1}$$

where a is the signal magnitude, f is the frequency and τ is the time-of-flight. The phase of this channel depends on time-of-flight as:

$$\angle h = -2\pi f\tau \mod 2\pi \tag{2}$$

Notice that the above equation depends directly on the signal's time-of-flight and hence, we can use it to measure the time-of-flight τ as:

$$\tau = -\frac{\angle h}{2\pi f} \mod \frac{1}{f} \tag{3}$$

The above equation gives us the time-of-flight modulo $1/f$. Hence, for a WiFi frequency of 2.4 GHz, we can only obtain the time-of-flight *modulo* 0.4 nanoseconds. Said differently, transmitters with times-of-flight 0.1 ns, 0.5 ns, 0.9 ns, 1.3 ns, etc. all produce identical phase in the wireless channel. In terms of physical distances, this means transmitters at distances separated by multiples of 12 cm (e.g., 3 cm, 15 cm, 27 cm, 39 cm, etc.) all result in the same channel phase. Consequently, there is no way to distinguish between these transmitters using their phase on a single frequency band.

Indeed, this is precisely why Chronos needs to hop between multiple frequency bands $\{f_1,\ldots,f_n\}$ and measure the corresponding wireless channels $\{h_1,\ldots,h_n\}$. The result is a system of equations, one per frequency, that measure the time-of-flight modulo different values:

$$\forall i \in \{1,2,\ldots,n\} \quad \tau = -\frac{\angle h_i}{2\pi f_i} \mod \frac{1}{f_i} \tag{4}$$

Notice that the above set of equations has the form of the well-known Chinese remainder theorem [45]. Such equations can be readily solved using standard modular arithmetic algorithms, even amidst noise [14] and have been used in prior work, in the context of range estimation ([44, 43]).[1] The theorem states that solutions to these equations are unique modulo a much larger quantity – the Least Common Multiple (LCM) of $\{1/f_1,\ldots,1/f_n\}$.

To illustrate how the above system of equations works, consider a source at 0.6 m whose time-of-flight is 2 ns. Say the receiver measures the channel phases from this source on five candidate WiFi frequency bands as shown in Fig. 2. We note that a measurement on each of these channels produces a unique equation for τ, like in Eqn. 4. Each equation has multiple solutions, depicted as colored vertical lines in Fig. 2. However, only the correct solution of τ will satisfy all equations. Hence, by picking the solution satisfying the most number of equations (i.e., the τ with most number of aligned lines in Fig. 2), we can recover the true time-of-flight of 2 ns.

Note that our solution based on the Chinese remainder theorem makes no assumptions on whether the set

[1] Algorithm 1 in §5 provides a more general version of Chronos's algorithm to do this while accounting for noise and multipath

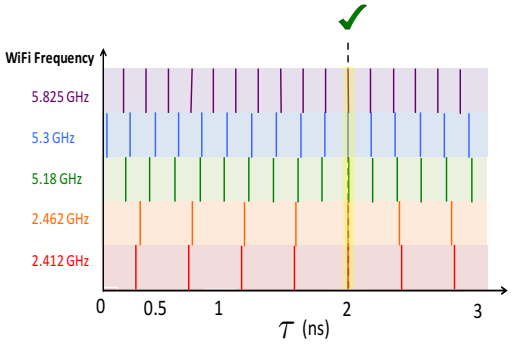


Figure 2: **Measuring Time-of-Flight:** Consider a wireless transmitter at a distance of 0.6 m, i.e. a time-of-flight of 2 ns. The phase of each WiFi channel results in multiple solutions, depicted as colored lines, including 2 ns. However, the solution that satisfies most equations, i.e. has the most number of aligned colored lines is the true time-of-flight (2 ns).

of frequencies $\{f_1, \ldots, f_n\}$ are equally separated or otherwise. In fact, having unequally separated frequencies makes them less likely to share common factors, boosting the LCM. Thus, counter-intuitively, the scattered and unequally-separated bands of WiFi (Fig. 1) are not a challenge, but an opportunity to resolve larger values of τ.

While the above provides a mathematical formulation of our algorithm, we describe below important systems considerations when dealing with commercial WiFi cards:

- Chronos must ensure both the WiFi transmitter and receiver hop synchronously between multiple WiFi frequency bands. Chronos achieves this using a frequency band hopping protocol driven by the transmitter. Before switching frequency bands (every 2-3 ms in our implementation), the transmitter issues a control packet that advertises the frequency of the next band to hop to. The receiver responds with an acknowledgment and switches to the advertised frequency. Once the acknowledgment is received, the transmitter switches frequency bands as well. As a fail-safe, transmitters and receivers revert to a default frequency band if they do not receive packets or acknowledgments from each other for a given time-out duration on any band.
- Our implementation of Chronos sweeps all WiFi bands in 84 ms (12 times per second). This is within the channel coherence time of indoor environments [39] and can empirically localize users at walking speeds (§10.3).
- Finally, we discuss and evaluate the implications of Chronos's protocol on data traffic in §9.3.

4. Eliminating Packet Detection Delay

So far, we computed time-of-flight based on the channels h_i, that signals experience when transmitted over the air on different frequencies f_i. In practice however, there is a difference between the channel over the air, h_i, and the channel as measured by the receiver, $\tilde{h}_i$. Specifically, the *measured* channel at the receiver, $\tilde{h}_i$, experiences a delay in addition to time-of-flight: the delay in detecting the presence of a packet. This delay occurs because WiFi receivers detect the presence of a packet based on the energy of its first few time samples. The number of samples that the receiver needs to cross its energy detection threshold varies based on the power of the received signal, as well as noise. While this variation may seem small, packet detection delays are often an *order-of-magnitude* larger than time-of-flight, particularly in indoor environments, where time-of-flight is just a few tens of nanoseconds (See §9.1). Hence, accounting for packet detection delay is crucial for accurate time-of-flight and distance measurements.

Thus, our goal is to derive the true channel h_i (which incorporates the time-of-flight alone) from the measured channel $\tilde{h}_i$ (which incorporates both time-of-flight and packet detection delay). To do this, we exploit the fact that WiFi uses OFDM. Specifically, the bits of WiFi packets are transmitted in the frequency domain on several small frequency bins called OFDM subcarriers. This means that the wireless channels $\tilde{h}_i$ can be measured on each subcarrier. We then make the following claim:

Claim 4.1. *The measured channel at subcarrier-*0 *does not experience packet detection delay, i.e., it is identical in phase to the true channel at subcarrier* 0.

To see why this claim holds, note that while time-of-flight and packet detection delay appear very similar, they occur at different stages of a signal's lifetime. Specifically, time-of-flight occurs while the signal is transmitted over the air (i.e., in passband). In contrast, packet detection delay stems from energy detection that occurs in digital processing once the carrier frequency has been removed (in baseband). Thus, time-of-flight and packet detection delay affect the wireless OFDM channels in different ways.

To understand this difference, consider the WiFi frequency band, i. Let $\tilde{h}_{i,k}$ be the measured channel of OFDM subcarrier k, at frequency $f_{i,k}$. $\tilde{h}_{i,k}$ experiences two phase rotations in different stages of the signal's lifetime:

- A phase rotation in the air proportional to the over-the-air frequency $f_{i,k}$. From Eqn. 2 in §3, this phase value for a frequency $f_{i,k}$ is:
$$\angle h_{i,k} = -2\pi f_{i,k}\tau \mod 2\pi,$$
where τ is the time-of-flight.
- An additional phase rotation due to packet detection after the removal of the carrier frequency. This additional phase rotation can be expressed as:
$$\Delta_{i,k} = -2\pi(f_{i,k} - f_{i,0})\delta_i,$$
where δ_i is the packet detection delay.

Thus, the total measured channel phase at subcarrier k is:
$$\angle\tilde{h}_{i,k} = (\angle h_{i,k} + \Delta_{i,k}) \mod 2\pi \quad (5)$$
$$= (-2\pi f_{i,k}\tau - 2\pi(f_{i,k} - f_{i,0})\delta_i) \mod 2\pi \quad (6)$$
Notice from the above equation that the second term $\Delta_{i,k} = -2\pi(f_{i,k} - f_{i,0})\delta_i = 0$ at $k = 0$. In other words, at

the zero-subcarrier of OFDM, the measured channel $\tilde{h}_{i,k}$ is identical in phase to the true channel $h_{i,k}$ over-the-air which validates our claim.

In practice, this means that we can apply the Chinese Remainder theorem as described in Eqn. 4 of §3 at the zero-subcarriers (i.e. center frequencies) of each WiFi frequency band. In the U.S., WiFi at 2.4 GHz and 5 GHz has a total of 35 WiFi bands with independent center frequencies.[2] Therefore, a sweep of all WiFi frequency bands results in 35 independent equations like in Eqn. 4, which we can solve to recover time-of-flight.

One problem still needs to be addressed. So far we have used the measured channel at the zero-subcarrier of WiFi bands. However, WiFi transmitters do not send data on the zero-subcarrier, meaning that this channel simply cannot be measured. This is because the zero-subcarrier overlaps with DC offsets in hardware that are extremely difficult to remove [22, 3]. So how can one measure channels on zero-subcarriers if they do not even contain data?

Fortunately, Chronos can tackle this challenge by using the remaining WiFi OFDM subcarriers, where signals are transmitted. Specifically, it leverages the fact that indoor wireless channels are based on physical phenomena. Hence, they are continuous over a small number of OFDM subcarriers [27]. This means that Chronos can interpolate the measured channel phase across all subcarriers to estimate the missing phase at the zero-subcarrier.[3] Indeed, the 802.11n standard [3] measures wireless channels on as many as 30 subcarriers in each WiFi band. Hence, interpolating between the channels not only helps Chronos retrieve the measured channel on the zero-subcarrier, but also provides additional resilience to noise.

To summarize, Chronos applies the following steps to account for packet detection delay: (1) It obtains the measured wireless channels on the 30 subcarriers on the 35 available WiFi bands; (2) It interpolates between these subcarriers to obtain the measured channel phase on the zero-subcarriers on each of these bands, which is unaffected by packet detection delay. (3) It retrieves the time-of-flight using the resulting 35 channels.

5. Combating Multipath

So far, our discussion has assumed that a wireless signal propagates along a single direct path between its transmitter and receiver. However, indoor environments are rich in multipath, causing wireless signals to bounce off objects in the environment like walls and furniture. Fig. 3(a) illustrates an example where the signal travels along three paths from its sender to receiver. The signals on each of these paths propagate over the air incurring different time delays as well as different attenuations. The ultimate received signal is therefore the sum of these multiple signal copies, each having experienced a different propagation delay. Fig. 3(b) represents this using a *multipath profile*. This profile has peaks at the propagation delays of signal paths, scaled by their respective attenuations. Hence, Chronos needs a mechanism to find such a multipath profile, so as to separate the propagation delays of different signal paths. This allows it to then identify the time-of-flight as the least of these propagation delays, i.e. the delay of the most direct (shortest) path.

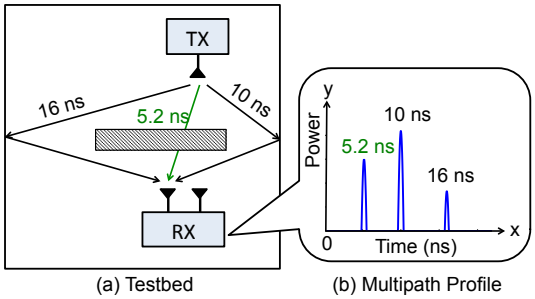


Figure 3: **Combating Multipath:** Consider a signal propagating from a transmitter to a receiver along 3 paths as shown in (a): an attenuated direct path and two reflected paths of lengths 5.2 ns, 10 ns and 16 ns respectively. These paths can be separated by using the inverse discrete Fourier Transform as shown in (b). The plot has 3 peaks corresponding to the propagation delays of the paths, with peak magnitudes scaled by relative attenuations.

5.1 Computing Multipath Profiles

Say that wireless signals from a transmitter reach a receiver along p different paths. The received signal from each path corresponds to amplitudes $\{a_1, \ldots, a_p\}$ and propagation delays $\{\tau_1, \ldots, \tau_p\}$. Observe that Eqn. 1 considers only a single path experiencing propagation delay and attenuation. In the presence of multipath, we can extend this equation to write the measured channel $\tilde{h}_{i,0}$ on center-frequency $f_{i,0}$ as the sum of the channels on each of these paths, i.e.:

$$\tilde{h}_{i,0} = \sum_{k=1}^{p} a_k e^{-j2\pi f_{i,0}\tau_k} \quad \text{, for } i = 1, \ldots, n \qquad (7)$$

Now, we need to disentangle these different paths and recover their propagation delays. To do this, notice that the above equation has a familiar form – it is the well-known Discrete Fourier Transform. Thus, if one could obtain the channel measurements at many uniformly-spaced frequencies, a simple inverse-Fourier transform would separate individual paths. Such an inverse Fourier transform has a closed-form expression that can be used to obtain the propagation delay of all paths and compute the multipath profile (up to a resolution defined by the bandwidth).

WiFi frequency bands, however, are not equally spaced – they are scattered around 2.4 GHz and multiple non-contiguous chunks at 5 GHz, as shown in Fig. 1. While we

[2]Including the DFS bands at 5 GHz in the U.S. which are supported by many 802.11h-compatible 802.11n radios, e.g., the Intel 5300.

[3]Our implementation of Chronos uses cubic spline interpolation.

can measure $\tilde{h}_{i,0}$ at each WiFi band, these measurements will not be at equally spaced frequencies and hence cannot be simply used to compute the inverse Fourier transform. In fact, since our measurements of the channels are not uniformly spaced, we are dealing with the *Non-uniform* Discrete Fourier Transform or NDFT [8]. To recover the multipath profile, we need to invert the NDFT.

5.2 Inverting the NDFT

The NDFT is an under-determined system, where the responses of multiple frequency elements are unavailable [19, 15]. Thus, the inverse of such a Fourier transform does not have a single closed-form solution, but several possible solutions. So how can Chronos pick the best among those solutions to find the true times-of-flight?

Chronos adds another constraint to the inverse-NDFT optimization. Specifically, this constraint favors solutions that are sparse, i.e., have few dominant paths. Intuitively, this stems from the fact that while signals in indoor environments traverse several paths, a few paths tend to dominate as they suffer minimal attenuation [10].[4] Indeed other localization systems make this assumption as well, albeit less explicitly. For instance, antenna-array systems can resolve a limited number of dominant paths based on the number of antennas they use.

We can formulate the sparsity constraint mathematically as follows. Let the vector $\mathbf{p}$ sample inverse-NDFT at m discrete values $\tau \in \{\tau_1, \ldots, \tau_m\}$. Then, we can introduce sparsity as a simple constraint in the NDFT inversion problem that minimizes the L-1 norm of $\mathbf{p}$. Indeed, it has been well-studied in optimization theory that minimizing the L-1 norm of a vector favors sparse solutions for that vector [7]. Thus, we can write the optimization problem to solve for the inverse-NDFT as:

$$\min \|\mathbf{p}\|_1 \tag{8}$$

$$\text{s.t.} \quad \|\tilde{\mathbf{h}} - \mathcal{F}\mathbf{p}\|_2^2 = 0 \tag{9}$$

where, $\mathcal{F}$ is the $n \times m$ Fourier matrix, i.e. $\mathcal{F}_{i,k} = e^{-j2\pi f_{i,0}\tau_k}$, $\tilde{\mathbf{h}} = [\tilde{h}_{1,0}, \ldots, \tilde{h}_{n,0}]^T$ is the $n \times 1$ vector of wireless channels at the n different center-frequencies $\{f_{1,0}, \ldots, f_{n,0}\}$, $\|\cdot\|_1$ is the L-1 norm, and $\|\cdot\|_2$ is the L-2 norm. Here, the constraint makes sure that the Discrete Fourier Transform of $\mathbf{p}$ is $\tilde{\mathbf{h}}$, as desired. In other words, it ensures $\mathbf{p}$ is a candidate inverse-NDFT solution of $\tilde{\mathbf{h}}$. The objective function favors sparse solutions by minimizing the L-1 norm of $\mathbf{p}$.

We can re-formulate the above optimization problem using the method of Lagrange multipliers as:

$$\min_{\mathbf{p}} \|\tilde{\mathbf{h}} - \mathcal{F}\mathbf{p}\|_2^2 + \alpha\|\mathbf{p}\|_1 \tag{10}$$

Notice that the factor α is a sparsity parameter that enforces the level of sparsity. A bigger choice of α leads to fewer non-zero values in $\mathbf{p}$.

This objective function is convex but not differentiable.

1 Algorithm to Compute Inverse NDFT

```
▷ Given: Measured Channels, h̃
▷ F: Non-uniform DFT matrix, such that F_{i,k} = e^{-j2πf_{i,0}τ_k}
▷ α: Sparsity parameter; ε: Convergence Parameter
▷ Output: Inverse-NDFT, p
▷ Initialize p_0 to a random value, t = 0, γ = 1/||F||_2.
while converged = false do
    p_{t+1} =SPARSIFY(p_t − γF*(Fp_t − h̃), γα)
    if ||p_{t+1} − p_t||_2 < ε then
        converged = true
        p = p_{t+1}
    else
        t = t + 1
    end if
end while
function SPARSIFY(p,t)
    for i = 1, 2, ...length(p) do
        if |p_i| < t then
            p_i = 0
        else
            p_i = p_i (|p_i| − t)/|p_i|
        end if
    end for
end function
```

Our approach to optimize for it borrows from proximal gradient methods, a special class of optimization algorithms that have provable convergence guarantees [24]. Specifically, our algorithm takes as inputs the measured wireless channels $\tilde{\mathbf{h}}$ at the frequencies $\{f_{1,0}, \ldots, f_{n,0}\}$ and the sparsity parameter α. It then applies a gradient-descent style algorithm by computing the gradient of differentiable terms in the objective function (i.e., the L-2 norm), picking sparse solutions along the way (i.e., enforcing the L-1 norm). Algorithm 1 summarizes the steps to invert the NDFT and find the multipath profile.[5]

Inverting the NDFT provides Chronos with the time-of-flight on all paths. Chronos still needs to identify the direct path to compute the distance between transmitter and receiver. To do this, Chronos leverages that: of all the paths of the wireless signal, the direct path is the shortest. Hence, the time-of-flight of the direct path is the time corresponding to the first peak in the multipath profile.

It is worth noting that by making the sparsity assumption, we lose the propagation delays of extremely weak paths in the multipath profile. However, Chronos only needs the propagation delay of the direct path. As long as this path is among the dominant signal paths, Chronos can retrieve it accurately. Of course, in some unlikely scenarios, the direct path may be too attenuated, which leads to poorer localization in that instance. Our results in §9.1 depict the sparsity of representative multipath profiles, and show its impact on overall accuracy.

6. CORRECTING FOR PHASE OFFSETS

To work with practical WiFi radios, Chronos has to ad-

[4] We empirically evaluate the sparsity of indoor multipath profiles in typical line-of-sight and non-line-of-sight settings in §9.1.

[5] MATLAB implementation of this algorithm takes 3.1 *s* (standard deviation 0.6 *s*) for Chronos's implementation in Sec. 8.

dress their inherent phase and frequency offsets:

- **PLL Phase Offset:** Frequency hopping causes a random phase offset in the measured channel. This is because the phase-locked loop (PLL) responsible for generating the center frequency for the transmitter and the receiver starts at random initial phase (say, $\phi^{tx}_{i,0}$ and $\phi^{rx}_{i,0}$ respectively). As a result, the channel measured at the receiver is corrupted by an additional phase offset $\phi^{tx}_{i,0} - \phi^{rx}_{i,0}$. This phase offset, if left uncorrected, could render the phase information uncorrelated with the time-of-flight of the signal.
- **Carrier Frequency Offset:** This offset occurs due to small differences in the carrier frequency of the transmitting and receiving radio. This leads to a time varying phase offset across each frequency band. Such differences accumulate quickly over time and need to be corrected for every WiFi packet. Mathematically, in the i^{th} WiFi frequency band, the receiver center frequency $f^{rx}_{i,0}$ is slightly different from the transmitter center frequency, $f^{tx}_{i,0}$. As a result, the channel measurements at the receiver have an additional phase change which is proportional to $f^{tx}_{i,0} - f^{rx}_{i,0}$.

Let us refer to the channel values that incorporate phase and frequency offsets as CSI (channel state information), which is the typical term use in communication systems. Then, the CSI measured at the receiver for the i^{th} frequency band can be written as:

$$\text{CSI}^{rx}_{i,0}(t) = \tilde{h}_{i,0} e^{j(f^{tx}_{i,0} - f^{rx}_{i,0})t + j(\phi^{tx}_{i,0} - \phi^{rx}_{i,0})} \quad (11)$$

So how do we remove the phase and frequency offsets from CSI? To address this issue, Chronos exploits that, the phase and frequency offsets measured on one node with respect to another change sign when measured on the second node with respect to the first. Thus, if one would measure the CSI on the transmitter with respect to the receiver, it would take the following value:

$$\text{CSI}^{tx}_{i,0}(t) = \tilde{h}_{i,0} e^{j(f^{rx}_{i,0} - f^{tx}_{i,0})t + j(\phi^{rx}_{i,0} - \phi^{tx}_{i,0})}. \quad (12)$$

Note that the channel, $\tilde{h}_{i,0}$, in equations 11 and 12 is the same due to reciprocity [20]. We can therefore multiply the CSI measurements at the receiver and the transmitter to recover the wireless channel as follows:

$$\tilde{h}^2_{i,0} = \text{CSI}^{rx}_{i,0}(t)\text{CSI}^{tx}_{i,0}(t) \quad (13)$$

One may wonder how Chronos measure the CSI at the transmitter. Note however that as part of our channel hopping protocol both nodes have to transmit packets to each other. Hence, the CSI can be measured on both sides and exchanged to apply Eqn. 13.

The above formulation helps us only retrieve the square of the wireless channels $\tilde{h}^2_{i,0}$. However, this is not an issue: Chronos can directly feed $\tilde{h}^2_{i,0}$ into its algorithm (Alg. 1 in §5) instead of $\tilde{h}_{i,0}$. Then the first peak of the resulting multipath profile will simply be at twice the time-of-flight.

To see why, let us look at a simple example. Consider a transmitter and receiver obtaining their signals along two paths, with propagation delays 2 ns and 4 ns. We can write the square of the resulting wireless channels from Eqn. 7 for frequency band i in a simple form:

$$\begin{aligned}\tilde{h}^2_{i,0} &= (a_1 e^{-j2\pi f_{i,0}\times 2} + a_2 e^{-j2\pi f_{i,0}\times 4})^2 \\ &= a_1^2 e^{-j2\pi f_{i,0}\times 2\times 2} + 2a_1a_2 e^{-j2\pi f_{i,0}\times (2+4)} + a_2^2 e^{-j2\pi f_{i,0}\times 4\times 2} \\ &= b_1 e^{-j2\pi f_{i,0}\times 4} + b_2 e^{-j2\pi f_{i,0}\times 6} + b_3 e^{-j2\pi f_{i,0}\times 8}\end{aligned}$$

Where $b_1 = a_1^2$, $b_2 = 2a_1a_2$, $b_3 = a_2^2$. Clearly, the above equation has a form similar to a wireless channel with propagation delays 4 ns, 6 ns and 8 ns respectively. This means that applying Chronos's algorithm will result in peaks precisely at 4 ns, 6 ns and 8 ns. Notice that in addition to 4 ns and 8 ns that are simply twice the propagation delays of genuine paths, there is an extra peak at 6 ns. This peak stems from the square operation in $\tilde{h}^2_{i,0}$ and is a sum of two delays. However, the sum of any two delays will always be higher than twice the lowest delay. Consequently, the smallest of these propagation delays is still at 4 ns – i.e., at twice the time-of-flight. A similar argument holds for larger number of signal paths, and can be used to recover time-of-flight.

Finally, we make a few observations: (1) In practice, the forward and reverse channels cannot be measured at exactly the same t but within short time separations (tens of microseconds), resulting in a small phase error. However, this error is significantly smaller than the error from not compensating for frequency offsets altogether (for tens of milliseconds). The error can be resolved by averaging over several packets. (2) Delays in the hardware result in a constant additive value to the time-of-flight. This constant can be pre-calibrated once in the lifetime of a WiFi-card, by measuring time-of-flight to a device at a known distance. (3) Standard Fourier Transform properties dictate that a minimum separation of Δf in frequencies of measured CSI values, leads to an ambiguity by multiples of $\frac{1}{\Delta f}$ in the time estimates (i.e the delay is measured modulo $\frac{1}{\Delta f}$). Since, Chronos uses CSI measurements at center frequencies, the minimum frequency separation is 5 *MHz* [6]. Hence, the time domain ambiguity is 200 *ns* which corresponds to a distance of 60 m, i.e., distance measurements are modulo 60 *m*. Thus, for indoor settings and typical WiFi propagation, one can ignore the modulo factor.

7. Computing Distances and Location

So far, we have explained how Chronos measures the time-of-flight between two antennas on a pair of WiFi cards. One can then compute the distance between the two antennas (i.e., the two devices) by multiplying the time-of-flight by the speed of light.

In order to get the location of the client from the distance measurements, Chronos follows a two-step proce-

[6]The frequency separation is less than the channel bandwidth of 20 MHz due to overlapping WiFi bands.

dure. In the first step, Chronos refines the distance measurements by utilizing geometric constraints, imposed by the relative locations of the antennas on the access point and the client. In the second step, Chronos formulates a quadratic optimization problem, based on the refined distances to get the accurate location of the client with respect to the access point.

Mathematically, we denote the separation between antenna i and antenna j on the access point by l_{ij}^{ap}. Similarly, antenna i and antenna j on the client are separated by $l_{i,j}^{cl}$. By using standard triangle inequality, we know that $|d_{ij} - d_{i'j}| < l_{ii'}^{ap}$, where d_{ij} is the distance measured by Chronos between antenna i on the access point and antenna j on the client. When a pair of distances measured by Chronos violates this constraint; clearly, one or both of the distance measurements must be declared invalid. Chronos uses a relaxed version of triangle inequality to eliminate erroneous distance measurements. Specifically, if we denote the maximum distance between any pair of antennas on a device by α, Chronos chooses the largest cluster, C, of distance measurements such that each measurement in this cluster is at most α away from at least one other distance measurement in the cluster. Chronos, then, discards the distance measurements that do not belong to C.

Finally, Chronos formulates the following constrained optimization problem to find the accurate position of the client. We denote the position of the i^{th} antenna on the access point by (x_i^{ap}, y_i^{ap}). Our goal is to optimize for the position of the client which we denote by (x, y), where x and y are 3×1 vectors of antenna coordinates:

$$\min_{\epsilon>0,x,y} \epsilon$$

such that

$$\forall (i,j) \in C, |dist((x_i^{ap}, y_i^{ap}), (x_j, y_j)) - d_{ij)}| < \epsilon$$

$$\forall (i,j) \in \{1,2,3\}, dist((x_i, y_i), (x_j, y_j)) = l_{i,j}^{cl}$$

where $dist((x_1, y_1), (x_2, y_2))$ denotes the euclidean distance between points (x_1, y_1) and (x_2, y_2). On a high level, Chronos optimizes for the minimal violation of the distance constraints while still maintaining the relative position of the antennas on the client. We formulate this problem as a quadratic-constrained optimization in MATLAB and use the fmincon solver to find the optimum solution. The average execution time for this algorithm is 0.09 s (standard deviation 0.01 s).

8. IMPLEMENTATION

We implemented Chronos as a software patch to the iwlwifi driver on Ubuntu Linux running the 3.5.7 kernel. To measure channel-state-information, we use the 802.11 CSI Tool [21] for the Intel 5300 WiFi card. We measure the channels on both 2.4 GHz and 5 GHz WiFi bands.[7]

[7]The Intel 5300 WiFi card is known to have a firmware issue on the 2.4 GHz bands that causes it to report the phase of the channel $\angle \tilde{h}_{i,0}$ modulo $\pi/2$ (instead of the phase modulo 2π) [18]. We resolve this issue by performing Chronos's algorithm at 2.4 GHz on $\tilde{h}_{i,0}^4$ instead of $\tilde{h}_{i,0}$. This does not affect the fact that the direct path of the signal will continue being the first peak in the inverse NDFT (like in §6).

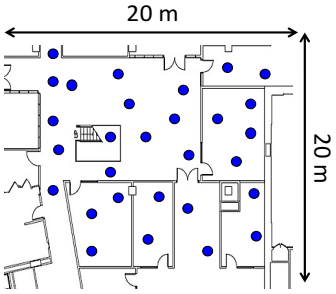


Figure 4: **Lab Testbed:** The figure depicts our testbed with candidate locations for the nodes marked with blue dots.

Unless specified otherwise, we pair two Chronos devices by placing each device in monitor mode with packet injection support on the same WiFi frequency. We implemented Chronos's frequency band hopping protocol (see §3) in the iwlwifi driver using high resolution timers (hrtimers), which can schedule kernel tasks such as packet transmits at microsecond granularity. Since the 802.11 CSI Tool does not report channel state information for Link-Layer ACKs received by the card, we use packet-injection to create and transmit special acknowledgments directly from the iwlwifi driver to minimize delay between packets and acknowledgments. These acknowledgments are also used to signal the next channel that the devices should hop to, as described in §3. We process the CSI to infer time-of-flight and device locations purely in software written in part in C++, MEX and MATLAB.

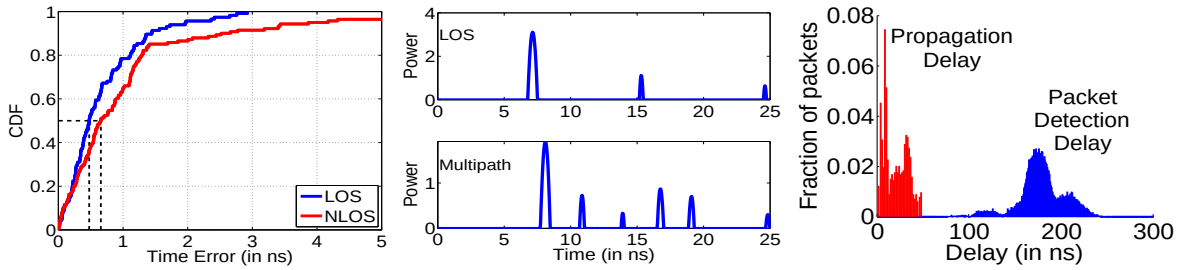
We note that all our experiments are conducted in naturally dynamic environments, specifically, an office building, a coffee shop and a home with four occupants. Chronos requires no modifications based on the changes in the environment. The environments have ambient WiFi traffic. We could sense 3 to 19 different access points across our testbeds. Chronos disables the contention mechanism during hopping in order to enable fast switching across different WiFi bands. This causes noise in Chronos's measurements when there is a collision with other WiFi packets. However, Chronos is resilient to noise on a small subset of the measurements. Moreover, since Chronos sends few packets on each WiFi band, it does not adversely effect the WiFi traffic.

9. RESULTS

We evaluate Chronos's ability to measure the time-of-flight, and compute a client's position using a single AP.

9.1 Time-of-Flight Accuracy

We examine whether Chronos can deliver on its promise



(a) Time of Flight (b) Multipath Profiles (c) Packet Detection Delay

Figure 5: **Accuracy in Time of Flight**: (a) The CDF of error in time-of-flight between two devices in Line of Sight (LOS) and Non-Line of Sight (NLOS). (b) Representative multipath profiles. (c) Histograms of time-of-flight and packet detection delay.

of measuring sub-nanosecond time-of-flight between a pair of commodity WiFi devices.

Method: We run our experiments in the testbed in Fig. 4. In each experiment, we randomly pick a location for the AP. We then randomly pick a client location that is within 15 meter from the AP. We experiment with both line-of-sight and non-line-of-sight settings. We perform our experiments using a 10" ASUS EEPC netbook as a client and a Thinkpad W300 Laptop emulating a WiFi AP via hostapd. Both devices are equipped with the 3-antenna Intel 5300 chipset. The antennas are placed at the corner of each device, which results an average antenna spacing of 30cm for the Thinkpad AP and 12cm for the ASUS client.

Using the above setup, we have run 400 localization experiments for different AP-client pairs. For each pair, we run Chronos channel hopping protocol. We compute the time of flight between each transmit antenna and receive antenna. We measure the ground-truth location using a combination of architectural drawings of our building and a Bosch GLM50 laser distance measurement tool [1], which measures distances up to 50 m with an accuracy of 1.5 mm. The ground truth time-of-flight is the ground truth distance divided by the speed of light.

Time-of-Flight Results: We first evaluate Chronos's accuracy in time-of-flight. Fig. 5(a) depicts the CDF of the time-of-flight of the signal in line-of-sight settings and non-line-of-sight. We observe that the median errors in time-of-flight estimation are 0.47 ns and 0.69 ns respectively. These results show that Chronos achieves its promise of computing time-of-flight at sub-nanosecond accuracy. To put this in perspective, consider SourceSync [38], a state-of-the-art system for time synchronization. SourceSync achieves 95th percentile synchronization error up to 20 ns, using advanced software radios. In contrast, the figure shows that Chronos's 95th percentile error is 1.96 ns in line-of-sight and 4.01 ns in non-line-of-sight. Thus, Chronos achieves 5 to 10 fold lower error in time-of-flight, and runs on commodity WiFi cards as opposed to software radios.

Multipath Profile Results: Next, we would like to examine whether multiple path profiles are indeed sparse. Thus, we plot candidate multipath profiles computed by Chronos in the above experiments. Fig. 5(b) plots representative multipath profiles in line-of-sight and multipath environments. We note that both profiles are sparse, with the profile in multipath environments having five dominant peaks. Across all experiments, the mean number of dominant peaks in the multipath profiles is 5.05 on average, with standard deviation 1.95 — indicating that they are indeed sparse. As expected, the profile in line-of-sight has even fewer dominant peaks than the profile in multipath settings. In both cases, we observe that the leftmost peaks in both profiles correspond to the true location of the source. Further, we observe that the peaks in both profiles are sharp due to two reasons: 1) Chronos effectively spans a large bandwidth that includes all WiFi frequency bands, leading to high time resolution; 2) Chronos's resolution is further improved by exploiting sparsity that focuses on retrieving the sparse dominant peaks at much higher resolution, as opposed to all peaks.

Packet Detection Delay Results: Past work on WiFi time measurement and/or synchronization cannot measure the time-of-flight of a packet separately from its detection delay [38]. ([35] measures the distribution of detection delays but not the detection delay of a particular packet.) In contrast, Chronos has a novel way for separating the detection delay from the time-of-flight. We would like to understand the importance of this capability for the success of Chronos. Thus, we use the measurements from the above experiments to compare time-of-flight in indoor environments against packet detection delay.

Fig. 5(c) depicts histograms of both packet detection delay and time-of-flight across experiments. Chronos observes a median packet detection delay of 177 ns across experiments. We emphasize two key observations: (1) Packet detection delay is nearly 8$\times$ larger than the time-of-flight in our typical indoor testbed. (2) Packet delay varies dramatically between packets, and has a high standard deviation of 24.8 ns. In other words, packet detection delays are large, highly variable, and hard to predict. This means that if left uncompensated, these delays could lead to a large error in time-of-flight measurements. Our results therefore reinforce the importance of accounting for these delays and demonstrate Chronos's ability to do so.

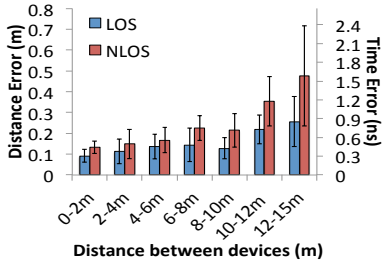


Figure 6: **Ranging Accuracy:** Plots error in distance across the true distance separating the transmitter from the receiver.

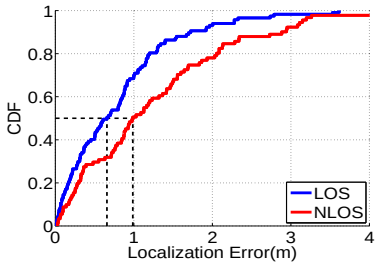


Figure 7: **Localization Accuracy:** Plots CDF of localization error in Line-of-Sight (LOS) and Non-Line-of-Sight (NLOS).

9.2 Localization Accuracy

We evaluate Chronos's accuracy in measuring distance and location using a single access point.

Method: We compute the time-of-flight between the AP and user client in the testbed as described in §9.1 above. We use the measured time-of-flight to compute the distance between antennas and localize the client with respect to the AP as described in §7. We repeat the experiment multiple times in line-of-sight and non-line-of-sight.

Location Results: Fig. 7 plots a CDF of localization error using Chronos in different settings. The device's median positioning error is 65 cm and 98 cm in line-of-sight and non-line-of-sight respectively. This result shows that Chronos's accuracy is comparable to state-of-the-art indoor localization that use multiple AP's [30, 32, 48].

Ranging Results: In some applications, it is important to maintain a particular distance between objects but the exact location is not necessary (e.g., preventing robot collision). Thus, here we plot the ranging results of Chronos. Fig. 6 plots the median and standard deviation of error in distance computed between the transmitter and receiver against their true distance. We observe that this error is initially around 10 cm and increases to at most 26 cm at 12-15 meters. The increase is primarily due to reduced signal-to-noise ratio at further distances. Note that the ranging accuracy is higher than the localization accuracy because ranging is a simpler problem (no need to find the exact direction) and Chronos's time-of-flight computation naturally yields the range between devices.

9.3 Impact on Network traffic

Chronos enables localization between a pair of WiFi devices without third party support. In many cases, these are user devices that do not otherwise communicate data between each other directly. However, an interesting question is the impact of Chronos on network traffic, if one of the devices is serving traffic, such as a WiFi AP. This experiment answers three questions in this regard: (1) How long does Chronos take to hop between all WiFi bands? (2) How does Chronos impact real-time traffic like video streaming applications? (3) How does Chronos affect TCP? We address these questions below:

Method: We consider a Thinkpad W530 Laptop emulating an AP and two ASUS EEPC netbook clients. We assume client-2 requests the AP for indoor localization at $t = 6$ s. We measure the time Chronos incurs to hop between the 35 WiFi bands. Meanwhile, client-1 runs a long-lasting traffic flow. We consider two types of flows: (1) VLC video stream over RTP; (2) TCP flow using iperf. We run the experiment 30 times and find aggregate results.

Results: Fig. 8(a) depicts the CDF of the time that Chronos incurs to hop over all WiFi bands. We observe that the median hopping time is 84 ms for the Intel 5300 WiFi card, like past work on commercial WiFi radios [29].

Next, Fig. 8(b) plots a representative trace of the cumulative bytes of video received over time of a VLC video stream run by client-1 (solid blue line). The red line plots the cumulative number of bytes of video played by the client. Notice that at $t = 6$ s, there is a brief time span when no new bytes are downloaded by the client (owing to the localization request). However, in this interval, the buffer has enough bytes of video to play, ensuring that the user does not perceive a video stall (i.e. the blue and red lines do not cross). In other words, buffers in today's video streaming applications can largely cushion such short-lived outages [26, 25], minimizing impact on user experience. Similarly, Fig. 8(c) depicts a representative trace of the throughput over time of a TCP flow at client-1. The TCP throughput dips only slightly by 18.5% at $t = 6$ s, when client-2 requests location. Thus, Chronos can support localization without much impact on data traffic. However, if more frequent localization is desired with large traffic demands, we recommend deploying a dedicated AP or WiFi beacon for localization.

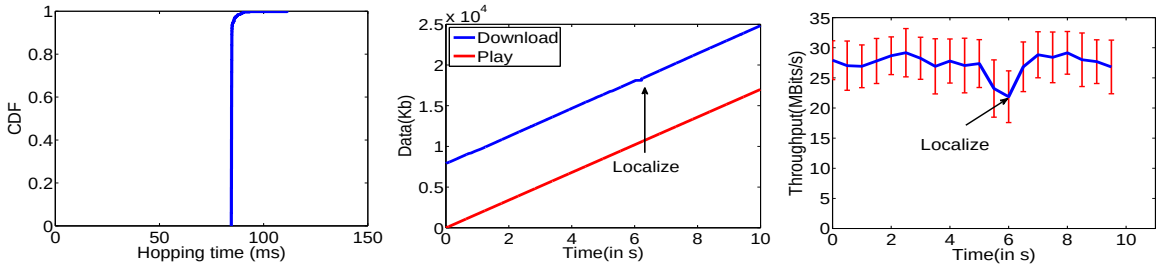
10. APPLICATIONS

We evaluate Chronos in three application scenarios.

10.1 Room Occupancy Detection

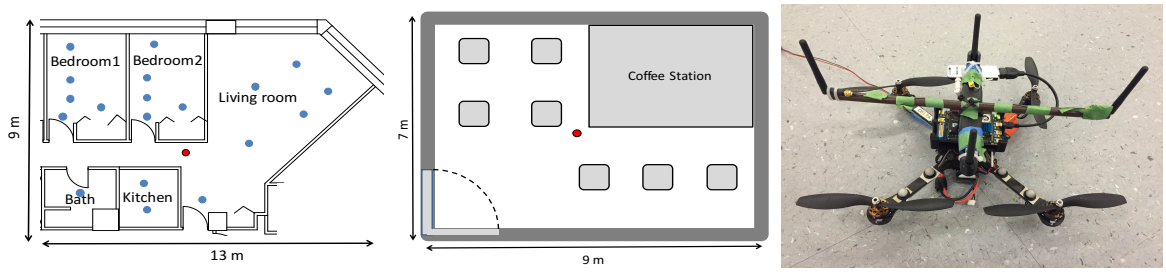
Smart home technologies, such as personalized heating and lighting, can vastly benefit from information about the number and identity of people in individual rooms. Chronos is a natural solution for this problem as it can localize and track people using their smartphones and wearables, even if the home has a single WiFi access point.

Method: To demonstrate this capability, we deployed Chronos in a two-bedroom apartment that has four res-



(a) Hopping Time (b) Video Streaming (c) TCP Throughput

Figure 8: **Impact on Network Traffic:** (a) measures the CDF of time taken by Chronos to hop between all WiFi bands – a small value of 84 ms. Consider a client-1 with a long-running traffic flow to an AP. The AP is asked to localize another client-2 at $t = 6$ s. (b) depicts a representative trace of the number of bytes of data downloaded and data played over time if the client-1 views a VLC video stream. (c) measures the throughput if client-1 runs a TCP flow using iperf. In either case, the impact of client-1's flow is minimal at $t = 6$ s.



(a) Home Floor Plan (b) Coffee Shop Schematic (c) Personal Drone

Figure 9: (a) Floor map of the apartment where Chronos is deployed. Red dot indicates the access point and the blue dots represent the client positions. (b) Coffee shop schematic. Red dot indicates the access point. (c) We implement Chronos on an AscTec Hummingbird quadrotor with an AscTec Atomboard.

idents. The floor map of the apartment is shown in Fig. 9(a). The Chronos access point is centrally placed in the home and is indicated by the red dot. Each resident is given an ASUS netbook, equipped with Intel 5300 WiFi cards, and running Chronos. The residents are then asked to move freely to locations within the apartment. Their locations are manually recorded and are marked by the blue dots in Fig. 9(a). Chronos measures the location of each resident and detects the room the person is in. In particular, Chronos distinguishes between the two bed rooms, living room, kitchen and bathroom.

Results: In our experiments, Chronos detects the user to be in the correct room in 94.3% of the experiments. Most of the errors occurred in Bedroom 1 in Fig. 9(a), and were due to the signal being too weak after traversing two walls and a closet. Overall, the results show that Chronos can enable applications based on room occupancy detection with a single home access point.

10.2 WiFi Geo-Fencing

Chronos can be used to authorize WiFi access in small businesses, which have only one access point. To demonstrate this capability, we deploy Chronos in a popular coffee shop with free WiFi, and use the distance from the access point to measure whether an individual is inside or outside the coffee shop (Fig. 9(b)).

Method: We conducted 100 experiments in the coffee shop. The user used an ASUS netbook, equipped with the Intel 5300 WiFi card to connect to the Chronos AP. In 50 of these experiments, the user was standing at a randomly chosen location inside the coffee shop, while in the others, the user was standing outside, while still being able to access the WiFi connection.

Results: Chronos correctly inferred whether the user was inside or outside in 97% of experiments. However, if we simply authenticate users based on location without any buffer zone, the accuracy is 97%, but one legitimate customer cannot access WiFi in his current location. In contrast, if we decide to accept users located within 30 cm of the premises, Chronos authenticates all legitimate customers but allows access incorrectly to people outside the premise in 5% of the experiments, decreasing the overall accuracy to 95%. Since it is more important to ensure customers can access WiFi, we believe that one should use some buffer zone.

10.3 Personal Drones

We apply Chronos to indoor personal drones [11]. These drones can follow users around while maintaining a convenient distance relative to the control device in the user's hand or pocket. Users can use these drones to take pictures or videos of them while performing an activity, even in indoor settings where GPS is unavailable.

Method: We use an AscTec Hummingbird quadrotor

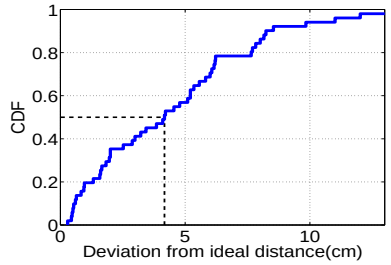


Figure 10: **Application to Personal Drones:** The drone uses Chronos to maintain a constant distance of 1.4 m to the user. The figure plots the CDF of errors in maintaining a distance of 1.4 m.

equipped with the AscTec Atomboard light-weight computing platform (with the Intel 5300 WiFi card), a Go-pro camera and a Yei-Technology motion sensor. Fig. 9(c) depicts our setup. Note that the Intel 5300 WiFi card supports 3-antennas; the fourth antenna on the quadrotor is placed only for balance and stability.

We perform our personal drone experiments in a 6 m $\times$ 5 m room augmented with the VICON motion capture system [2]. We use VICON to find the ground-truth trajectories of the personal drone and the user control device. In each experiment, the personal drone tracks an ASUS EEPC netbook with the Intel 5300 WiFi card held by a user. The distance measurements from Chronos are integrated with drone navigation using a standard negative feedback-loop robotic controller [12]. The drone maintains a constant height and follows the user to maintain a constant distance of 1.4 m relative to the user's device. The drone also captures photographs of the user along the way using the Go-Pro camera mounted on the Hummingbird quadrotor, keeping the user at 1.4 m in focus. The drone uses the compass on the user's device and the quadrotor to ensure that its camera always faces the user.

Results: Fig. 10 measures the CDF of root mean squared deviation in distance of the drone relative to the desired value of 1.4 m — a median of 4.17 cm. Our results reveal that the drone tightly maintains its relative distance to the user's device. Notice that our error in distance is significantly lower in this experiment relative to §9.2. This is because drones measure multiple distances as they navigate in the air, which helps de-noise measurements and remove outliers. Chronos is the first system to achieve such a high accuracy in device to device positioning using no support from surrounding infrastructure.

11. Related Work

Chronos builds on vast literature on indoor WiFi-based localization [40, 13, 9, 51, 18, 48, 32, 30, 50, 4]. However, past work that delivers sub-meter location accuracy typically requires cooperation across multiple (four or five) AP's [18, 48, 32, 30, 50].

A few prior proposals have aimed to localize with a single WiFi AP. They may be divided into two categories: some proposals [31, 52] require exhaustive fingerprinting of received signal power prior to deployment. Such proposals exhibit localization errors of several meters and incur a large overhead due to fingerprinting. The second class of proposals attempt to measure time-of-flight either directly [35], or indirectly using the phase [53]. However, since they cannot accurately measure the time-of-flight, they need the user to walk around, perform measurements in multiple locations, and intersect those measurements with the help of an accelerometer. In contrast, Chronos has tens of centimeter accuracy, and neither requires fingerprinting nor user motion.

A few past papers on WiFi-based localization leverage channel hopping [50, 49]. However, unlike Chronos which measures the absolute time-of-flight and localizes with a single AP, those systems measure differences in the time-of-flight and require the deployment of multiple AP's.

Prior theoretical ranging algorithms [44, 43] have used the Chinese Remainder theorem. However, Chronos differs from those algorithms in multiple ways. First, those algorithms ignore multipath and assume that wireless signals propagate in free space with a single time-of-flight value. In contrast, Chronos addresses the crucial problem of multipath, and hence its complete algorithm uses non-uniform Fourier transform as opposed to the Chinese Remainder theorem. Second, those algorithms ignore pratical issues such as the frequency offset between the transmitter and the receiver and the inability of the receiver to separate the time of flight from the packet detection delay.

Finally, some past work has explored measuring the time-of-flight of WiFi signals for reasons other than localization. There have been several studies that resolve time-of-flight to around ten nanoseconds using the clocks of WiFi cards or other methods [46, 33, 17, 34, 38]. In contrast, Chronos can achieve sub-nanosecond resolution which is necessary for accurate localization.

12. Conclusion

This paper presents Chronos, a system that measures sub-nanosecond time-of-flight on commercial WiFi radios. Chronos uses these measurements to enable WiFi device-to-device positioning at state-of-the-art accuracy, without support of additional WiFi infrastructure or non-WiFi sensors. By doing so, Chronos opens up WiFi-based positioning to new applications where additional infrastructure and sensors may be unavailable or inaccessible, e.g., geo-fencing, home occupancy measurements, finding lost devices, maintaining robotic formations, etc.

Acknowledgements: We thank the NETMIT group, Arthur Berger, our reviewers and our shepherd, Alex Snoeren, for their insightful comments. This work is funded by NSF. We thank members of the MIT Center for Wireless Networks and Mobile Computing: Amazon, Cisco, Google, Intel, Mediatek, Microsoft, ST Microelectronics and Telefonica for their interest and support.

13. REFERENCES

[1] Bosch Laser Distance Measurer GLM50. http://www.boschtools.com/Products/Tools/Pages/BoschProductDetail.aspx?pid=GLM\%2050.

[2] VICON T-Series. http://www.vicon.com/products/documents/Tseries.pdf.

[3] IEEE 802.11n-2009 Standard. 2009. http://standards.ieee.org/findstds/standard/802.11n-2009.html.

[4] O. Abari, D. Vasisht, and D. Katabi. Caraoke: An E-Toll Transponder Network for Smart Cities. SIGCOMM, 2015.

[5] F. Adib, Z. Kabelac, D. Katabi, and R. C. Miller. 3D Tracking via Body Radio Reflections. NSDI, 2014.

[6] Automated Home. Apple iBeacons Explained: Smart Home Occupancy Sensing Solved?, 2013.

[7] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Convex Optimization with Sparsity-Inducing Norms, 2011.

[8] S. Bagchi and S. K. Mitra. *The Nonuniform Discrete Fourier Transform and Its Applications in Signal Processing*. 1999.

[9] P. Bahl and V. Padmanabhan. RADAR: An in-building RF-based User Location and Tracking System . INFOCOM, 2000.

[10] W. U. Bajwa, J. Haupt, A. Sayeed, and R. Nowak. Compressed Channel Sensing: A New approach to Estimating Sparse Multipath Channels. Proceedings of the IEEE, 2010.

[11] Ben Popper. The Drone You Should Buy Right Now, 2014. http://www.theverge.com/2014/7/31/5954891/best-drone-you-can-buy.

[12] Y. Brun, G. Marzo Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Müller, M. Pezzè, and M. Shaw. *Software Engineering for Self-Adaptive Systems*. 2009.

[13] K. Chintalapudi, A. Padmanabha Iyer, and V. N. Padmanabhan. Indoor Localization without the Pain. MobiCom, 2010.

[14] C. Ding, D. Pei, and A. Salomaa. *Chinese Remainder Theorem: Applications in Computing, Coding, Cryptography*. 1996.

[15] A. Dutt and V. Rokhlin. Fast Fourier Transforms for Nonequispaced Data. *SIAM J. Sci. Comput.*, 1993.

[16] S. Gezici, Z. Tian, G. B. Biannakis, H. Kobayashi, A. F. Molisch, V. Poor, Z. Sahinoglu, S. Gezici, Z. Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu. Localization via Ultra-wideband Radios. In *IEEE Signal Processing Magazine*, 2005.

[17] D. Giustiniano and S. Mangold. CAESAR: Carrier Sense-based Ranging in Off-the-shelf 802.11 Wireless LAN. CoNEXT, 2011.

[18] J. Gjengset, J. Xiong, G. McPhillips, and K. Jamieson. Phaser: Enabling Phased Array Signal Processing on Commodity WiFi Access Points. MobiCom, 2014.

[19] L. Greengard and J. Lee. Accelerating the Nonuniform Fast Fourier Transform. *SIAM REVIEW*, 2004.

[20] M. Guillaud, D. Slock, and R. Knopp. A Practical Method for Wireless Channel Reciprocity Exploitation through Relative Calibration. 2005.

[21] D. Halperin, W. Hu, A. Sheth, and D. Wetherall. Tool Release: Gathering 802.11n Traces with Channel State Information. ACM SIGCOMM CCR, 2011.

[22] J. Heiskala and J. Terry. *OFDM Wireless LANs: A Theoretical and Practical Guide*. Sams Publishing, 2001.

[23] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins. *Global Positioning System: Theory and Practice*. Springer Science & Business Media, 2013.

[24] K. Hou, Z. Zhou, A. M.-C. So, and Z.-Q. Luo. On the Linear Convergence of the Proximal Gradient Method for Trace Norm Regularization. NIPS, 2013.

[25] T.-Y. Huang, R. Johari, and N. McKeown. Downton Abbey Without the Hiccups: Buffer-based Rate Adaptation for HTTP Video Streaming. FhMN, 2013.

[26] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. SIGCOMM, 2014.

[27] A. T. Islam and I. Misra. Performance of Wireless OFDM System with LS-Interpolation-Based Channel Estimation in Multi-path Fading Channel. IJCSA, 2012.

[28] K. Joshi, S. Hong, and S. Katti. PinPoint: Localizing Interfering Radios. NSDI, 2013.

[29] S. Kandula, K. C.-J. Lin, T. Badirkhanli, and D. Katabi. FatVAP: Aggregating AP Backhaul Capacity to Maximize Throughput. NSDI, 2008.

[30] M. Kotaru, K. Joshi, D. Bharadia, and S. Katti. SpotFi: Decimeter Level Localization Using WiFi. SIGCOMM, 2015.

[31] C. Kumar, R. Poovaiah, A. Sen, and P. Ganadas. Single Access Point-based Indoor Localization Technique for Augmented Reality Gaming for Children. Students' Technology Symposium (TechSym), 2014 IEEE, 2014.

[32] S. Kumar, S. Gil, D. Katabi, and D. Rus. Accurate Indoor Localization with Zero Start-up Cost. MobiCom, 2014.

[33] S. Lanzisera, D. Zats, and K. Pister. Radio Frequency Time-of-Flight Distance Measurement for Low-Cost Wireless Sensor Localization. *Sensors*

Journal, IEEE, 2011.

[34] A. Marcaletti, M. Rea, D. Giustiniano, V. Lenders, and A. Fakhreddine. Filtering Noisy 802.11 Time-of-Flight Ranging Measurements. CoNEXT, 2014.

[35] A. T. Mariakakis, S. Sen, J. Lee, and K.-H. Kim. SAIL: Single Access Point-based Indoor Localization. MobiSys, 2014.

[36] A. Nag and S. Mukhopadhyay. Occupancy Detection at Smart Home Using Real-Time Dynamic Thresholding of Flexiforce Sensor. *Sensors Journal, IEEE*, 2015.

[37] New York Times. Homes Try to Reach Smart Switch, 2015. `http://www.nytimes.com/2015/04/23/business/energy-environment/homes-try-to-reach-smart-switch.html`.

[38] H. Rahul, H. Hassanieh, and D. Katabi. SourceSync: A Distributed Wireless Architecture for Exploiting Sender Diversity. ACM SIGCOMM, 2010.

[39] H. Rahul, S. Kumar, and D. Katabi. MegaMIMO: Scaling Wireless Capacity with User Demands. ACM SIGCOMM, 2012.

[40] A. Rai, K. K. Chintalapudi, V. N. Padmanabhan, and R. Sen. Zee: Zero-effort Crowdsourcing for Indoor Localization. MobiCom, 2012.

[41] P. Setlur, G. Alli, and L. Nuzzo. Multipath Exploitation in Through-Wall Radar Imaging Via Point Spread Functions. *IEEE Transactions on Image Processing*, 2013.

[42] D. Tse and P. Vishwanath. *Fundamentals of Wireless Communications*. Cambridge University Press, 2005.

[43] C. Wang, Q. Yin, and H. Chen. Robust Chinese Remainder Theorem Ranging-method based on Dual-Frequency Measurements. *IEEE Transactions on Vehicular Technology*, 2011.

[44] C. Wang, Q. Yin, and W. Wang. An Efficient Ranging Method for Wireless Sensor Networks. ICASSP, 2010.

[45] E. Weisstein. Chinese Remainder Theorem. `http://mathworld.wolfram.com/ChineseRemainderTheorem.html`.

[46] S. B. Wibowo, M. Klepal, and D. Pesch. Time of Flight Ranging using Off-the-self IEEE802.11 WiFi Tags. POCA, 2009.

[47] WiFi Alliance. Make Security a Priority in 2011: Protect Your Personal Data on Wi-Fi Networks, 2011. `http://www.wi-fi.org/news-events/newsroom/make-security-a-priority-in-2011-protect-your-personal-data-on-wi-fi-networks`.

[48] J. Xiong and K. Jamieson. ArrayTrack: A Fine-Grained Indoor Location System. NSDI, 2013.

[49] J. Xiong, K. Jamieson, and K. Sundaresan. Synchronicity: Pushing the Envelope of Fine-grained Localization with Distributed Mimo. HotWireless, 2014.

[50] J. Xiong, K. Sundaresan, and K. Jamieson. ToneTrack: Leveraging Frequency-Agile Radios for Time-Based Indoor Wireless Localization. MobiCom, 2015.

[51] M. Youssef and A. Agrawala. The Horus WLAN Location Determination System. MobiSys, 2005.

[52] G. V. Zàruba, M. Huber, F. Kamangar, and I. Chlamtac. Indoor Location Tracking using RSSI Readings From a Single Wi-Fi Access Point. *Wireless networks*, 2007.

[53] X. Zheng, C. Wang, Y. Chen, and J. Yang. Accurate Rogue Access Point Localization Leveraging Fine-grained Channel Information. IEEE Conference on Communications and Network Security (CNS), 2014.