LongShoT: Long-Range Synchronization of Time

Ceferino Gabriel Ramirez
Carnegie Mellon University, Silicon Valley
Moffett Field, CA
cef.ramirez@sv.cmu.edu

Assya Dyussenova
Carnegie Mellon University, Silicon Valley
Moffett Field, CA
assya.dyussenova@sv.cmu.edu

Anton Sergeyev
Carnegie Mellon University, Silicon Valley
Moffett Field, CA
anton.sergeyev@sv.cmu.edu

Bob Iannucci
Carnegie Mellon University, Silicon Valley
Moffett Field, CA
bob@sv.cmu.edu

ABSTRACT

Low-Power Wide Area Networks, such as LoRaWAN, are rapidly gaining popularity in the field of wireless sensing and actuation. While LoRaWAN is heavily studied in applications and performance, the concept of time has rarely been characterized in such networks. Many applications will require synchronized local clocks with varying levels of precision in order to maintain consistency and coordination in the network. Traditional time synchronization protocols however do not fit LoRaWAN's delay-inherent, low duty cycle, network model and wide-area deployment topology. Meanwhile, relying on GPS for time is not an option for low-power applications.

In this paper, we present LongShoT, a time synchronization scheme built on LoRaWAN capable of synchronizing device clocks to within $10\mu\text{s}$ of a reference clock with a single network request. This is achieved by utilizing the deterministic properties of LoRaWAN networks along with hardware- and MAC-level timestamping of packets. LongShoT was implemented on consumer off-the-shelf hardware and evaluated over physically distributed devices using GPS 1PPS as a reference. Our results show that LongShoT achieves an average synchronization error of less than $2\mu\text{s}$ and compensates oscillator drift to less than 0.1ppm with devices distributed within 4km of a gateway.

CCS CONCEPTS

• **Networks** → **Time synchronization protocols**; • **Computer systems organization** → **Sensor networks**.

KEYWORDS

Time Synchronization, LoRaWAN, Low-Power Networking

ACM Reference Format:

Ceferino Gabriel Ramirez, Anton Sergeyev, Assya Dyussenova, and Bob Iannucci. 2019. LongShoT: Long-Range Synchronization of Time. In *The 18th International Conference on Information Processing in Sensor Networks (co-located with CPS-IoT Week 2019) (IPSN '19)*, April 16–18, 2019, Montreal, QC, Canada.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

IPSN '19, April 16–18, 2019, Montreal, QC, Canada

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-6284-9/19/04...\$15.00

<https://doi.org/10.1145/3302506.3310408>

QC, Canada. ACM, New York, NY, USA, 12 pages. <https://doi.org/10.1145/3302506.3310408>

1 INTRODUCTION

Low-Power Wide Area Networks (LPWANs), such as LoRaWAN[15], are rapidly gaining popularity in the field of wireless sensing and actuation [2, 20]. Their long-range, low-power properties makes them an enticing option for Smart City applications[10].

LoRaWAN is one of the most often cited LPWANs. Its open architecture and consumer hardware makes it easily accessible for deployment and application studies. While several studies have looked into LoRaWAN in terms of applicability[22, 27] and performance[1, 4, 19, 35], we concern ourselves with an important aspect of distributed systems — *the concept of Time*.

As with traditional sensor networks, a common notion of time is important to any sensing and actuator system to maintain data consistency and coordination of devices throughout the distributed system[32]. The level of time accuracy and precision varies by application requirements, whereas a traffic signaling system may require only second-level accuracy, while a localization and tracking system will require microsecond-level accuracy[31, 33].

Existing time synchronization protocols do not scale well to LoRaWAN, where stringent bandwidth and battery constraints limit the rate at which devices can synchronize for accurate time. Traditional wireless sensor network (WSN) synchronization approaches [6, 9, 18, 24] are not applicable due to the difference in network topology (star vs. mesh) and the preference for devices to be shut-off most of the time. LoRaWAN devices can neither participate in LAN and WAN based methods[5, 21] to acquire time. A recent update to the LoRaWAN specification[16] includes a *DeviceTimeRequest* command which defines that the network should provide time to a device with an accuracy of $\pm 100\text{ms}$. The specification, however, does not provide any implementation details or strategies on how to maintain synchronicity of devices.

Other options acquire time through ambient signals, such as GPS. This is especially applicable to systems distributed over a wide geographical area. These approaches, however, do not fit the constrained energy requirements and will quickly drain the batteries of such devices.

There exists a need for a time synchronization protocol for LoRaWAN that distributes precise time over the network while maintaining its low-power properties. Such a protocol must address the following challenges:

- **Delay-Inherent Network** - Low data rates in the physical layer and network protocol amounts to considerable, but deterministic, delays in network transactions such that oscillator drift becomes a visible factor in synchronizing time. The protocol must be able to compensate for oscillator drift over the presumed delay period.
- **Substantial Propagation Delay** - At the ranges that LoRa supports, wireless propagation delay also begins to be a factor. Approximately, a $1\mu s$ offset is added for every 300m of distance. Without knowing the precise locations of the gateway and device, the protocol must be able to account for propagation delay.

In this paper, we present LongShoT, a time synchronization scheme built on LoRaWAN capable of synchronizing time to within $10\mu s$ of a reference with one LoRaWAN transaction at low duty cycles. We implement LongShoT on commercial-off-the-shelf hardware and evaluate with a set of devices distributed within a 4km radius of a LoRaWAN gateway. Our results show that LongShoT maintains an average synchronization error of less than $2\mu s$, measured over two hours with a single synchronization transaction every five minutes. The key contributions of this paper are:

- an in-depth timing analysis of LoRaWAN Class A devices to implement a sub- $10\mu s$ accurate time synchronization protocol while preserving low-power requirements;
- an evaluation of LongShoT implemented on off-the-shelf hardware and comparing the accuracy against GPS time at a distance of up to 4km;
- an energy comparison between LongShoT and a GPS receiver as a means for wide-area time synchronization.

The remainder of this paper is as follows. Section 2 provides a timing analysis of LoRaWAN and the underlying hardware to determine the physical limits that bound synchronization accuracy. Section 3 describes the protocol and the approaches to overcome the various sources of error. Section 4 contains the experiment setup and results gathered from our evaluation of LongShoT. Section 5 presents an energy comparison of synchronizing time using LongShoT and a consumer grade GPS receiver as a means for wide-area time synchronization. Section 6 offers a review of related work on time synchronization and their comparison with LongShoT. Finally, the paper summarizes our contributions in Section 7.

2 TIMING IN LORAWAN

LoRaWAN is a relatively new protocol and its timing aspects are least understood. This section gives a brief overview of LoRaWAN protocols for low-power devices. We present the challenges that LongShoT faces in determining the local clock offset along with compensating for both oscillator drift and propagation delay. Furthermore, we investigate and discuss timing uncertainties related to the wireless transmission and present a strategy to mitigate these effects.

2.1 LoRaWAN Protocol

LoRaWAN uses a star of stars topology where end-devices communicate directly with a gateway. It specifies three classes for end-devices[16]. Each class is targeted at different energy requirements.

LongShoT is designed to operate for Class A devices, the most energy constrained class.

Class A devices conserve energy by being offline most of the time. An end-device cannot receive a message unless it has transmitted an uplink packet. After transmission, the end-device opens two possible receive windows. The first receive window is a network parameter defined as $RxDelay$ and by default is set to one second. The second receive window is opened 1 second after the first, if no message is received in the first window.

LoRaWAN also defines a *DeviceTimeRequest* command that is compatible with Class A devices. This command requires the network to respond with time accurate to $\pm 100ms$. LongShoT does not rely on this command and provides $\pm 10\mu s$ accurate time, along with strategies to compensate for drift.

2.2 Two-Way Time Transfer and Drift

The two-way time transfer model is the basis of many time synchronization schemes[5, 11, 21]. It holds an advantage over one-way models to account for propagation delay between two devices. It is also compatible with LoRaWAN Class A device protocols in which a synchronization request is end-device initiated. Figure 1 shows a diagram of the two-way model annotated with LoRaWAN parameters.

Figure 1: Two-way time transfer model

The timestamps t_0 , t_1 , t_2 and t_3 mark the packet send and receive times based on the local clocks of a Client and Reference time server. At a given time, the local clock of a client device is offset from the reference clock by some value of θ .

$$t_{Reference} = t_{Client} + \theta \quad (1)$$

Using the send and receive timestamps, we can calculate the clock offset for uplink and downlink packets, while taking into account the propagation delay, T_{prop} .

$$t_1 = t_0 + \theta_{uplink} + T_{prop} \quad (2)$$

$$t_2 = t_3 + \theta_{downlink} - T_{prop} \quad (3)$$

where

$$\theta_{uplink} + T_{prop} = t_1 - t_0 \quad (4)$$

$$\theta_{downlink} - T_{prop} = t_2 - t_3 \quad (5)$$

In the classical model, it is assumed that θ remains constant over the time-transfer period. This approach, however, cannot be directly applied to LoRaWAN due to (a) the roundtrip time (T_{rtt}) of a request, in the order of seconds; and (b) the local oscillator drift, α ,

which for inexpensive sensors can be in the orders of $10 - 100\text{ppm}$, such that

$$\theta_{downlink} = \theta_{uplink} + \alpha \cdot T_{rtt} \quad (6)$$

$$T_{rtt} = T_{RxDelay} + 2 \cdot T_{prop} \quad (7)$$

An $RxDelay$ of 1s is significant enough to introduce up to $5\mu\text{s}$ error with a 5ppm crystal.

The challenge is further compounded by substantial propagation delays in the wireless range of LoRaWAN. A distance of 1.5km between an end-device and gateway adds a $5\mu\text{s}$ offset between the local clocks. Having drift error and propagation delay in the same order of magnitude prevents us from compensating for one effect without independently measuring the other. LongShoT presents a method to account for both drift and propagation delay with two consecutive synchronization requests.

2.3 Timing Uncertainties

The sources of error in time synchronization protocols are mainly determined by the variability of delays in measuring time and sending them over the network. These delays have been decomposed extensively in literature[9, 18]. We offer a view of how these values are present in LoRaWAN and if they can be mitigated.

2.3.1 LoRa PHY Limitations. The LoRa PHY layer uses a Chirp Spread Spectrum (CSS) modulation scheme with a narrow bandwidth of either 125 , 250 or 500kHz . In this scheme, each symbol is encoded using a sequence of chips on a linearly swept frequency, or chirps.

The length of each chip is derived from the bandwidth of the signal, which defines the chipping rate. A 125kHz bandwidth corresponds to 125 kilo-chips-per-second ($kcps$) [28], or $8\mu\text{s}$ per chip.

When a signal is transmitted over the air, noise and other effects, such as multi-path, may blur the boundaries between each chip, therefore causing a misalignment during demodulation. In such cases, a frame misalignment of more than the chip length will cause a demodulation failure and ultimately drop the frame. As such, a demodulated frame may have a maximum timestamp error up to the chip length in low SNR scenarios.

Similar protocols that use CSS modulation, such as $802.15.4a$, have been studied to provide time of arrival measurements down to the order of a few nanoseconds provided good SNR [3]. Given acceptable SNR, we can expect the same accuracy in arrival time estimation using LoRa. As SNR lowers, the delay uncertainty rises towards the upper bound as defined by bandwidth limitations.

With consumer LoRa radio hardware, we cannot inspect the approaches taken internally to mark the arrival of a packet. As such, we treat the radio subsystem as a black box that is capable of marking an incoming packet to within a few nanoseconds given good signal conditions, but may exhibit worst-case errors of up to $8\mu\text{s}$ for 125kHz transmissions with low SNRs.

2.3.2 LoRaWAN Hardware. In LoRaWAN, the radio hardware of an end-device is different from the gateway radio. The end-device is typically an SX1272/76[29] which is capable of half-duplex communication. The end-device radio does not contain a hardware counter and relies on an interrupt line to signal the host microcontroller when a transaction is completed.

The gateway radio typically contains an SX1301[30] which is capable of decoding up to 8 simultaneous LoRa packets. The SX1301 contains a 32-bit hardware counter with a resolution of $1\mu\text{s}$, which can also be slaved to an external GPS 1PPS signal. This unique addition allows the gateway to timestamp incoming packets in the radio hardware or to initiate sending on a scheduled time — removing all variable delays from OS system calls on the gateway side.

2.3.3 Sources of Delay. To determine the bound of LongShoT's accuracy, we decompose and describe the sources of the delay found in LoRaWAN and discuss how they can be mitigated.

Send Time and Receive Time — These delays are incurred due to variability in software and processor load when an application initiates a transmission cycle. End-devices can mitigate these delays by performing time labeling at the boundary between the MAC and the radio, usually on an interrupt signal. On the gateway, these delays are deterministic due to the radio hardware, which performs internal timestamping on Rx and supports scheduled Tx based on a $1\mu\text{s}$ hardware counter.

Access Time — A delay used to access the wireless channel on transmit and minimize collisions. Since LoRaWAN is an ALOHA-based protocol, this is negligible. Furthermore, LoRa radios provide an interrupt after a transmission is completed. Capturing time on this signal is more deterministic. Gateways can ignore this time since they implement scheduled transmissions to respect protocol definitions.

Transmission and Reception Time — The time it takes to emit or receive an entire packet in the wireless channel, also known as Time On Air (ToA). This time can vary from tens to a few hundred milliseconds, but can be calculated by the length of the frame and the selected radio data rate.[29]

Propagation Time — The time it takes for a wireless signal to travel the distance between sender and receiver. This is deterministic for non-moving devices, but can be in the order of a few to tens of microseconds for devices placed beyond 300m from a gateway.

Interrupt Handling Time — This time is dependent on the behavior of the host microcontroller on end-devices. These are usually less than a few microseconds but can vary due to the number of interrupts being handled by the processor. Interrupt routines must be carefully designed to minimize this effect and ensure high-accuracy of time labeling.

Radio State Delays — These delays are attributed to different stages a hardware radio moves through to perform specific functions. Typical delays are encoding/decoding time and frequency synthesis. While these delays are from deterministic state machines, they dilute the precision of time if not accounted for. We describe these delays in further detail and how they are handled in the following section.

2.4 Precise Time Labeling

Time synchronization accuracy depends on how well we can label specific points in time, e.g. the time a packet was sent or received. These specific points in time, however, are embedded in hardware functions that prevent us from precisely measuring those moments — unless properly supported by the hardware themselves. Such is

Figure 2: Timing measurements of SX1276 Tx/RxDone interrupts. In (a), the interrupt line is raised after the radio ramps down the power amplifier. This value is configured in registers and is set to $40\mu s$ by default. In our measurements, we see an additional $3.6\mu s$ from the time the power draw starts to fall. The ramp down time is independent of data rate, packet length and transmit power. In (b), the interrupt line is raised $1.48ms$ after receiving the last symbol ($t < 0.5ms$). This is most likely due to processing and decoding, which is supported by the slight increase in power draw during that interval. The decoding time varies depending on the data rate used for downlink.

the case for *Radio State Delays* which envelop the precise event time of our packets.

We define a Precise Time Labeling scheme for LongShoT to clearly specify the packet event times that need to be captured. This is presented in Figure 3. LongShoT uses the time of the last bit to enter/exit the radio as the reference point for determining the precise time label. Furthermore, we go into detail on how each of the time labels will be measured based on observable signals.

Figure 3: Determining a precise time label. Without hardware timestamp support, we can only measure a packet event from either t_{pre} or t_{post} after a notification from a radio. If we can measure precise timing behavior of the hardware, we can determine $t_{precise}$ by applying that knowledge to event time measured by devices.

2.4.1 End-Device Time Labels. For end-devices, we rely on interrupt signals from a LoRa radio to indicate the end of Tx and Rx. We label these events as t_{tx_done} and t_{rx_done} , respectively. These, however, do not indicate the exact moment a packet is sent or received by the radio.

We determine the timing behaviors of the radio by comparing the Tx/Rx RF lines, power input line and interrupt line with an oscilloscope. These can be seen in Figure 2.

For Uplink Send Time, t_0 , we see a significant delay between the TxDone interrupt and the drop in voltage for both power and RF lines. According to the datasheet[29], this is attributable to the ramp down of the power amplifier after transmission. The ramp down time, T_{ramp} , is configurable through device registers, and is set to $40\mu s$ by default. However, our measurements indicate a

Downlink Data Rate	Decode Time (μs)
DR10 / SF10	1487 ± 1
DR11 / SF9	717 ± 1
DR12 / SF8	339 ± 1
DR13 / SF7	176 ± 1

Table 1: Measured decoding time of a 8-byte payload Lo-RaWAN downlink packet at different data rates.

$43 \pm 3\mu s$ delay, most likely due to additional state transitions of the radio.

For Downlink Receive Time, t_3 , we see an even higher delay between the RxDone interrupt and the last symbol observable on the RF input line. This is likely due to processing and decoding of the received signal, as indicated by the rise in power over that period of time. The decoding time is dependent on the data rate used for downlink. Table 1 shows our measured decoding times for different data rates.

2.4.2 Gateway Time Labels. Gateway time labels are expected to be more accurate due to the use of an internal hardware counter for timestamping and scheduled downlink transmissions. Consequently, without access to timing signals externally, we cannot reliably measure the accuracy of these mechanisms.

According to the SX1301 datasheet [30], receive packets are timestamped with a $\pm 1\mu s$ accuracy. We cannot however determine whether this reported timestamp includes any processing delays. It has been reported by [26] that a gateway can timestamp packets with an accuracy of $\pm 3\mu s$. We take the reported timestamp as is and assume that it accurately represents exact receive time of the incoming packet, t_1 .

For measuring scheduled downlink transmission accuracy, we rely on a client radio RxDone signal. We place the client radio to continuously wait for a downlink packet and using modified gateway software, we inject known downlink packets to send on the rising edge of GPS 1PPS. By comparing the 1PPS rising edge and the RxDone rising edge, we determine the length of a complete downlink transmission. Subtracting out the airtime of the packet, the

known decoding time of the client radio, and negligible propagation delay, should result into the actual start time of transmission. Our measurements show a residual time of $6 \pm 1\mu s$ from the total time. This indicates that a gateway emits the first symbol on to the air at $6\mu s$ after the scheduled time. We label this delay as $T_{GwTxDelay}$. By adding the downlink packet time on air with this transmit delay, we can compute the expected value for time label t_2 .

2.4.3 Summary. We define how standard time labels, t_0 , t_1 , t_2 and t_3 , are measured based on the observable signals. Table 2 summarizes these definitions, while Table 3 consolidates the network delay values for LoRaWAN. Consequently, the network roundtrip time, T_{rtt} can be computed using Equation 8.

$$T_{rtt} = T_{RxDelay} + T_{GwTxDelay} + T_{ToA} + T_{decode} - T_{ramp} + 2 \cdot T_{prop} \quad (8)$$

Time Label	Definition
t_0	$t_{tx_done} - T_{ramp}$
t_1	as reported by gateway
t_2	$t_1 + T_{GwTxDelay} + T_{RxDelay} + T_{ToA}$
t_3	$t_{rx_done} - T_{decode}$

Table 2: Precise Time Label Definitions

Measurement	Value
T_{ramp}	$43\mu s \pm 3\mu s$ (Configurable)
$T_{GwTxDelay}$	$6\mu s \pm 1\mu s$
T_{prop}	$1\mu s$ per 300m of range
$T_{RxDelay}$	Network configured parameter
T_{ToA}	See [29]
T_{decode}	See Table 1

Table 3: Summary of Network Delays Values

3 LONGSHOT PROTOCOL

In the previous section, we described how the classical two-way time transfer model is inadequate for synchronizing time in LoRaWAN due to (a) the accumulated drift over the network roundtrip time and (b) the inability to distinguish the offset due to drift versus that of propagation delay. In this section, we present how LongShoT addresses these factors to provide synchronization to within $10\mu s$.

We first describe the packet transactions made by the end-device and the network response in order to determine the necessary time measurements. We then describe two synchronization approaches: One-Shot Synchronization and Two-Step Correction. Both provide accurate time and drift correction but are targeted for different use cases to comply with energy and duty cycle requirements.

3.1 Synchronization Request

LongShoT begins with determining the precise time labels to estimate offset and drift. Figure 4 shows a single LoRaWAN transmission cycle annotated with the expected timing delays throughout the network.

A request is initiated by the end-device by sending an uplink packet to the network server. The contents of this packet do not

matter as it does not affect the computation of time. A device simply needs to indicate to the Network Server that it is requesting for time. As such, a request can be mixed in with any telemetry packet of a device. After the radio sends the packet successfully, the end-device MCU captures time, t_{tx_done} , from the TxDone interrupt of the radio and computes t_0 based on the known radio ramp down time, T_{ramp} , as configured.

On the gateway side, the packet is received and the arrival time is timestamped in the radio hardware. This timestamp is t_1 . The network server, upon seeing that the end-device requests for time, replies with the value of t_1 as the response. t_1 is encoded as an 8-byte integer declaring time in microseconds.

The end-device then expects a response with $T_{RxDelay}$ seconds after its uplink and opens a receive window to accept the downlink packet. Upon receiving the packet, the end-device then acquires t_{rx_done} from the radio interrupt signal. With knowledge of the data rate used and number of bytes in the packet, we can then compute for decoding time, T_{decode} and packet time on air, T_{ToA} , to acquire both t_2 and t_3 based on our knowledge of t_1 and t_{rx_done} .

With availability of all required timestamps, we can then compute for offset and drift using the approaches described in the next sections.

3.2 One-Shot Synchronization

One-shot synchronization uses a single network request to adjust local clocks and perform a first-order compensation of drift rate. This approach is useful for very low duty cycle operations, where a device communicates infrequently, but may benefit from accurately compensated drift rates. However, this approach does not account for propagation time and will introduce errors proportional to distance from the reference gateway.

For one-shot synchronization, it is best to use the downlink timestamps to compute one-way offset,

$$\theta_{OneShot} = t_2 - t_3 \quad (9)$$

$\theta_{OneShot}$ is expected to represent the time offset between end-device and gateway more accurately than the uplink time measurements due to the presence of drift. However, this value includes propagation delay in the offset.

To compensate for drift, α , we rely on the predictable network roundtrip time and compare it against the device measured time delay,

$$t_3 - t_0 = \alpha \cdot t_{rtt} \quad (10)$$

such that,

$$\alpha = \frac{t_3 - t_0}{T_{rtt}} \quad (11)$$

This approach has two potential sources of error: (a) the tick resolution of the device clock in which it can measurably compare the accumulated drift over the span of time and (b) the difficulty of measuring accumulated drift against other delays of similar magnitude, such as propagation delay.

To address these, we take advantage of LoRaWAN's configurable network delay, $RxDelay$. By increasing $RxDelay$, i.e. the time between reception of an uplink packet and the transmission of a response, we allow the drift to accumulate over time, so that it is dominantly measurable compared to other delays. We expect errors

Figure 4: LoRaWAN Class A transmission cycle timing diagram annotated with the sources of delay along the network path. The process begins with an end-device initiating a Tx and timestamping the end of transmission with t_{tx_done} based on a hardware interrupt signal. The packet is received by a gateway and timestamped (t_1) in radio hardware with a 32-bit $1\mu s$ counter. The gateway host forwards the received packet to a LoRaWAN Network Server, which in turn responds with a scheduled downlink packet to be sent at $t_{send} = t_1 + T_{RxDelay}$. The gateway host queues this packet into the radio. When the radio hardware counter reaches t_{send} , it initiates the downlink transmission. Meanwhile, the end-device, knowing the expected response time, receives the packet and timestamps the end of reception with t_{rx_done} . The precise time labels (t_0 , t_2 , and t_3) can then be calculated from the measured timestamps (t_1 , t_{tx_done} , and t_{rx_done}) by applying known deterministic delays in the network path.

to be inversely proportional to $RxDelay$ resulting to higher drift compensation accuracy as we increase $RxDelay$.

3.3 Two-Step Correction

The Two-Step Correction approach uses two network requests to estimate drift, account for propagation time and determine the exact clock offset compared to the reference. This approach is appropriate for applications that expect constant synchronization and maintain accuracy within a certain bound. While it requires at least two network requests, these two requests can be spaced according to duty cycle limitations. The device is free to choose an appropriate interval as needed. The larger the interval between requests, the better LongShoT can correct long-term drift.

Following the model in Figure 5, we begin by performing the first synchronization request. From this request, we can estimate an initial offset by using the time measurements and perform a first-order correction. In this step however, we avoid performing a drift correction and maintain the natural drift rate of the device.

$$\theta_i = t_1^i - t_0^i \quad (12)$$

The device then performs a second request after $T_{interval}$ to collect a second set of time measurements. Using the measurements from the two sets, we compute drift, α as follows:

Figure 5: Two-Step Correction Model

$$\alpha = \frac{t_0^{i+1} - (t_0^i + \theta_i)}{t_1^{i+1} - t_1^i} \quad (13)$$

With an estimate for drift, we can then solve for the propagation delay by examining the total roundtrip time,

$$t_3^{i+1} = t_0^{i+1} + \alpha \cdot (T'_{rtt} + 2 \cdot T_{prop}) \quad (14)$$

where,

$$T'_{rtt} = T_{RxDelay} + T_{GwTxDelay} + T_{ToA} + T_{decode} - T_{ramp} \quad (15)$$

thus,

$$T_{prop} = \frac{t_3^{i+1} - t_0^{i+1}}{2 \cdot \alpha} - \frac{T'_{rtt}}{2} \quad (16)$$

With a proper estimate for both drift and propagation delay, we can then compute the true offset,

$$\theta_{i+1} = t_2^{i+1} - t_3^{i+1} + T_{prop} \quad (17)$$

Using Equations 13,16 and 17, we can synchronize the device clocks, correct long-term drift and estimate the wireless propagation time between the gateway and device.

The accuracy of results will depend on the accuracy at which we can determine drift. The sources of error for this approach are mainly due to long-term drift stability and jitter in measuring time. Drift stability can be affected by several factors, such as temperature and vibration. Compensation for these effects are possible but outside the scope of LongShoT.

4 EVALUATION

We evaluate LongShoT to determine the level of accuracy we can synchronize devices at long distance. In this section, we describe our experiment approach, the hardware and software elements required, followed by our target metrics. The first experiment measures the performance of using the one-shot synchronization approach. In the second experiment, we measure synchronization of devices physically distributed up to 4km from the reference gateway.

4.1 Experiment Setup

LongShoT was implemented on consumer-off-the-shelf hardware, as described in Figure 6. An ESP-WROOM-32[7] is used as the microcontroller and a Semtech SX1276[29] as the sensor radio. Each sensor includes an OriginGPS ORG1411[23] GPS Receiver that provides a 1PPS signal and GPS time. The sensor firmware uses Semtech's LoRaMac stack, modified to provide MAC-level timestamps to the application. The local clock is driven by the ESP-32's timer module configured to run at 40MHz. Internally, an ESP-WROOM-32 uses a 40MHz reference crystal oscillator with an accuracy of $\pm 10ppm$.

Figure 6: Block diagram of the LongShoT sensor implemented with an ESP-WROOM-32 and an SX1276 LoRa radio. A GPS 1PPS signal is used to sample the device's current time and establish ground truth with respect to GPS time.

For the LoRaWAN network, a Raspberry Pi 3 is used as the gateway along with a RisingHF RHF0M301 LoRa Concentrator (internally a Semtech SX1301) module. A U-Blox Neo-6M GPS module provides the time reference to both the Raspberry Pi and concentrator. The gateway software is a vanilla Semtech Packet Forwarder¹.

¹Semtech Github - https://github.com/loranet/packet_forwarder

An open-source LoRaWAN Server² is used along with a simple Node.js application server to provide responses to LongShoT requests.

In all experiments, the gateway hosts the reference clock which is assumed to be perfectly synchronized with GPS time, hence we will refer to GPS time as the system reference. To collect data with GPS as ground truth, each device samples its clock on the falling edge of the GPS 1PPS signal, which is 200ms after the rising edge. The falling edge is used to not conflict with the local timer tick interrupt. The GPS 1PPS accuracy error is measured to be $< 50ns$ for the ORG1411.[25] Each local time sample is tagged with the corresponding GPS timestamp to align samples from all devices. To optimize data transfer over LoRaWAN, only samples on each 10th second of GPS time are recorded. GPS position is also collected for each synchronization interval. All devices synchronize at 5-minute intervals, based on GPS time. All LoRaWAN messages are sent with DR0 to provide consistent behavior irrespective of distance from the gateway.

4.2 Metrics

We define our figures of merit based on the accuracy of synchronizing our device clocks with GPS time and the rates at which we can maintain synchronization. The following metrics are used to measure the performance of LongShoT:

Synchronization Error is defined as the offset of a local clock compared to GPS time measured using the first sample immediately after a synchronization request. This indicates how accurate LongShoT is able to synchronize time with each request. Due to variations in sources of network delays and physical limitations in signal bandwidth and hardware, the expected errors may rise up to a few microseconds. We thus set a goal of $\pm 10\mu s$ from the reference as a measure of acceptable synchronization error performance.

Drift Rate is the result of devices correcting their local clock rates based on LongShoT measurements. This metric is calculated from each consecutive offset measurement and averaged over the corresponding synchronization interval. Reducing drift increases the length of valid synchronization and reduces the need for frequent requests. In LongShoT, we set 1ppm as the standard for drift and aim to be well below those bounds.

4.3 One-Shot Synchronization Results

The first experiment investigates the effectiveness of One-Shot Synchronization for correcting both synchronization offset and drift.

4.3.1 Experiment Procedure. The experiment uses two devices, co-located at around 200m from the gateway. In this experiment, we vary the value of $RxDelay$ from 1s to 9s, in increments of 2 seconds. With each case, we examine the effect on synchronization offset and drift rate.

We also gather a special case where $RxDelay = 1s$ but only performing offset correction without drift correction. This provides a baseline for offset accuracy and natural device drift.

The experiment gathered two hours worth of device time data for each case tested. Devices synchronize every 5 minutes and each

²LoRa Server - <https://loraserver.io>

Figure 7: Effects of increasing RxDelay for One-Shot Synchronization. Higher synchronization accuracy and better overall drift rates are seen with increasing RxDelay. An RxDelay value of at least 5s provides satisfactory results with a single synchronization request.

synchronization request is independent of the previous. A total of 6 cases were collected, resulting in 12 hours worth of device time data.

4.3.2 Analysis. The results for this experiment can be seen in Figure 7 where (a) shows Synchronization Error and (b) shows the resulting Drift Rates. Mean and standard deviation of results can be found in Table 4.

We can see from the baseline case that simply applying classical two-way time transfer does not guarantee accurate synchronization. The baseline drift rates vary between devices but are naturally stable. This also confirms Equation 7 that synchronization error is larger with higher drift rates due to the network roundtrip time.

Case	Sync. Error (μ s)		Drift Rate (ppm)	
	Device 1	Device 2	Device 1	Device 2
Base	-2.0 ± 2.6	-37 ± 21	-1.3 ± 0.9	-4.6 ± 0.8
1s	-4.2 ± 9.4	-3.2 ± 5.9	-1.1 ± 3.4	-1.7 ± 2.7
3s	1.1 ± 1.9	1.2 ± 8.2	-0.1 ± 1.0	-0.1 ± 1.1
5s	1.4 ± 5.6	2.7 ± 4.5	-0.2 ± 0.6	0.03 ± 0.6
7s	2.3 ± 3.3	2.7 ± 2.6	-0.01 ± 0.4	0.03 ± 0.5
9s	2.9 ± 2.4	2.1 ± 4.2	0.04 ± 0.2	-0.001 ± 0.2

Table 4: One-Shot Synchronization Results

With One-Shot Synchronization active, we immediately see improvements in both synchronization error and drift rates. Synchronization error is inversely proportional to RxDelay. At RxDelay = 5s, the error is consistently within bounds of 10μ s of GPS time. This indicates 5s as a good candidate value for use with LongShoT.

Looking at Drift Rates, we see the same trend of increasing accuracy with higher RxDelays. Importantly, using low RxDelays (< 5 s), the resulting rates could be higher than the natural drift rate of the device. The lower the natural drift rate, the higher the probability of miscalculating the correction factor due to measurement errors, as seen in RxDelay = 1s. RxDelay = 5s also provides sufficient time to bring our corrected drift rates to below 1ppm. This resolves to within physical limits of ± 1 -bit quantization error with the ESP-32's internal 40MHz counter.

With One-Shot Synchronization, we achieve synchronization errors within 3μ s of GPS time and successfully correct drift rates to

below 1ppm with a single synchronization request. It is important to note that One-Shot Synchronization cannot factor in propagation delay and will thus appear as a constant offset depending on the distance of device and gateway. It is suggested to use One-Shot Synchronization for devices within close range of the gateway, or for those applications which can tolerate the deterministic errors introduced by propagation delay.

4.4 Long-Range, Two-Step Synchronization

Our second experiment investigates the performance of synchronization at long-range. In this experiment, Two-Step correction approach was used to properly synchronize while accounting for propagation delay. The experiment was conducted with three devices placed at significant distances from the reference gateway.

4.4.1 Experiment Procedure. In this experiment, we placed three devices at ranges where propagation delay would play a significant factor in time synchronization. Two devices were placed at approximately 1km from the gateway, while a third device was placed at a distance of 4km. Figure 8 shows the locations of devices on a map with respect to the gateway, along with photos of actual device placements.

All devices use RxDelay of 5s while performing offset and drift correction using the Two-Step approach. Devices are left to synchronize with the gateway at 5-minute intervals for a time span of two hours. Each device reports its GPS 1PPS sampled time and the device temperature every 10 seconds.

The collected data set can be seen in Figure 9. Measurement errors due to GPS 1PPS interrupt conflicts are filtered from the data set — these are sudden impulses in the data which result in several milliseconds of error. These errors are mainly seen in Device 2 and amount to only 1.2% of the data set.

4.4.2 Analysis. Observing the time plot in Figure 9, we can see that at all times, the offset from GPS time is within $\pm 100\mu$ s. Device 3 exhibits a few crossings of 50μ s due to significant temperature shift and increased drift over the synchronization interval. LongShoT does not take into account temperature changes for drift correction, but it is feasible to implement for such devices as suggested in [36]. Being able to account for temperature changes will allow

Device	Distance (m)	Temp. Range (C)	Synch. Error (μ s)	Drift (ppb)	Prop. Delay / Error (μ s)
Device 1	1112	34.3 - 35.9	$\mu = 1.61, \sigma = 2.41$	$\mu = 5, \sigma = 53$	$\mu = 2.6, \sigma = 1.5, \epsilon = -1.1$
Device 2	1113	34.1 - 35.3	$\mu = 1.64, \sigma = 2.35$	$\mu = -14, \sigma = 45$	$\mu = 2.4, \sigma = 1.6, \epsilon = -1.3$
Device 3	3960	30.2 - 32.8	$\mu = 1.22, \sigma = 3.74$	$\mu = -38, \sigma = 124$	$\mu = 12.1, \sigma = 2.2, \epsilon = -1.1$

Table 5: Summary of results for synchronization accuracy, drift and propagation delay for devices distributed over a wide area.

(a) Device Locations: gateway (green) and three sensors (blue, red, purple) (Map Source: OpenStreetMaps)

(b) Device 1

(c) Device 3

Figure 8: Approximate locations of gateway and devices. (a) shows the location of the gateway on one end of a runway, with two devices placed 1.1km away at the other end. A third device is placed 3.96km away from the gateway. Actual device placements are seen in (b) and (c).

LongShoT to perform slight adjustments short term drift corrections and maintain tighter synchronization between intervals. LongShoT is more suited to adjusting for long-term drift and oscillator aging.

It can be seen in Figure 10a that synchronization errors for all devices are within $\pm 10\mu$ s. This is within the bounds of our acceptable performance range. Furthermore, the mean synchronization error for all devices is 1.49μ s with a standard deviation of 2.83μ s. Most notably, we do not observe a significant offset due to propagation delay, in which Device 3 would exhibit up to 13.3μ s at 4km away from the reference. This indicates that LongShoT can compensate for propagation delay in synchronizing local device clocks.

The achieved drift rates are also remarkable as seen in Figure 10b. All device drift rates have a mean of $15ppb$ with a standard deviation of $83ppb$. These compensated rates are at the limit of our devices' 40MHz crystal oscillators. This is at least two orders of magnitude

Figure 9: Time Offset from GPS time of three devices placed at significant distances from the gateway. All devices are well-below $\pm 100\mu$ s at all times. Device temperatures swing between 30 to 35°C. It can be seen in the time plots that the drift rates of devices are affected by small changes in temperature. LongShoT attempts to correct for these changes with each request.

better than the natural drift rates exhibited by these devices. Achieving these rates mean that we can maintain synchronization longer and use fewer packets to re-synchronize. Factoring in temperature compensation will improve this further, but is outside the scope of this work.

In addition to collecting offset and drift data, the devices record the propagation delay computed with each request and is shown in Figure 11. While the distribution of results is quite high ($\sigma = 1.7\mu$ s), the mean estimates fall closely to the expected value taking into account the speed of light traveling the device distance. A mean error of -1.1μ s is observed across all devices.

The estimation of propagation delay relies directly on the stability of drift over the synchronization interval. This can easily be affected by several factors, such as temperature and vibration. While the results are not precise to enable inversely estimating the distance from a gateway, it is enough to synchronize device clocks at the microsecond level. One such approach to improve these results further involves increasing the clock resolution on the gateway, which is limited to 1μ s ticks, and allow nanosecond level timestamps. Another approach is to statistically accumulate propagation delay estimates over several independent probes and converge on the most likely value for propagation delay.

Our results show that LongShoT can accurately synchronize device clocks over significant distances with single periodic network requests. Table 5 summarizes the results of synchronization error, drift and propagation delay estimates across all three devices.

Figure 10: Results of Synchronization at Long-Range. All 3 devices perform comparatively well despite varying distance from the gateway. The higher drift distribution for Device 3 is due to temperature shifts at its location.

Figure 11: Propagation delay as estimated by end-devices using two-step synchronization requests. The estimated delays show an average error of $-1.1\mu s$ with respect to the expected delay given distance traveled over the speed of light.

4.4.3 Synchronization Performance. Finally, we observe how well our devices remain synchronized over the span of time of the experiment through the use of LongShoT. Figure 12 shows the distribution of all reported time offsets collected over a two-hour period. This plot shows that synchronizing at a periodic 5-minute interval keeps our devices synchronized to within $\pm 100\mu s$ of our reference time at all times.

Looking closely, Device 1 stays within $\pm 50\mu s$ of the reference at all times, while Devices 2 and 3 hold within $\pm 50\mu s$ for 96% of

the total time. Depending on the application, the synchronization interval can be adjusted if the application requires tighter synchronization at the expense of more energy, or can be relaxed to reduce the amount of energy used overall. Table 6 summarizes the synchronization performance of all devices over the entire experiment duration.

Figure 12: Distribution of device offsets for three devices over 2 hours. Devices maintain synchronization to within $\pm 100\mu s$ at all times, with 97.5% of all offsets falling within $\pm 50\mu s$ of the reference.

Device	Mean Abs. Offset (μs)	Offset Within $50\mu s$
Device 1	$\mu = 6.6, \sigma = 6.8$	100%
Device 2	$\mu = 6.3, \sigma = 6.0$	96.4%
Device 3	$\mu = 16.4, \sigma = 16.8$	96.1%
All	$\mu = 9.8, \sigma = 12.0$	97.5%

Table 6: Synchronization performance summary showing the mean absolute offset of device clocks with respect to GPS time over the duration of the experiment.

5 LONGSHOT AND GPS ENERGY

Synchronizing time over a wide area is typically done using on-board GPS receivers. While GPS receivers provide synchronization accuracy to within $100ns$ [34], it requires at least 10X more energy than synchronizing time with LongShoT. In this section, we compare the energy usage of performing LongShoT requests versus using a GPS receiver to synchronize time over long distances. For a fair comparison, we evaluate the energy required for one synchronization attempt, or Energy Per Sync (EPS), on consumer grade devices.

5.1 LongShoT Request Energy

The energy required to perform a LongShoT transaction is equal to the energy consumed by one LoRaWAN transaction. The energy efficiency of LoRaWAN packet transactions have been studied in [19]. Assuming LongShoT operates on a fixed synchronization interval, the EPS for LongShoT can be calculated given Equation 18,

$$EPS_{LongShoT} = V * [(I_{tx} * T_{tx}) + (I_{rx} * T_{rx}) + (I_{idle} * T_{RxDelay}) + (I_{sleep} * T_{sleep})] \quad (18)$$

where, the total energy is the sum of the energy consumed with each state of the LoRaWAN transaction (Tx, Idle, Rx, Sleep). Table 7

provides the typical values for a single LongShoT request performed in the US915 band specification and as used in the evaluations in Section 4.

Using the maximum transmit power of 20dBm and an RxDelay of 5s, the energy per sync is roughly 190mJ.

Parameter	Conditions	Value
V	Input Voltage	3.3V
I_{tx}	$RFOutPower = 20dBm$	120mA
I_{rx}	$BW = 500kHz$	12.6mA
I_{idle}	-	1.8mA
I_{sleep}	-	1 μ A
T_{tx}	4 bytes @ $SF = 10$, $BW = 125kHz$	329.728ms
T_{rx}	8 bytes @ $SF = 10$, $BW = 500kHz$	92.672ms
T_{idle}	Based on RxDelay	RxDelay
T_{sleep}	Sync Interval	Varies

Table 7: LoRaWAN Energy Parameters based on an SX1276 transmitting at full power in the US915 band. [28, 29]

5.2 GPS Energy Comparison

GPS receivers are capable of providing time as soon as it can establish a fix in location. For consumer-grade GPS receivers, the accuracy of the first fix may be off by several meters[12]. This translates to $< 100ns$ of error, but is sufficient enough to compare against LongShoT's sub-10 μ s accuracy. To quantify the energy usage of synchronizing time with GPS, we consider the time-to-first-fix (T_{tff}) given different startup states of a GPS receiver.

The time-to-first-fix typically depends on the previous state held by the GPS receiver and the validity of the satellite ephemeris data in memory. Within these conditions, a GPS receiver may perform the following on start up sequences:

- *Hot Start* - the receiver has a valid last-known position and valid ephemeris data.
- *Warm Start* - the receiver has a valid last-known position but the ephemeris data has expired.
- *Cold Start* - both position and ephemeris data are invalid, typically on first power up.

Satellite ephemeris data are updated every 30 minutes and takes at least 30s to acquire with full-power acquisition. On each cold start and each subsequent 30 minutes, a GPS receiver must expend energy to re-acquire the ephemeris. Table 8 shows the typical energy consumed to start up a GPS receiver and waiting for the first fix.

Mode	T_{tff}	Power	Energy Per Sync
Cold Start	35s	77mW	2695mJ
Warm Start	32s	67mW	2144mJ
Hot Start	2s	67mW	134mJ

Table 8: OriginGPS Nano-Hornet ORG1411 Typical Start up Mode Energy Consumption @ 1.8V [23]

From Table 8, we can observe that performing a Cold or Warm Start is at least 10X more costly than a single LongShoT request. While additional Hot Starts provide low-latency and low-energy

cost time fixes, it requires ephemeris downloads every 30 minutes. Figure 13 compares the total energy consumed by LongShoT and GPS with an increasing interval between consecutive requests. Based from this projection, LongShoT uses less energy than GPS for intervals longer than 50s between synchronization requests.

Figure 13: Total energy consumed for time synchronization by synchronization interval. LongShoT consumes less energy than GPS for intervals greater than 50s.

6 RELATED WORK

Time synchronization in wireless sensor networks has been studied in depth. Synchronization is typically achieved by message passing and regression of multiple time measurements across the message exchange. Reference Broadcast Synchronization (RBS)[6] uses a broadcast message from a reference node and the reception time on each node as a marker for time across the system of devices. Similarly, the LoRaWAN standard[16] specifies a mode (Class B) for receiving broadcast messages from gateways to synchronize nodes.

TPSN [9] uses a two-way message passing scheme across pairwise nodes to factor in delays incurred in the network stack. It however assumes that clock drift during message exchange is negligible and would amount to significant errors in the timescales of LoRaWAN.

Flooding Time Synchronization Protocol (FTSP) [18], Pulse-Sync [13] and Glossy [8] synchronize time across sizeable, multi-hop networks by utilizing network flooding to achieve accuracy. Furthermore, propagation delay has been shown to accumulate with each hop along the network. TATS [14] improves on Glossy and Pulse-Sync to factor propagation delay and increase the synchronization accuracy across the network of nodes.

In contrast to LongShoT, these approaches do not apply to LoRaWAN due to bandwidth limitations and the fundamental difference in network topology. LongShoT operates with the range achievable by LoRaWAN, factoring in propagation delay and the compounded effects due to clock drift.

Furthermore, the use of a mesh topology or broadcast packets increases the energy required to synchronize time, either by keeping the device awake to receive or transmitting more packets than needed. LongShoT enables time synchronization on-demand allowing a device to remain in deep-sleep and maintain a wake-sync-measure-sleep routine as necessary for ultra low-power operation.

Other methods such as Stitch[17], similarly, uses narrow-band transceivers to perform frequency syntonization of local oscillators. While Stitch does not provide phase synchronization, its ease of syntonization can benefit LongShoT in addressing drift between two oscillators and thus lowering energy cost further.

7 CONCLUSION

In this paper, we presented LongShoT, a time synchronization scheme built on LoRaWAN that allows device synchronization to within $10\mu\text{s}$ of a common clock at both long-range and low-power through an in-depth timing analysis of LoRaWAN and its hardware. We implemented LongShoT on off-the-shelf components and in our experiments show that LongShoT can achieve an average synchronization error of less than $2\mu\text{s}$ and it can maintain synchronization to within $50\mu\text{s}$ of the reference time with a single packet synchronization request every 5 minutes.

We have shown that accurate synchronization and drift correction may be achieved with a single packet by taking advantage of LoRaWAN's configurable network delay. This allows a simple wake-sync-measure-sleep pattern which can be performed at the application's discretion and support ultra low-power consumption for battery-powered devices. Furthermore, LongShoT can synchronize devices up to 4km away from a LoRaWAN gateway. Synchronization at such distances was previously only achieved by having GPS timing receivers on sensor devices, but LongShoT unlocks this capability for devices incapable of supporting the energy requirements of a sustained, active GPS receiver.

LongShoT is capable of achieving accurate results with simplistic approaches, in particular, by relying on predictable timing behaviors of the radio hardware. Following IEEE-1588 and Stitch, it would be beneficial for the network interface to manage the clock and distribute time to the rest of the system. Similarly, LongShoT demonstrates the same capability for LPWANs and recommend that hardware manufacturers consider building this functionality in.

REFERENCES

- [1] Ferran Adelantado, Xavier Vilajosana, Pere Tuset-Peiro, Borja Martinez, Joan Melia-Segui, and Thomas Watteyne. 2017. Understanding the Limits of LoRaWAN. *IEEE Communications Magazine* 55, 9 (2017), 34–40. <https://doi.org/10.1109/MCOM.2017.1600613>
- [2] Marco Centenaro, Lorenzo Vangelista, Andrea Zanella, and Michele Zorzi. 2016. Long-range communications in unlicensed bands: The rising stars in the IoT and smart city scenarios. *IEEE Wireless Communications* 23, 5 (2016), 60–67.
- [3] Chiara Maria De Dominicis, Student Member, Paolo Pivato, Paolo Ferrari, David Macii, Emiliano Sisinni, and Alessandra Flammini. 2013. Timestamping of IEEE 802.15. 4a CSS signals for wireless ranging and time synchronization. *IEEE Transactions on Instrumentation and Measurement* 62, 8 (2013), 2286–2296. <https://doi.org/10.1109/TIM.2013.2255988>
- [4] A. Dongare, C. Hesling, K. Bhatia, A. Balanuta, R.L. Pereira, B. Iannucci, and A. Rowe. 2017. OpenChirp: A Low-Power Wide-Area Networking architecture. In *2017 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops 2017*. <https://doi.org/10.1109/PERCOMW.2017.7917625>
- [5] John Eidson. 2005. IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)* (2005), 1–269. <https://doi.org/10.1109/IEEESTD.2008.4579760>
- [6] Jeremy Elson, Lewis Girod, and Deborah Estrin. 2002. Fine-grained Network Time Synchronization Using Reference Broadcasts. *SIGOPS Oper. Syst. Rev.* 36, SI (12 2002), 147–163. <https://doi.org/10.1145/844128.844143>
- [7] Espressif Systems. 2018. ESP32 Datasheet. (2018), 58.
- [8] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. 2011. Efficient network flooding and time synchronization with Glossy - IEEE Xplore Document. *Information Processing in Sensor Networks (IPSN), 2011 10th International Conference on* (2011), 73–84. <http://ieeexplore.ieee.org/document/5779066/?arnumber=5779066>
- [9] Saurabh Ganeriwal, Ram Kumar, and Mani B. Srivastava. 2003. Timing-sync Protocol for Sensor Networks. In *Proceedings of the 1st International Conference on Embedded Networked Sensor Systems (SenSys '03)*. ACM, New York, NY, USA, 138–149. <https://doi.org/10.1145/958491.958508>
- [10] Bob Iannucci and Anthony Rowe. 2017. Crowdsourced Smart Cities. *Intelligent Transportation Systems (ITS) World Congress* (2017).
- [11] Dieter Kirchner. 1991. Two-Way Time Transfer Via Communication Satellites. *Proc. IEEE* 79, 7 (1991), 983–990. <https://doi.org/10.1109/5.84975>
- [12] Mikko Lehtinen, Ari Happonen, and Jouni Ikonen. [n. d.]. Accuracy and time to first fix using consumer-grade GPS receivers. ([n. d.]).
- [13] Christoph Lenzen, Philipp Sommer, and Roger Wattenhofer. 2015. PulseSync: An efficient and scalable clock synchronization protocol. *IEEE/ACM Transactions on Networking* 23, 3 (2015), 717–757. <https://doi.org/10.1109/TNET.2014.2309805>
- [14] Roman Lim, Balz Maag, and Lothar Thiele. 2016. Time-of-Flight Aware Time Synchronization for Wireless Embedded Systems. *International Conference on Embedded Wireless Systems and Networks* (2016), 149–158.
- [15] LoRa-Alliance. 2015. A technical overview of LoRa and LoRaWAN. November (2015), 1–20. <https://doi.org/10.1038/scientificamerican1013-22>
- [16] LoRa Alliance. 2018. LoRaWAN 1.0.3 specification. *LoRa-Alliance.Org* 1 [Online]. Accessible: <https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf> (2018), 1–72. <https://lora-alliance.org/sites/default/files/2018-07/lorawan1.0.3.pdf>
- [17] Anh Luong, Peter Hillyard, Alemayehu Solomon Abrar, Charissa Che, Anthony Rowe, Thomas Schmid, and Neal Patwari. 2018. A Stitch in Time and Frequency Synchronization. *2018 17th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)* (2018), 96–107. <https://doi.org/10.1109/IPSN.2018.00016>
- [18] M Maróti, B Kusy, G Simon, and ÁÁ Lédéczi. 2004. The flooding time synchronization protocol. *SenSys'04 - Proceedings of the Second International Conference on Embedded Networked Sensor Systems* (2004), 39–49. <https://doi.org/10.1145/1031495.1031501>
- [19] Samarth Mathur, Anand Sankar, Prajwal Prasan, and Bob Iannucci. 2017. Energy Analysis of LoRaWAN Technology for Traffic Sensing Applications. *Intelligent Transportation Systems (ITS) World Congress* (2017).
- [20] Yasir Mehmood, Farhan Ahmad, Ibrar Yaqoob, Asma Adnane, Muhammad Imran, and Sghaier Guizani. 2017. Internet-of-Things-Based Smart Cities: Recent Advances and Challenges. *IEEE Communications Magazine* 55, 9 (2017), 16–24. <https://doi.org/10.1109/MCOM.2017.1600514>
- [21] David L. Mills. 1991. Internet Time Synchronization: The Network Time Protocol. *IEEE Transactions on Communications* 39, 10 (1991), 1482–1493. <https://doi.org/10.1109/26.103043>
- [22] Umer Noreen, Ahcene Bounceur, and Laurent Clavier. 2017. A study of LoRa low power and wide area network technology. *2017 International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)* (2017), 1–6. <https://doi.org/10.1109/ATSIP.2017.8075570>
- [23] OriginGPS. 2014. Nano Hornet (ORG1411) GPS Antenna Module. (2014).
- [24] Kristofer S. J. Pister and Lance Doherty. 2008. TSMF: Time synchronized mesh protocol. *Pdcs JANUARY* 2008 (2008), 391–398.
- [25] Inc. Qualcomm. 2015. SiRFstar GSD4e 1PPS Timing Accuracy. (2015).
- [26] Mattia Rizzi, Paolo Ferrari, Alessandra Flammini, and Emiliano Sisinni. 2017. Evaluation of the IoT LoRaWAN Solution for Distributed Measurement Applications. *IEEE Transactions on Instrumentation and Measurement* 66, 12 (2017), 3340–3349. <https://doi.org/10.1109/TIM.2017.2746378>
- [27] H. G. Schroder Filho, J. Pissolato Filho, and V. L. Moreli. 2016. The adequacy of LoRaWAN on smart grids: A comparison with RF mesh technology. *IEEE 2nd International Smart Cities Conference: Improving the Citizens Quality of Life, ISC2 2016 - Proceedings* (2016). <https://doi.org/10.1109/ISC2.2016.7580783>
- [28] Semtech. 2013. LoRa Modem Design Guide. July (2013), 1–9.
- [29] Semtech. 2016. SX1276/77/78/79 Datasheet. August (2016).
- [30] Semtech. 2016. SX1301 Datasheet. (2016), 1–63.
- [31] Gyula Simon, Miklós Maróti, Ákos Lédéczi, György Balogh, Branislav Kusy, András Nádas, Gábor Pap, János Sallai, and Ken Frampton. 2004. Sensor Network-based Countersniper System. In *Proceedings of the 2Nd International Conference on Embedded Networked Sensor Systems (SenSys '04)*. ACM, New York, NY, USA, 1–12. <https://doi.org/10.1145/1031495.1031497>
- [32] Fikret Sivrikaya and Bulent Yener. 2004. Time Synchronization in Sensor Networks: A Survey. (2004).
- [33] Sanjib Sur, Teng Wei, and Xinyu Zhang. 2014. Autodirective audio capturing through a synchronized smartphone array. *Mobisys* (2014), 28–41. <https://doi.org/10.1145/2594368.2594380>
- [34] Marc Weiss. 2014. Accurate Time and Frequency Transfer During Common-View of a GPS Satellite. February 1980 (2014). <https://doi.org/10.1109/FREQ.1980.200424>
- [35] Andrew J. Wixted, Peter Kinnaird, Hadi Larjani, Alan Tait, Ali Ahmadiania, and Niall Strachan. 2017. Evaluation of LoRa and LoRaWAN for wireless sensor networks. *Proceedings of IEEE Sensors 0* (2017), 5–7. <https://doi.org/10.1109/ICSENS.2016.7808712>
- [36] Miao Xu, Wenyan Xu, Tingrui Han, and Zhiyun Lin. 2016. Energy-Efficient Time Synchronization in Wireless Sensor Networks via Temperature-Aware Compensation. *ACM Transactions on Sensor Networks* 12, 2 (2016), 1–29. <https://doi.org/10.1145/2876508>