Universal Timestamping with Ambient Sensing

Adeel Nasrullah, Momin Ahmad Khan, Fatima Muhammad Anwar
Department of Electrical and Computer Engineering, UMass Amherst
Email: {anasrullah, makhan, fanwar}@umass.edu

2022 19th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON) | 978-1-6654-8643-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/SECON55815.2022.9918555

***Abstract*—Universal time is critical for coordinating functionalities among co-located as well as geographically distributed IoT devices. Current time alignment approaches for IoT devices rely on radio-based communication that puts an extra burden on the already resource-constrained devices. Other timing approaches depend either on customized hardware frontends or on fixed networking capabilities. Intermittent network connectivity further deteriorates timing performance for these devices. These constraints motivate us to create a new design that actively embeds time information into the surroundings of IoT devices that can harvest timing signals with off-the-shelf sensing capabilities. Our design decouples clock performance from the network uncertainties, introduce resource efficiency and extensibility, and requires no modification in existing devices. A unique property of our design is that it leverages the ubiquitous Electric Network Frequency (ENF) fluctuations as global time reference for the sensing devices, and takes on a variety of challenges at the intersection of sensing and signal processing to provide a universal sense of time without custom hardware frontends and network dependability. We evaluate the extensibility of our design in remote setups and show its robustness in real world settings.**

***Index Terms*—Ambient Sensing; Time Synchronization; Edge Computing**

I. Introduction

Internet of things (IoT) is increasingly becoming an integral part of modern infrastructure. Wireless Sensor Networks (WSNs) proliferate the urban infrastructure from smart homes, parking systems [1] to city-wide sensor networks [2]. These systems rely on a common notion of time to offer coordinated services. Universal time across a sensor network is essential for establishing causality among events detected by various sensors. Misaligned clocks across the sensor network make it difficult to draw complex relationships in data collected across devices [3]. As such, the devices, when not aligned, are more vulnerable to attacks by malicious agents [4]. Hence, a resilient and fault-tolerant time service is critical for WSNs and IoT applications.

While there exists many well-established techniques to achieve time synchronization across sensor networks, our work is motivated by several reasons. First, the existing approaches (NTP [5], PTP [6], FTSP [7]) rely on high packet exchange rate for good synchronization performance, however this high bandwidth requirement is an additional burden on the resource-constrained IoT devices. Second, most of these protocols require continuous network availability, but in reality, edge networks are intermittent and edge device clocks exhibit large drifts during network outages. These clock drifts are further exacerbated by packet delay attacks [4]. Third, the design of existing timing services assumes uniform networking stacks across the IoT devices. For example, while NTP works with all the devices with TCP/IP stack, BLE-enabled devices cannot directly access NTP services. Thus time synchronization quality may vary across sensor network deployments with heterogeneous networking resources.

Alternatively, sensor networks can be synchronized using external reference signals (Syntonistor [8], WiFi beacons [9], periodic FM broadcasts [10]). While these approaches ubiquitously sense the grid and radio signals, they rely on customized hardware frontends for signal sensing. These non-trivial extensions are not always feasible during network deployments.

Our work aims for a ubiquitous timing service for off-the-shelf sensing edge devices without any hardware modification and resource overhead. We take inspiration from ambient sensing that extracts timing information from environmental phenomenons. Our work employs passive sensing rather than active communication on edge devices, and significantly reduce the power and bandwidth requirements for synchronization. A key component of our approach is a timing signal with the following properties: (i) ubiquitous i.e. omnipresent over a wide area, (ii) takes a unique form over any given period to encode time information, and (iii) easy to capture. For example, Electric Network Frequency (ENF) signals are ubiquitous, and as demonstrated by Li, Tan, and Yau [11], ENF fluctuations are unique and are easily measured from a power socket using low-cost off-the-shelf components [12].

Leveraging ENF signals for timing is not new; a variety of approaches use hardware customization to sense ENF for universal timing. In contrast, our goal is to extend the ubiquity and uniqueness of ENF to commercial off-the-shelf (COTS) sensing devices, which otherwise have lower timing performance with existing universal timing solutions or lower extensibility with rigid designs. Different from ENF based time synchronization [11], we provide a new mechanism to actively embed ENF time information on sensing signals.

We design an edge time server that modulates and broadcasts ENF timing signals over a variety of common sensing modalities such as audio and light. Edge sensing devices harvest our ENF embedded signals for aligning their clocks globally. Our approach is the first one to explore the potential of commodity devices to broadcast reference timing signals in the environment and leverage them for time synchronization. A fundamental property of this design is to shift the burden of time synchronization away from resource-constrained edge devices to an edge server; that is responsible for both the broadcast and extraction of the time information from sensing data. Though a variety of sensors can be used, for brevity, we have chosen two of the most common sensing modalities i.e.

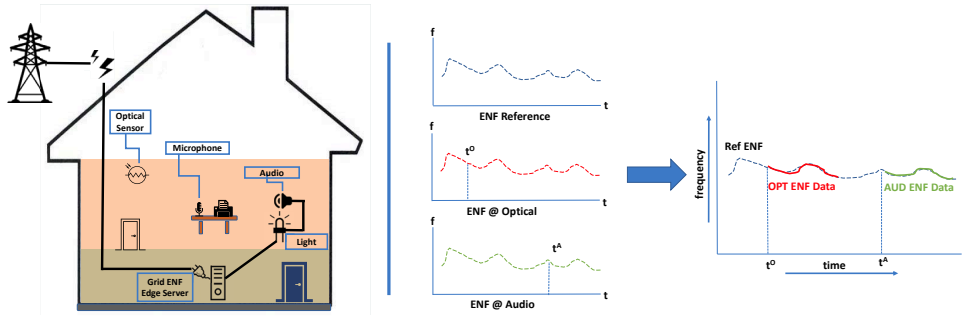


Fig. 1: System Overview

optical and audio sensors on IoT devices.

We demonstrate our design idea in Figure 1 with a smart space equipped with a variety of sensors. Our edge server records grid ENF from a power socket, then broadcasts the recorded ENF signal using light, and audio modalities[1]. The ENF artifacts in these signals allow different sensing devices (optical sensors and microphones in the figure) to observe the same pattern at any given time. By comparing these ENF artifacts in sensor data against the reference ENF at the server, we decode timestamps to provide a common notion of time to all IoT devices as shown in signal representations of Figure 1.

While ENF signal is widespread, modulating this signal for sensing devices poses several challenges. First, we should establish physical layer communication using visible light (VLC), sound, and other properties of sensing devices with low sampling rates using COTS components. Recent works have made significant progress towards establishing a VLC channel between devices [13]–[15]. However, these systems either use customized front-ends [14], [15] or employ a high sampling rate at the receiver. On the other hand, the use of sound for digital communication is rare. Google Tone [16], one of its kind product, uses sound to share URIs between personal computers. While the implementation details are not available, it targets the devices that can generate pure tones and record audio at high sampling rates. In addition, COTS devices typically have low sampling rates as compared to dedicated front-ends in customized VLC [14], [15] and microphones receivers for recording high-quality sound.

In contrast, our design is agnostic to customized frontends and instead use COTS hardware properties along with software optimizations to encode and decode ENF signals for timing. Our server design leverages a pulse width modulated (PWM) signal to achieve frequency modulation for ENF signal encoding. Most micro-controllers can produce PWM in hardware while dedicating computational resources to other tasks.

One of our other challenges is the lack of symbol frame synchronization between the sender (server) and receiver (IoT device). Both devices use COTS microcontrollers with a high mutual clock drift. We leverage preambles to synchronize the sender and receiver symbol frames periodically. Our last challenge is detection of invalid ENF data at the receiver to provide error-free ENF signal demodulation for low sampling rate sensor data. These errors result from unsuitable environmental conditions. We leverage an insight of periodic preambles transmissions to verify the validity of ENF data captured by IoT sensors.

[1]Server has the capability to use other modalities too such as vibration

Contributions of the paper. Our work provides universal time to ubiquitous IoT devices using ambient sensing while being resilient to network variations and agnostic to customized hardware deployments. Our main contributions are as follows:

- Our biggest challenge is in modulating ENF signals on a variety of sensing channels. We design a frequency-based data modulation technique that caters to low sampling rates on IoT devices using COTS equipment. Our results suggest that IoT devices equipped with optical and audio sensors (extensible to other sensors in future) can obtain timestamps with a comparable accuracy to NTP.
- Our algorithms run in real-time on the edge server and require no changes to off-the-shelf IoT devices. Our system requires no hardware extensions to capture the modulated ENF signal. During network outages, the devices stay in synchronization as long as the ENF signal is broadcasted via sensing channels.
- We conduct evaluation of the proposed system using a real-world and remote sensor network deployment. Our results suggest – depending upon the kind of sensing modality – that universal timestamps provided to IoT devices are accurate to 10s of milliseconds, which is significantly better than NTP performance on similar network deployments.

II. Literature Survey

Traditional time synchronization approaches rely on communication based protocols. NTP [5] is the most commonly used and universal protocol, but requires frequent packets exchange and reliable network access; which is often too expensive or not available for resource-constrained IoT devices. Similarly, FTSP [7] is a common protocol in aligning wireless sensor networks but it is restricted to devices with same hardware and software resources. GPS and PTP [6] are used to provide high timing accuracy for computing systems but they require specialized hardware at the edge devices. In contrast, our system provides a general framework for providing common time to resource-constrained devices with heterogeneous hardware and software platforms and works even in absence of reliable network leveraging electric network frequency artifacts.

Alternatively, ambient events can be used to establish common notion of time. For example, Trinity calibrates local device clocks using energy harvester's natural frequency [17]. Gupchup et al. [18] relies on annual solar cycles to extract natural timestamps from optical sensor data and Lukac et al. [19] develops models for seismic wave propagation, which enable them to synchronize time across seismic sensor network. These approaches, however, have limited applications and only provides coarse grained time information (in the order of several minutes). Our system design proposes a generalized solution that works with heterogeneous platforms with off-the-shelf sensing capabilities.

Electro-Magnetic Radiation (EMR) from electric powerlines have been explored extensively for time services in IoT systems. Syntonistor [12] relies on EMR signal periodicity to calibrate clocks, whereas, Li et al. leverages Electric Network Frequency (ENF) fluctuations, measured directly from electric sockets, as timestamp opportunities [11], [20]. While our system also exploits ENF fluctuations for clock calibration, we rely on prevalent sensing capabilities on commodity IoT devices to provide timestamps rather than using customized front ends.

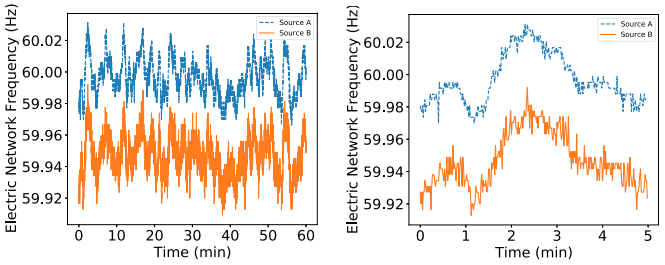
A variety of sensor-specific solutions have been proposed for clock calibration and verification purposes. For instance, ENF traces extracted from audio and video streams of multimedia files have been leveraged for clock synchronization [21]. Fluorescent fluctuations can be used to synchronize wireless sensor networks observing the fluorescent light sources connected to a single grid [22]. These works depend entirely on sensor specific sources i.e. fluorescent lights, which may not be present in most environments. They also rely on sophisticated external circuitry for sensing ENF artifacts. Such external peripherals have also been explored for time synchronization via radio broadcasts [10], [23]. In our work, however, we provide a ubiquitous solution for prevalent sensing modalities that leverage global ENF artifacts on commodity platforms, and at the same time, improve the accuracy of timestamp extraction to sub-seconds.

III. Background

This section explains the key idea behind our system design, i.e. the use of ENF fluctuations as timestamps.

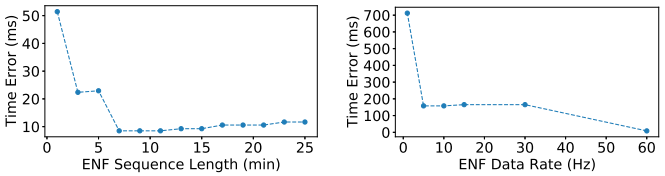
A. Electric Network Frequency (ENF) as Timestamps

The frequency of electric grids – an integral part of the modern infrastructure – is controlled centrally, and it is a function of power generation and load on the grid. Typically, the load on the electric network varies continuously, and the power generation is adjusted in response to the variation in grid load. The unpredictable changes in power input and output of an electric network give rise to unique fluctuations in ENF. These ENF fluctuations have traditionally been used to encode/decode time information [11]. Any device that can capture ENF fluctuations can obtain Universal Coordinated Time (UTC) from a server that stores historical (reference) ENF data along with UTC timestamps. Figure 2 shows that



(a) Over one hour (b) Over 5 minutes

Fig. 2: Fluctuations in the ENF data recorded from two separate power sockets supplied by the same electric network. For better visualization the two signals are offset (artificially) by 0.04Hz.



(a) Variation in time error with change in ENF series length (b) Time error with varying ENF sampling rate

Fig. 3: Time error among UTC timestamps of the ENF signal recorded from two power different sockets supplied by the same grid.

the fluctuations in ENF values recorded from two separate power sockets supplied by the same electric grid have identical changes over an hour. When sampled and correlated across distributed sources, yield a matching index.

Decoded Timestamps. An *ENF sequence* consists of a series x of n ENF measurements in time interval (t_x^S, t_x^E). Let y be a series of m reference ENF measurements at a server measured during time interval (t_y^S, t_y^E). We can decode the timestamp x if $t_x^S > t_y^S$ and $t_x^E < t_y^E$. We also record a UTC timestamp with each reference ENF measurement. To decode the ENF sequence x, we compute its root mean square error (RMSE) with y using a sliding window of length n. We determine the start index k of the sliding window for which the two sequences have minimum RMSE as

$$k = \underset{0 \le k \le m-n}{arg\ min}\ RMSE(x, y[k : k+n]) \tag{1}$$

The UTC timestamp t_k corresponding to index k is referred to as decoded timestamp for the ENF sequence x. While the error between decoded timestamp and the actual UTC timestamp, obtained for n ENF samples at the IoT device, is the decoding error or simply the time error. We will use this time error as a measure to evaluate the effectiveness of our design.

Electric grids often supply geographically wide areas. In theory, the ENF signal recorded from any two power sockets supplied by the same grid should yield zero time error when compared against each other. Figure 3a shows time error variation with the length of ENF series used for decoding timestamp. We observe that a series length between 420-600 seconds is optimal for decoding the ENF data. Figure 3b shows

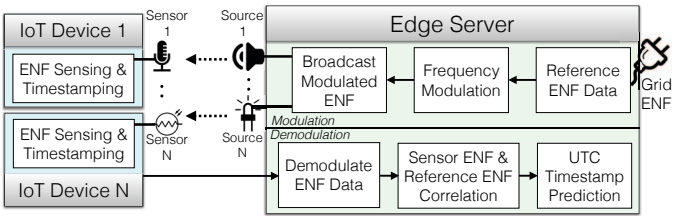


Fig. 4: Overview of System Design

the time error when using different numbers of ENF measurements per second (ENF sampling rate) for the decoding. It is clear a time error of few milliseconds is obtained. It is based on the ENF data rate and length of the ENF series used for decoding and results from the sampling noise at the two devices capturing the ENF signal. As a result the devices using ENF signal as reference clocks are indeed very tightly synchronized.

IV. System Design

We have designed our Universal Timestamping approach to align clocks on off-the-shelf IoT devices using prevalent sensing capabilities. Unlike traditional approaches that either rely on specific communication channels, customized hardware front ends or tied to a particular sensing technology, our system is agnostic to these requirements, rather align clocks globally with high accuracy on intermittently-connected devices with a variety of sensing capabilities.

In our system, we maintain a controller that acts as an edge time server. It maintains a reference ENF signal with encoded time information. One such server is enough to globally synchronize a variety of commodity IoT devices in a smart space. Figure 4 shows that our system design consists of a variety of IoT devices and an edge server. The edge server samples Grid ENF, modulates it and broadcasts time-encoded ENF signals through a variety of sensing sources (audio, light sources etc.), and the commodity IoT devices sample and timestamp these signals via corresponding sensing capabilities. This network-agnostic sensing-based communication puts no extra overhead on IoT devices, and bypasses uncertainties that stem from unreliable networks. After collecting enough timestamped sensor data, the IoT devcies send the data back to edge server for demodulation. This communication is not time-sensitive and does not impact clock accuracy. The server correlates demodulated ENF signals across multiple devices with reference ENF at the server to extract global universal coordinated (UTC) time.

Note that compute-intensive tasks of modulation and demodulation only occur at the server, thus relieving IoT devices with compute and communication overhead. In short, our design allows devices with a variety of networking and sensing capabilities to align time, in contrast to other approaches such as NTP [5] that assumes the availability of IP networking stack, PTP [6] that requires IEEE1588 compliant hardware, and Natural timestamping [11] with customized frontends tied to one sensing signal (powerline radiations). Below, we provide details of each design block in our system design i.e. modulation of ENF signals for broadcasting, how different

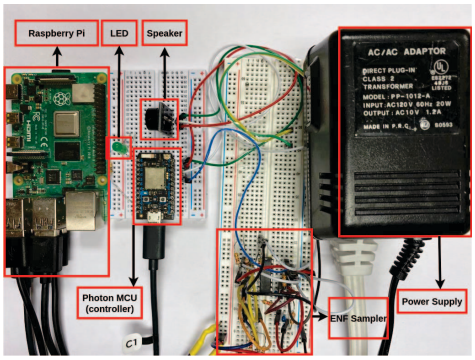


Fig. 5: Reference ENF Generation at Edge Server

sensors at IoT devices capture the broadcasted signal in the presence of noise and environmental variations, and how the server demodulates the signal to extract UTC time.

A. Edge Server

Edge server performs two critical tasks. First, it captures and then broadcasts time information encoded in ENF fluctuations. It is equipped with a reference ENF producing device originally designed by Sreejaya et al [20]. As shown in Figure 5, the server measures ENF directly from the wall power socket producing accurate ENF measurements. The alternating voltage from the electric network is converted into an analog PWM signal which replicates the input voltage's frequency. Then analog PWM is fed to an MCU, which measures the PWM period (period's inverse represents ENF frequency) using interrupts. MCU communicates these measurements to another processing unit which obtains UTC timestamps for each data point. We refer to these measurements at the edge server as our reference ENF measurements. The MCU also performs frequency-based modulation of the encoded data for broadcast into the environment using COTS components i.e. LEDs for lights signals and speaker for audio signals. We also refer to this MCU as a controller device in this paper. Figure 5 shows our hardware implementation of the edge server.

The second function of the edge server is to process the sensor data from IoT devices. It receives the sensor data offloaded by IoT devices and performs demodulation to extract ENF. The edge server then correlates the recovered ENF signal with the reference ENF measurements and determines the UTC timestamp against the encoded timestamp.

B. Time Embedding in Sensing Signals

The key component of our system design is broadcasting ENF data as encoded sensing input. We encode ENF data using frequency based modulation. Below we describe our approach behind the system design for broadcasting ENF data.

Unified Modulation Approach. While we modulate data for different sensing modalities, we aim to have a unified design at the controller. Such a design generates identical modulation of the ENF data for all sensing modalities. The only difference is in the hardware used for ENF data broadcast i.e. an LED for the optical and a speaker for the audio channel. It simplifies the controller design, saving computational resources. Typically,

pulse width modulation is used for the optical communication channels between the devices [14], [15]. But these techniques use customized front ends to decode the data. Low sampling rates on commercial IoT sensors are not feasible for PWM-based modulation techniques because their will only be a few samples to capture a single symbol, making it prone to noise. Amplitude-based modulation approaches are also not a good fit. Detecting amplitude changes in the signal is difficult without specialized tools, as it also varies with the distance. However, frequency-based modulation approaches are more robust. We can detect frequency changes in the signal much more reliably than the amplitude changes. Thus, many applications rely on frequency analysis of the signal to detect patterns [16], [24].

1) Modulation: Our system adopts a binary frequency modulation approach to encode ENF data for broadcast. In essence, this approach uses two frequencies f_0 and f_1 to encode 0 and 1 respectively. To modulate an ENF measurement using this method: i) convert each ENF measurement into a binary sequence of length a, ii) encode each bit of this binary sequence with a frequency (f_0 and f_1) signal of duration T and iii) transmit all frequency signals in a time slots with duration T. Our modulation design choices are explained below:

Modulation Signal. As shown in figure 5, an MCU attached to the Edge server is responsible for modulating the ENF measurements. It requires the MCU to produce sinusoidal signals of frequencies f_0 and f_1. However, most MCUs are not equipped with the hardware to produce sinusoidal signals, and do it in software. This consumes precious computational resources, and software-based generation may be interrupted frequently by context switching with other tasks. Alternatively, most MCUs are equipped with hardware to produce a pulse width modulated (PWM) signal that saves precious computational resources for the MCU.

Using PWM as a modulation signal restricts our choice of f_0 and f_1 for encoding binary data. PWM does not consist of a pure frequency, rather a combination of frequencies: $x(t) = \sum_{n=-\infty}^{\infty} c_n exp(jnwt)$. This represents the Fourier expansion of a periodic signal $x(t)$ where $x(t)$ is a PWM with a 50/% duty cycle, which is the case in our design. $w = 2\pi f$ represents base PWM frequency where f is the frequency in Hz. We can see that PWM has infinite frequency components (harmonics), where each component has a frequency that is an integer $n = 1, 2, 3, ..., \infty$ multiple of base frequency f. c_n is the complex co-efficient that determines the amplitude and phase of a given harmonic with frequency n. In case of a pure frequency (a sinusoidal signal), $c_n = 0$ for $n = 1, 2, 3, ...\infty$, whereas, a PWM signal with 50% duty cycle, $c_n = \frac{A}{2}\frac{sin(\pi n/2)}{\pi n/2}$ i.e. a non-zero value for $n = 1, 3, 5, ..., \infty$. Therefore, we use PWM to modulate our data as long as the two frequencies (f_0 and f_1) are not odd harmonics of each other and hence do not interfere. At the receiver, we filter the data to retain the base components of the broadcast frequencies and ignore the harmonics.

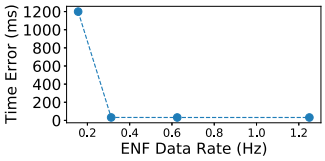
Modulation Frequency. Our modulation frequencies are limited by the sampling rate available at our target sensing IoT devices. Typical sensing devices have no customized front-ends and have relatively low sampling rates. These sensing devices have a modest sampling rate i.e. 1kHz as compared to 10kHz used in previous works [13]. According to the Nyquist principle, with 1 kHz sampling rate, the highest frequency we can capture is 500Hz. This determines the upper limit for the frequencies we can choose for modulation.

We follow these guidelines when selecting the modulation frequencies: 1) the selected frequencies should not be the odd integer multiples of each other (explained before through PWM signal properties), 2) there should be no interference from external sources, and 3) the selected frequencies should not disrupt the surrounding environment. The biggest source of external interference for optical and audio sensors are electromagnetic radiations (EMR) from power-lines at 60 Hz [25]. Another source of interference for optical sensors are fluorescent lights at 120 Hz. We avoid these frequencies as well as their harmonics to avoid interference. A visible light source operating at frequencies lower than 60 Hz may also cause flickering for some people.

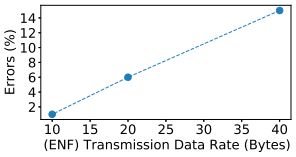
Keeping in mind these observations, we chose 210 and 270 Hz as our modulation frequencies. These frequencies are not integer multiples of each other (guideline 1). Neither of these frequencies overlaps with 60 Hz and its harmonics hence avoiding external interference (guideline 2). And they are high enough so that LEDs will not flicker (guideline 3). One can select any other frequency pair that follows this guideline. An additional advantage of our selected frequencies is that they lie on the edge of frequencies produced by adult human speech (85-255 Hz) [26]. Thus, we largely avoid interference from conversations between people as they are low pitch enough to blend in the background.

2) Demodulation: The modulated ENF measurements embedded in the environment are captured by the IoT devices via sensing. This data is offloaded to the edge server and demodulated to recover ENF measurements. A single ENF measurement is received as a frequency encoded signals of duration T. ENF Data is demodulated via these steps: (i) for each duration T of the received signal, determine magnitude of the two modulation frequencies (f_0 and f_1) (ii) compare the magnitudes of f_0, f_1 and (iii) output a zero bit if magnitude of f_0 is greater than f_1 and vice versa. (iv) convert a bit long binary sequence into an integer (which represents demodulated ENF data).

Traditionally, frequency-based modulation is decoded via Fourier transform analysis of an encoded bit duration T. However, the frequency granularity of the Fourier transform is limited at low sensor sampling rates. For example, consider a baud rate of $B = 10$ at a sampling frequency of $f_s = 1000$ Hz, we have $N = 100$ data points to detect each symbol. A Fast Fourier transform, in this case, will have a resolution of $\frac{f_s}{N} = 10$ Hz only. This level of granularity is insufficient if the selected modulation frequencies are not multiples of 10. It limits the choice of modulation frequencies for the system designers. Instead, we use an alternate approach to estimate the two frequency components in the sensor data.



(a) Time Error versus the ENF data rate (number of ENF measurements per second)



(b) Percentage Error in ENF data-points received at various Baud Rates

Fig. 6: (a) We observe low Time Error across range of ENF data rates except for 0.156 Hz when ENF data rate becomes too low to capture ENF fluctuations (b) Transmission Error in data increases with increase in Baud Rate, we select a baud rate of 10 bits/second where we get negligible error.

We compute the normalized correlation of the sensor data with sinusoidal signals of the frequencies ($f_0 = 210$ and $f_1 = 270$) used for modulation. It gives the magnitude of each frequency component in the data. Granularity is no longer an issue. We can chose any combination of modulation frequencies as long as they meet the guidelines outlined in section IV-B1.

3) Transmission Optimization: In this section, we analyze different parameters to achieve optimal ENF data transmission between the server and the IoT devices.

Baud Rate. This is the number of symbols transmitted per second B. It is eight times higher than the ENF data transmission rate $D = \frac{B}{8}$. We refer to the errors in the ENF data received at the IoT sensors as *transmission errors*. These are the number of incorrect bytes (one byte represents one ENF measurement) received at the IoT per 100 bytes (measured as percentage).

Ideally, we want to transmit all ENF measurements (an ENF sequence) obtained at the Edge server (60 measurements per second) with zero errors ($D = 60Hz$). In such a scenario, the IoT device will receive an ENF sequence identical to the one stored at the Edge server. Let x be the ENF sequence received at the IoT device and y be the reference ENF sequence stored at the Edge server. According to the equation (1), we will obtain the index k with RMSE = 0. It is the minimum possible RMSE value and the corresponding timestamp t_k will give us the minimum time error that can be obtained by decoding the ENF sequence x. This result is based on two assumptions: 1) the ENF sequence x is unique and matches y at one and only one location (section III-A) 2) The ENF sequence x is identical to the ENF sequence y stored at Edge server. Any ENF sequence which retains these two properties should yield minimum time errors, irrespective of the ENF data rate (D). Thus, we choose the baud rate B for which ENF transmission rate $D = \frac{B}{8}$ captures sufficient ENF fluctuations to be unique in a given period and have zero transmission error.

Figure 6 shows the transmission errors for different baud rates. We observe for $B = 10bps$, the transmission error is less than 1 percent. We assume that it is negligible error. Next, we proceed to determine if the ENF sequence received at $D = 1.25$ Hz ($B = 10$ bits per second) uniquely matches the reference ENF. If we see a time error less than $\frac{1}{D}$, an exact match has occured between the ENF sequence received at the IoT device and the one stored at the server. Otherwise, an error will be $\frac{j}{D}$ where j is the offset of the ENF sequence from exact matching index. Figure 6a, shows the time error at different ENF data rates D. From the figure it is clear that the time error for $D = 1.25$ Hz ($B = 10$ bits per second) is only few milliseconds, which is less than $\frac{1}{D}$. Not only that ENF sequence at $D = 1.25$ can yield an exact match, we can also assume 1% transmission error at this transmission rate negligible. As $D = 1.25$ Hz is the highest ENF transmission rate D for which we have negligible error, we choose a baud rate of $B = 10bps$ for our further experiments.

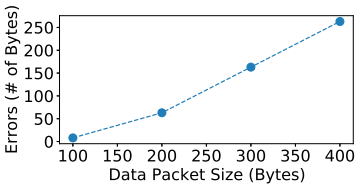


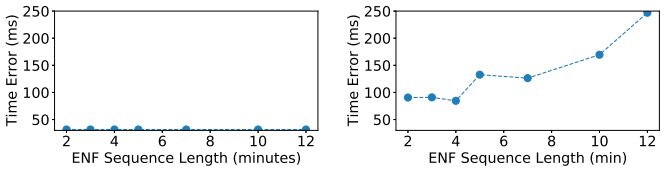
Fig. 7: Error rate with respect to the packet size

With negligible transmission error, timestamps obtained against received ENF sequence will give us the minimum time error. However, as this error is still non-zero, as the propagation of modulated ENF data over the physical medium (visible light and sound) takes time. This is often referred to as propagation delay. And with one way signal transmission, we cannot eliminate this error.

Bit Frame Synchronization. At $B = 10bps$, the server transmits one bit every $T = 100ms$, and we refer to this duration as bit frame. To demodulate the data, we need to accurately determine the start of bit frame. An offset of a few samples may alter the relative magnitude of f_0 and f_1 obtained from the bit frame. To avoid this problem, we align the bit frame between the sender and receiver using a preamble sequence before the transmission of ENF data. The preamble is a one-byte long sequence of bits with a fixed pattern 00000010. At the receiver, we scan for this preamble using auto-correlation. Once detected, the boundary between one and zero is considered as the starting point of the next bit frame.

Data Error Rate. As discussed above, we use preamble to align sender and receiver bit frames, however, their clocks exhibit relative drift and misalign over time after initial synchronization via preamble. We address this issue by periodic preamble transmission after a fixed number of ENF transmissions, thus allowing repeated calibration of bit frames during demodulation. We refer to one preamble and the ENF data transmitted in between two preambles as one data packet.

The periodic preamble reduces the amount of useful transmitted data. Figure 7 shows ENF data errors for different packet sizes. While there is a single error in a packet of 100 bytes (100 ENF measurements), error grows rapidly for larger packets. We can reduce the error by choosing a smaller packet size that will effectively reduce the useful bandwidth for ENF data broadcast. We end up choosing the packet size to be 100 bytes, where one byte is the preamble.



(a) Audio Sensor (b) Optical Sensor

Fig. 8: Time error variation with ENF sequence length

C. Optimizing ENF Decoding

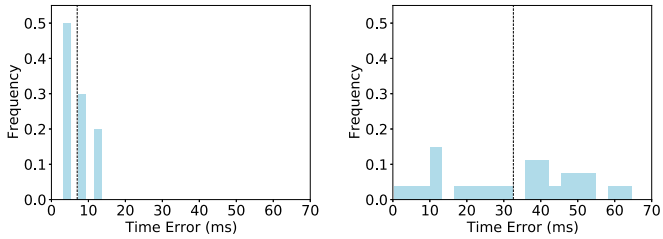
Our design broadcasts time information with encoded ENF fluctuations. With a baud rate of $B = 10bps$, we receive 1.25 ENF measurements per second. Before, we proceed with the evaluation of our design, we investigate the length of ENF sequence, that will give us the minimum decoding error. Figure 8a shows the time error with respect to the ENF sequence length. We can see that decoding error is in the same order of magnitude irrespective of ENF sequence length. It is the result of ENF data transmission with less than 1% transmission error. Thus, IoT devices receive an ENF sequence that is identical to the reference ENF at the edge server. This means that even 2 minutes of ENF measurements are sufficient to decode the timestamp.

However, in the presence of transmission errors (approximately 15%) during ENF data transmission ($B = 40$ bits per second) the relationship of time error with ENF sequence length is given by figure 8b. It shows that we need larger trace lengths in presence of errors (4 minutes) but too large a trace length will increase time errors. This also validates our design choice in section IV-B3 to use lower baud rate ($B = 10$ bps) with negligible transmission (1%) errors rather than higher baud rate ($B = 40$ bps) with higher transmission error rate (15%).

V. Implementation & Evaluation

We implement both local and remote sensing setup in a real world smart space, and evaluate the timing performance of our design as well as the system robustness.

Measurement Setup. For our experiments, we setup a Raspberry Pi based edge server as shown in Figure 5. This server receives high resolution ENF measurements captured by Particle Photon MCU, which has STM32F205RGY6 processor based on ARM Cortex M3. It then broadcast the received values over sensing channels using off-the-shelf components (an LED and speaker for optical and audio channels respectively). Our IoT device is Sparkfun ESP32 Things based micro-controller with Tensillica LX6 dual core processor, and run FreeRTOS based software stack. The MCUs used in this setup represent typical resource constrained IoT devices. To capture modulated ENF artifacts broadcast in the environment by the server, our IoT devices have sensors (photo-resistor and a microphone) interfaced with analog-to-digital converter pins of ESP32 devices. The sampling rate is set to 1kHz. The data from our IoT device is communicated to another Raspberry Pi device (different from our Edge server) over USB where we obtain UTC timestamps for each data sample.



(a) Optical Sensor (b) Audio Sensor

Fig. 9: Time error over sensing channel

The second Raspberry Pi device is part of a temporary setup and the timestamps obtained are only for the purpose of error calculation with the decoded timestamps.

A. Sensing Channel Performance

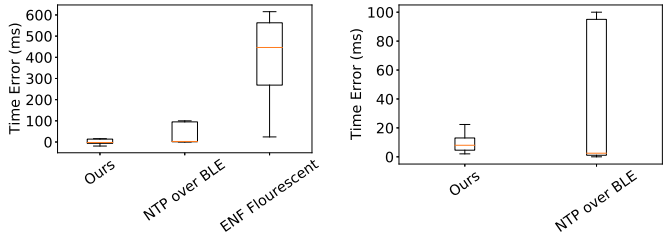
We first determine the error in UTC timestamps obtained over different sensing channels, then compare the performance with other baseline implementations.

1) Time Error over the Sensing Channel: To analyze the performance of our system, we collect data for both sensing modalities. The ENF data is broadcasted at a baud rate of 10 bits/sec. The preambles are transmitted every 100 bytes of data. As expected, we have less than 1% error in the ENF data received over the sensing modalities. It means we have received an ENF sequence at the sensing devices that is identical to the one recorded at the edge server. We use equation (1) to obtain timestamp against the received ENF sequence. Figure 9 shows the distribution of time error for the optical and audio sensing modalities. The mean error is 7 and 32 ms for the optical and audio sensing modalities respectively. As noted previously, the low error is due to the identical nature of the two ENF sequences being compared with each other, and the resulting errors are due to the delays from modulation at the photon and the propagation delays in the air. As speed of sound in the air is much lower than the speed of light, we see higher errors with audio sensor.

2) Comparison with the Baseline: To put the performance of our approach in context, we also implement NTP on our IoT device (ESP32 Things) that communicates with a RPi4 gateway device over a BLE connection. This represents a typical environment most sensing devices operate in. ESP32 performs NTP synchronization session every 15 seconds. For the optical sensor, we implement another baseline that consists of observing fluorescent lamps whose output fluctuates directly in response to input sinusoidal voltage from the grid. From figure 10a, we see that while median error for the optical channel is similar to that of NTP, our approach achieves better error spread and outperforms fluorescent light baseline in both median and error spread. The fluorescent light baseline performs indirect ENF observations from the fluorescent light, which contains significant error. On the other hand, our approach uses high precision circuit to record ENF and broadcasts it to the optical sensor with minimal error.

For the audio sensor in Figure 10b, our median error is slightly higher than the NTP baseline with a comparable error

spread for both. The slightly higher but stable error for the audio setup comes from the propagation delay over the air. As speed of sound is significantly lower than radio waves. On the other hand, NTP over BLE has lower median error. But, large errors may be experienced sometimes due to queuing and other delays on either end of the BLE connection.



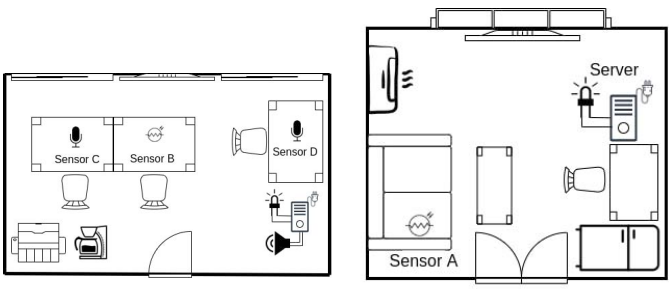
(a) Optical sensing with NTP and Fluorescent light baselines (b) Audio sensing with NTP baseline

Fig. 10: Error comparison of proposed sensing approach to baselines

B. Case Study

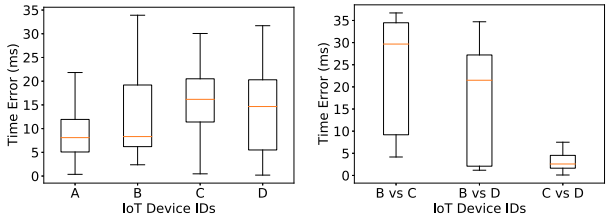
We leverage the ubiquitous ENF signal in modern sensing devices to achieve wide area synchronization. We setup two evaluation environments supplied by the same electric network. Figure 11a shows four IoT devices deployed in a work-space within an office building. The work-space has several power outlets in the wall powering several appliances including several LED screens, a printer and a coffee machine. A single ESP32 device is equipped with optical sensor while two devices are equipped with microphones. It also has RPi4 based Edge server measuring reference ENF and obtains UTC timestamps for each ENF measurement. The server attached to the RPi4 modulates and broadcasts the ENF measurements into the environment.

Our second setup consists of a single ESP32 Things device, interfaced with optical sensor, deployed in an apartment's living room (Figure 11b). This space also has several wall powered electronic devices in it. It is also equipped with an RPi4 based edge server that embeds ENF encoded timestamps into the environment. To calculate the error in decoded timestamps, we obtain UTC timestamp for sensor data from all



(a) Deployment of two audio and one optical sensor in an office space. (b) Deployment of an optical sensor in an apartment living room.

Fig. 11: Evaluation setup. Speakers and LEDs, which act as transmitters, are connected to the edge server.



(a) Time Error w/ corresponding edge Server (b) Time Error w/ colocated sensing devices

Fig. 12: Error median and spread for IoT devices in office and in remote apartment

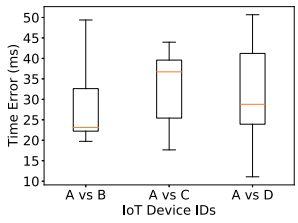


Fig. 13: Error median and spread among remote devices

our IoT devices in both these setups. All the IoT devices are operating at sampling rate of 1KHz. We are using a baud rate of 10 bits/second for these experiments, and decode ENF data sequence of 2 minutes period using equation (1) to obtain a UTC timestamp. These parameters have been selected based on our studies in Section V-A. The preamble is transmitted every 100 ENF measurements, and the packet size used is 100 bytes unless otherwise mentioned.

1) Time Errors between Co-located IoT devices: Figure 12a shows time decoding errors in IoT devices in an office space. We observe that median error for all the devices with respect to local edge server is under 20 milliseconds (msec) and overall errors are under 35 msec. Another important observation is that the median errors for the optical devices (A & B) is lower than the audio devices (C & D) due to the difference in speed of sound and light in the air. While the light only takes few nanoseconds to reach the IoT device, the propagation error will be several milliseconds for the audio sensor.

We also determine the error among pairwise IoT devices in figure 12b. An interesting observation is the error between the same and different sensing modalities. The error between optical (device B) and audio sensors (device C & D) vary significantly. The error between two different sensing modalities accumulates while the error between the same modality experience an improvement. The audio sensors are better synchronized with each other than they are synchronized with the Edge server. It is because the timestamps encoded in ENF sequences experience similar propagation delay over the audio channel. So, both of them have a similar offset with the server which is much larger than the error between the two audio devices.

2) Time Errors between Remote IoT devices: The previous section presents the time error among devices that share the same physical location. We observed that the error between

co-located IoT devices is less than 35 milliseconds. We can also synchronize devices across physical spaces spread over a wide geographical area. This area can be as large as supplied by a single electric grid. Figure 13 shows the error between IoT devices located at two distant spaces. The overall error between the remote IoT devices is comparable to error among co-located devices because the devices sample the same electric grid via broadcasts from their respective edge servers.

We also observe lower median error between same sensing modalities (devices A & B) than the different sensing modalities (devices A, C & D). It is the same behaviour we observed earlier where error across same sensing modalities is lower due to similar propagation delays.

C. System Robustness

While our system is capable of achieving better performance than other baselines in V-A, it is also robust to variations in real world deployments. For instance, signal broadcasts can be disrupted and produce an invalid ENF sequence with large time error. We argue that an ENF sequence of length n with a preamble every m data-points should contain n/m equally spaced preambles. In our design, the IoT device validates the received ENF sequence if the data packet has exactly n/m preambles in a sequence of length n and packet size m and all preambles are spaced at exactly m ENF measurements. By adjusting packet size m, we can tune noise based signal distortion with a trade off of transmission overhead.

VI. Discussion & Conclusion

Our design provides time services to COTS sensing devices at the edge by actively embedding universal timestamps in electric network frequency signal broadcasts. We overcome a variety of challenges in modulating unique ENF signals over different sensing channels and then demodulating these signals to extract universal timestamps. Though our approach relies on an edge server, it does not require hardware or software changes at the IoT devices. Our evaluation in indoor environments shows our design's extensibility to both local and remote environments with a variety of sensing devices.

Extensible Design. In this paper, we only explore two of the most common sensing modalities i.e. optical and audio sensors. However, our insights can be extended to other sensor types, for example, MEMs based inertial measurement units (IMUs) are known for responding to sound waves [27]. In future, we plan to extend our work with audio sensors to IMUs.

Universal Timing. The concept of universal timing was introduced by NTP based on its applicability to all kinds of devices. Recently, devices at the end of the network suffer poor performance with NTP due to multihop synchronization errors exacerbated by unreliable communication technologies. Our ENF based sensing solution removes network based uncertainties for end devices and provide universal time with high accuracy, however it comes at the cost of maintaining an edge time server. We argue that edge servers are already prevalent in networking applications serving as gateways, which we can leverage for timing capabilities for a modest cost.

References

[1] F. Al-Turjman and A. Malekloo, "Smart parking in iot-enabled cities: A survey," *Sustainable Cities and Society*, vol. 49, p. 101608, 2019.

[2] M. J. Potosnak, P. Banerjee *et al.*, "Array of Things: A high-density, urban deployment of low-cost air quality sensors," in *AGU Fall Meeting Abstracts*, vol. 2019, Dec. 2019, pp. A24G–04.

[3] C. Chen, S. Rosa, C. X. Lu, N. Trigoni, and A. Markham, "Selectfusion: A generic framework to selectively learn multisensory fusion," *arXiv preprint arXiv:1912.13077*, 2019.

[4] M. Ullmann and M. Vögeler, "Delay attacks — implication on ntp and ptp time synchronization," in *ISPCS*, 2009, pp. 1–6.

[5] D. L. Mills, "Internet time synchronization: the network time protocol," *IEEE Transactions on Communications*, vol. 39, no. 10, pp. 1482–1493.

[6] K. Lee, J. C. Eidson, H. Weibel, and D. Mohl, "Ieee 1588-standard for a precision clock synchronization protocol for networked measurement and control systems," in *Conference on IEEE*, vol. 1588, 2005, p. 2.

[7] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi, "The flooding time synchronization protocol," in *SenSys*, 2004.

[8] A. Rowe, V. Gupta, and R. Rajkumar, "Low-power clock synchronization using electromagnetic energy radiating from ac power lines," 01 2009, pp. 211–224.

[9] T. Hao, R. Zhou, G. Xing, and M. Mutka, "Wizsync: Exploiting wi-fi infrastructure for clock synchronization in wireless sensor networks," in *2011 IEEE 32nd Real-Time Systems Symposium*, 2011, pp. 149–158.

[10] L. Li, G. Xing, L. Sun, W. Huangfu, R. Zhou, and H. Zhu, "Exploiting fm radio data system for adaptive clock calibration in sensor networks," in *MobiSys*, 2011, p. 169–182.

[11] Y. Li, R. Tan, and D. K. Y. Yau, "Natural timestamps in powerline electromagnetic radiation," *ACM Trans. Sen. Netw.*, vol. 14, Jun. 2018.

[12] A. Rowe, V. Gupta, and R. R. Rajkumar, "Low-power clock synchronization using electromagnetic energy radiating from ac power lines," in *SenSys*, 2009, p. 211–224.

[13] R. Bloom, M. Z. n. Zamalloa, and C. Pai, "Luxlink: Creating a wireless link from ambient light," in *SenSys*, ser. SenSys '19, 2019, p. 166–178.

[14] J. Li, A. Liu *et al.*, "Retro-vlc: Enabling battery-free duplex visible light communication for mobile and iot applications," in *HotMobile*, 2015.

[15] X. Xu, Y. Shen *et al.*, "Passivevlc: Enabling practical visible light backscatter communication for battery-free iot applications," in *Mobicom*, 2017, p. 180–192.

[16] (2021) Google tone. [Online]. Available: https://chrome.google.com/webstore/detail/google-tone/nnckehldicaciogcbchegobnafnjkcne?hl=en

[17] F. Li, Y. Yang, Z. Chi, L. Zhao, Y. Yang, and J. Luo, "Trinity: Enabling self-sustaining wsns indoors with energy-free sensing and networking," *ACM Trans. Embed. Comput. Syst.*, vol. 17, no. 2, Feb. 2018.

[18] J. Gupchup, R. Musăloiu-E., A. Szalay, and A. Terzis, "Sundial: Using sunlight to reconstruct global timestamps," in *Wireless Sensor Networks*, U. Roedig and C. J. Sreenan, Eds., 2009, pp. 183–198.

[19] M. Lukac, P. Davis, R. Clayton, and D. Estrin, "Recovering temporal integrity with data driven time synchronization," in *IPSN*, 2009.

[20] S. Viswanathan, R. Tan, and D. K. Y. Yau, "Exploiting power grid for accurate and secure clock synchronization in industrial iot," in *2016 IEEE Real-Time Systems Symposium (RTSS)*, 2016, pp. 146–156.

[21] H. Su, A. Hajj-ahmad, M. Wu, and D. W. Oard, "Exploring the use of enf for multimedia synchronization," in *in IEEE Int'l Conf. Acoustics, Speech, and Signal Processing (ICASSP*, 2014.

[22] Z. Li, W. Chen, X.-Y. Li, and Y. Liu, "Flight: clock calibration using fluorescent lighting," 08 2012.

[23] Y. Chen, Q. Wang, M. Chang, and A. Terzis, "Ultra-low power time synchronization using passive radio receivers," in *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*, 2011, pp. 235–245.

[24] Y. Michalevsky, D. Boneh, and G. Nakibly, "Gyrophone: Recognizing speech from gyroscope signals," in *USENIX Security 14*, pp. 1053–1067.

[25] N. Fechner and M. Kirchner, "The humming hum: Background noise as a carrier of enf artifacts in mobile device audio recordings," in *2014 Eighth International Conference on IT Security Incident Management IT Forensics*, 2014, pp. 3–13.

[26] I. R. Titze and D. W. Martin, "Principles of voice production," *The Journal of the Acoustical Society of America*, vol. 104, 1998.

[27] S. Khazaaleh, G. Korres, M. Eid, M. Rasras, and M. F. Daqaq, "Vulnerability of mems gyroscopes to targeted acoustic attacks," *IEEE Access*, vol. 7, pp. 89 534–89 543, 2019.