RFClock: Timing, Phase and Frequency Synchronization for Distributed Wireless Networks

Kubra Alemdar, Divashrey Varshey, Subhramoy Mohanti, Ufuk Muncuk, Kaushik Chowdhury
Institute for the Wireless Internet of Things, Northeastern University
{alemdar.k,varshey.d,mohanti.s,u.muncuk,k.chowdhury}@northeastern.edu

ABSTRACT

Emerging applications like distributed coordinated beamforming (DCB), intelligent reflector arrays, and networked robotic devices will transform wireless applications. However, for systems-centric work on these topics, the research community must first overcome the hurdle of implementing fine-grained, over-the-air timing synchronization, which is critical for any coordinated operation. To address this gap, this paper presents an open-source design and implementation of 'RFClock' that provides timing, frequency and phase synchronization for software defined radios (SDRs). It shows how RFClock can be used for a practical, 5-node DCB application without modifying existing physical/link layer protocols. By utilizing a leader-follower architecture, RFClock-leader allows follower clocks to synchronize with mean offset under 0.107Hz, and then corrects the time/phase alignment to be within a 5ns deviation. RFClock is designed to operate in generalized environments: as standalone unit, it generates a 10MHz/1PPS signal reference suitable for most commercial-off-the-shelf (COTS) SDRs today; it does not require custom protocol-specific headers or messaging; and it is robust to interference through a frequency-agile operation. Using RFClock for DCB, we verify significant increase in channel gain and low BER in a range of $[0 - 10^{-3}]$ for different modulation schemes. We also demonstrate performance that is similar to a popular wired solution and significant improvement over a GPS-based solution, while delivering this functionality at a fractional price/power point.

CCS CONCEPTS

• **Hardware** → **Integrated circuits**; • **Computer systems organization** → **Real-time system architecture**.

ACM Reference Format:
Kubra Alemdar, Divashrey Varshey, Subhramoy Mohanti, Ufuk Muncuk, Kaushik Chowdhury. 2021. RFClock: Timing, Phase and Frequency Synchronization for Distributed Wireless Networks . In *The 27th Annual International Conference on Mobile Computing and Networking (ACM MobiCom '21), October 25–29, 2021, New Orleans, LA, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3447993.3448623

1 INTRODUCTION

Wireless network architectures are undergoing a radial transformation, moving away from centralized control towards a distributed paradigm where devices make local decisions towards a shared, global objective. For example, network densification in 5G involves thousands of small cell base stations operating in close proximity for an anticipated 1000x improvement in throughput [42]; intelligent reflector arrays have large numbers of low cost antennas to create smart surfaces [3, 23, 27, 40, 43]; distributed coordinated beamforming (DCB) enables a number of radios to synchronize phase offsets and start times *exactly* to beamform towards a target receiver [4, 5, 37, 45]. However, from a system viewpoint, many of these applications are yet to realize their full potential, as devices remain shackled to a centralized clock. To date, there is no open source, physical layer solution that can provide the 10MHz reference with a 1 pulse per second (PPS) signal required for SDR-based experimentation through the wireless medium. This work proposes the design and implementation of RFClock that achieves *both* frequency and time reference, *without* modifying existing physical/link layer protocols. We will open source design files for RFClock to equip the community with an important tool for future systems-focused work on fully distributed wireless architectures.

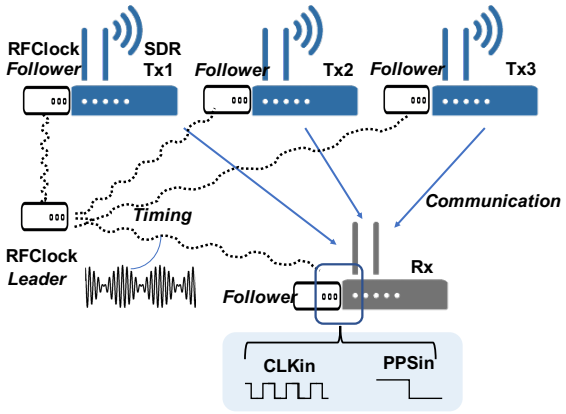


Figure 1: Network architecture showing the distributed timing enabled by RFClock.

1.1 Problem

As shown in Fig. 1, RFClock follows the *leader-follower* model, with the leader generating the reference clock that is distributed to all followers. RFClock is designed to provide (p1) carrier frequency synchronization that overcomes clock frequency offsets and locks each device to the same reference frequency, (p2) timing synchronization so that each device can perform the desired action at coordinated intervals, such as the rising/falling edge of the clock, and (p3) carrier phase synchronization, so that clock signal arrives with the same phase for all followers.

• *Carrier Frequency Synchronization (p1):* Active wireless devices forming a link derive their carrier frequency from their own local

ACM MobiCom '21, October 25–29, 2021, New Orleans, LA, USA

ACM ISBN 978-1-4503-8342-4/21/10…$15.00
https://doi.org/10.1145/3447993.3448623

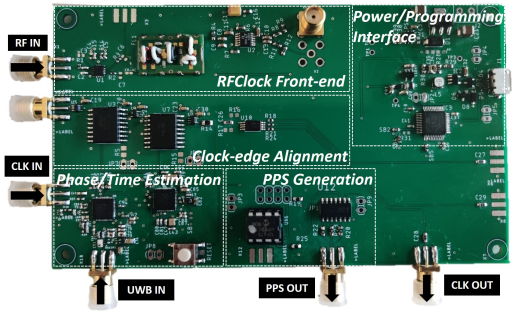


Figure 2: RFClock follower implementation with custom designed components. The board dimensions are 13cm x 7.5cm. The total weight is 35g.

oscillators (LO). Any drift in the LO results in a carrier frequency offset (CFO). The direct outcome of CFO is increased inter-symbol interference, or worse, the entire signal of interest can be filtered out by the front end if it does not fall in the desired frequency range.

• *Time Synchronization (p2):* In a distributed system, signal copies arrive at the receive antenna from different transmit antennas. These arriving signals need to be aligned on a per-symbol basis. This imposes strict timing constraints, with permissible deviations in the order of only few nanoseconds.

• *Carrier Phase Synchronization (p3):* Each emitted signal adds up constructively at the receiver. For optimal constructive effect, the received phases of the signals from individual transmitters must also be aligned at the receiver. Since transmitters are at different distances with respect to the receiver, this phase adjustment must be performed on a per-transmitter basis.

1.2 Limitations of Existing Solutions

The wired Ettus Octoclock [15] or a GPS disciplined reference solves (p1-p3) by providing separate inputs for the 10 MHz carrier and PPS rising/falling edge signals that aid in time and phase synchronization. Although the Octoclock limits the separation between antennas due to the requirement of direct physical connection, we use it at a benchmark: one of the design goals of RFClock is to perform as close as possible to the Octoclock. COTS GPS clocks, such as GPSDO [14], cost over $1K USD per unit, do not work as well in indoor environments, and suffer from intermittent link outages with satellites. As we discuss in depth in Sec. 2, the seminal work AirShare [1] comes closest to RFClock. AirShare solves (p1 and p3) and relies on a software solution SourceSync [38] for (p2). SourceSync requires a specific method of beamforming with customized wait times to achieve symbol level timing synchronization. Mega-MIMO [39] tackles (p1) and (p3), but also uses a custom MIMO approach for synchronizing the phase of distributed transmitters. Similarly, AirSync [2] requires continuous RF carrier tracking and compensation for phase rotation during ongoing application to solve (p1-p3). Pulsar [12] solves (p2), but requires an atomic clock source. To the best of our knowledge, no prior work addresses all the (p1-p3) concerns without dependency on a specific MAC layer/application.

1.3 RFClock Design Overview

RFClock has three functional blocks:

(1) Low power front-end design: RFClock *leader* transmits a two tone frequency signal at f_1 and f_2 over the air, separated by the desired input clock frequency (typically, 10 MHz), similar to AirShare [1]. However, different from AirShare, the RFClock *follower* extracts the envelop of the transmitted signal and passes it through a customized filtering process to obtain the reference clock. Thus, all nodes have the same LO drift, as they are locked to a common reference, and do not lose synchronization even if there is a frequency drift in the leader's clock. At the follower, RFClock's envelope detector measures the *beat* frequencies $f_2 - f_1$ and $f_2 + f_1$. The difference frequency $f_2 - f_1$ drives the virtual LO of 10MHz, and in turn, the receiver's phased locked loop (PLL). RFClock's front-end extracts the 10MHz signal with ultra-low power, passive, off-the-shelf components, consuming only 6.6µW.

(2) Interference-mitigating clock distribution: In practical interference conditions, the PLL may lose its lock with the reference. For lengthy interruptions, RFClock has a frequency-agile architecture that allows us to dynamically select f_1 and f_2, to avoid the spectrum prone to interference. It allows for optimizing the matching filter of the RF front-end, which ensures extraction of the 10MHz/1PPS reference remains uninterrupted. For minor interruptions, RFClock includes a holdover circuit that stores up to 120 seconds of historical frequency data, which is then extrapolated to obtain the clock signal. We show that RFClock is resilient in multipath scenarios where the follower may receive multiple delayed versions of the signal. Furthermore, in unstable environments, unpredictable phase changes introduce jitter at the clock edges. RFClock mitigates this problem by setting the optimized value of the digital loop filter bandwidth of the phased locked loop (PLL) to carefully tradeoff signal fidelity with phase noise.

(3) Accurate time/phase estimation: In addition to the 10MHz reference, each radio requires a PPS signal to perform processing tasks at the same time. However, even if all the devices have their LO driven by the reference clock frequency, there can still be phase difference between clock edges. Thus, any time offset between PPS edges for individual RFClock followers needs to be compensated. RFClock includes a clock alignment algorithm and an auxiliary correction mechanism to increase resiliency, which selects inputs from a cheap, off-the shelf GPS module costing around $35 and/or ultra-wide band (UWB) module. Whenever GPS signal is available, RFClock followers correct their individual time offsets with respect to this global PPS reference. In GPS denied environments, RFClock receivers use UWB ranging to produce high-resolution timestamps (with pico-second precision) and estimate phase offset with respect to the RFClock leader. This eliminates explicit pair-wise messaging.

1.4 Summary of Outcomes

- We design and implement RFClock that achieves tight frequency, phase and time synchronization required for distributed wireless applications. Power consumption is in the range of $[170 - 390]mW$, 70% lower than some state-of-the-art solutions like GPSDO, and costs $91.
- We implement the complete RFClock leader-follower design (the follower board is shown in Fig. 2), and compare its performance with the COTS wired Ettus Octoclock and GPS-based systems. We observe that RFClock performs as well as the Octoclock, with less than 5 nano-second level time deviation and operates in the 95 percentile for 0.21Hz and 0.93Hz frequency offset at 915MHz and 2.4GHz, respectively.

Prior Work	Sync Type	HW/SW	Synchronization Accuracy	Modify APP/MAC	Application
RBS[13]	Time	SW	µs level	Yes	-
TPSN[20]	Time	SW	µs level	Yes	-
SourceSync[38]	Time, Frequency, Phase	SW	[5 − 20]ns when 25dB > SNR > 5dB	Yes	Opportunistic routing
AirSync[2]	Time, Frequency, Phase	SW	phase misalignment < 0.078rad time: within CP of OFDM (0.8-3.2µs)	Yes	Dist. MU-MIMO
MegaMIMO[39]	Frequency, Phase	SW	phase misalignment < 0.05rad	Yes	Dist. MU-MIMO
AirShare[1]	Frequency, Phase	HW	median: 0.11Hz/<0.005rad @900MHz and 0.4Hz/<0.016rad @2.4GHz	Yes[1]	Dist. MIMO, Dist. rate adaptation
PULSAR[12]	Time	HW	<5ns	No	-
Vidyut[46]	Time, Frequency, Phase	HW	mean:225ns phase misalignment <0.0218rad	No	OFDMA, MIMO
RFClock	**Time, Frequency, Phase**	**HW**	**median: 0.097Hz @915MHz** **< 5ns**	**No**	**Dist. SU-MISO**

Table 1: Comparison of different wireless synchronization methods.

- We demonstrate how RFClock can operate flexibly in GPS-enabled and GPS-denied environments using a selection of GPS and UWB, and in presence of rich multipath indoor/outdoor settings.
- We integrate RFClock with Ettus B210 SDRs for a 5-node DCB setup, wherein four transmitting SDRs act as a virtual antenna phased array with coherent signal combination at the receiver. We verify the expected increase in channel gain. Moreover, the resulting beamforming shows Bit Error Rate (BER) probability close to 10^{-6} for BPSK and QPSK modulation schemes in moderate SNR regime.

In the remainder of the paper, Sec. 2 distinguishes our proposed approach from existing solutions. Sec. 3 derives the clock model and validates it in an experimental setup. In Sec. 4 and 5, we present the design elements of RFClock in detail. In Sec. 6, we describe the implementation of RFClock and present performance evaluation results covering time, phase and frequency synchronization. We show how RFClock enables DCB application in Sec. 7. Finally, we conclude in Sec. 8.

2 RELATED WORK

We provide a summary of related work in Table 1. Only author-reported values are included for comparison.

•**Wired and COTS Alternatives:** A wired connection between the reference source, like the Ettus Octoclock [15], and deployed devices is the most straightforward way to eliminate frequency and phase offsets. However, because the length of the cable determines the phase of the received clock signal, cable inputs to each device should have matched conductive properties and lengths. Although distributed transmitters should not be constrained by fixed wire-lengths, we use the Octoclock as the baseline for comparison with RFClock in Sec. 6 and Sec. 7.1. Highly stable oscillators such as GPS-disciplined oscillators [14] (GPSDO), oven-controlled oscillators (OCXO) [32] and chip scale atomic clock (CSAC) [31] can potentially minimize frequency offsets. However, these are expensive solutions with high power consumption of around 1W. In addition, GPSDO requires line-of-sight to satellites, which makes it applicable only for outdoors. On the contrary, RFClock bill of materials costs $91USD, with 70% lower power consumption than the GPSDO. Moreover, RFClock is resilient to multipath and can operate in both NLOS outdoor and indoor scenarios. The WWVB atomic clock broadcast from National Institute of Standards and Technology (NIST) [36] can synchronize receivers in the order of seconds, but this is too coarse for many PHY-layer operations like DCB. NIST also has an optical method that can synchronize clocks to within one femtosecond across a 4 km free space link, but this requires LOS [9].

•**Synchronization through Message Exchange:** Classical approaches developed for wireline solutions like Network Time Protocol (NTP) [33] can achieve millisecond level of accuracy. Precision Time Protocol (PTP) [22] is similar to NTP but reaches sub-microsecond level performance. It uses hardware-generated timestamps to estimate propagation time of signals and can achieve time synchronization in a wired network accurate to 25ns. White Rabbit [28] gives sub-nanosecond accuracy over optical fibers by integrating packet-based synchronization used by the PTP with Synchronous Ethernet [17]. Reference Broadcast Synchronization (RBS) [13] uses inter-node timestamp exchange to compensate for transmission delays to achieve sub-µs accuracy while The timing sync protocol for sensor networks (TPSN) [20] achieves microsecond level accuracy. However, both RBS and TPSN assume that time of flight is negligible and do not account for clock drift. As clock skew increases over time, they require frequent re-synchronization which increases energy consumption and bandwidth usage. Flooding-Time Synchronization Protocol (FTSP), Glossy, and PulseSync address the problem of time synchronization by constructive interference through controlled flooding [18, 26, 29]. However, PulseSync and Glossy are topology dependent and do not consider channel effects of interference and possible packet losses.

•**Protocol-dependent Synchronization:** SourceSync [38] harnesses sender diversity through a specially constructed synchronization header. While this approach can achieve better than 20 ns accuracy, it imposes constraints on the application or underlying MAC protocol. For synchronization, SourceSync includes header fields before the payload, which comprises of the 802.11 legacy preamble (80µs), followed by a channel estimation field (25.6µs), flag ID (25.6µs), SIFS (10µs) and ends the header portion with the co-sender

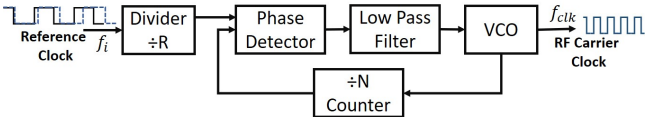


Figure 3: RF carrier generation.

channel estimation field (25.6μ)s, resulting in 166.8us of total overhead. Now, to achieve synchronization accuracy of less than 20ns, SourceSync replaces the data in each packet with 200 repetitions of the initial header, which results in a total incurred overhead induced delay of 33.36ms before the payload can be transmitted. While this method can be used to evaluate the extent of synchronization error, the ensuing overhead limits its use in real-world scenarios. Also, before transmitting, the transmitters have additional wait time, calculated from the propagation delay from the lead transmitter to the receiver. This wait time is compounded when the senders change their original location during the transmission, such as in the case of mobile nodes. Finally, SourceSync relies on opportunistic channel access of the standard 802.11 protocol, and this may introduce additional delays in high density scenarios. Different from SourceSync's software-based synchronization approach, RFClock's hardware-based synchronization makes it protocol-independent. Notable works like AirSync [2] and MegaMIMO [39] require similar modifications. AirSync enables distributed MU-MIMO using the cyclic prefix of OFDM symbol. It achieves time, phase synchronization and carrier phase coherence with a synchronization accuracy of 2.35 degrees and the 95th percentile of the synchronization error is at most 4.5 degrees (0.078rad). MegaMIMO reduces the 95th percentile phase misalignment to 2.86 degrees (0.05rad).

•**Specialized Synchronization Hardware:** AirShare [1] enables multiple nodes to share a reference clock by minimizing CFO across devices as a hardware solution. However, it delegates the task of time synchronization to SourceSync, which requires a specialized MAC protocol as discussed above. AirShare utilizes multiple non-linear components, such as LNA, power splitter and mixer to extract the reference clock, which increase the system noise figure and harmonic distortion. Each non-linear component contributes second-order harmonics of the extracted reference signal (i.e. $2(f_1 - f_2)$, $3(f_1 - f_2)$), inducing jitter within the clock signal and increasing the clock offset. We implemented AirShare architecture with the off-the-shelf components reported in [1] and observed less than −29dBc in second-order harmonics in comparison less than −37dBc with RFClock, resulting in larger even harmonics when converting 10MHz reference signal to square wave clock signal. Moreover, AirShare requires higher power (in the range of mW) due to the LNA in its receiver design. On the other hand, RFClock front-end consumes only 6.6μW power, as it has an input impedance matching network followed by a passive envelope detector to reduce complexity and power consumption. RFClock provides 150ft coverage range in the easily accessible 900MHz band (experimentally validated), almost equal to what AirShare achieves (reported theoretical distance is 210ft at 170-180MHz), without utilizing any active amplification in the front-end chain, and slightly better CFO accuracy (see Table 1). PULSAR [12] is a wireless hardware platform that achieves an accuracy of 5ns for GPS denied devices. It requires a tree-like time distribution network where clock synchronization errors accumulate per hop. It also relies on expensive atomic clocks, each of which costs over $5K. In contrast, RFClock uses off-the-shelf components and errors do not accumulate as receivers extract the clock from a reference signal continuously.

[1]AirShare employs SourceSync for time synchronization that constrains the MAC layer

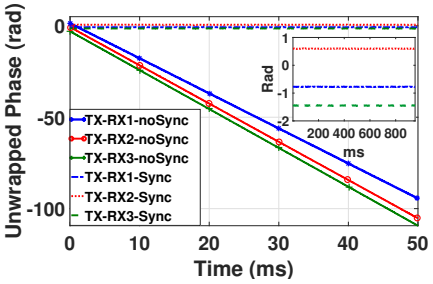


Figure 4: Instantaneous unwrapped phase of unmodulated signal received at different receiver SDRs

•**Infrastructure-dependent Synchronization:** Vidyut [46] exploits the power line infrastructure to achieve time synchronization within 450ns with a mean of approximately 225ns. Finally, [35] is a hybrid synchronization method that leverages WLAN infrastructure to reach sub-microsecond level network synchronization. This proposed peer-level synchronization between access-points assumes that the message transmission delay is negligible for 1 hop.

3 MOTIVATION

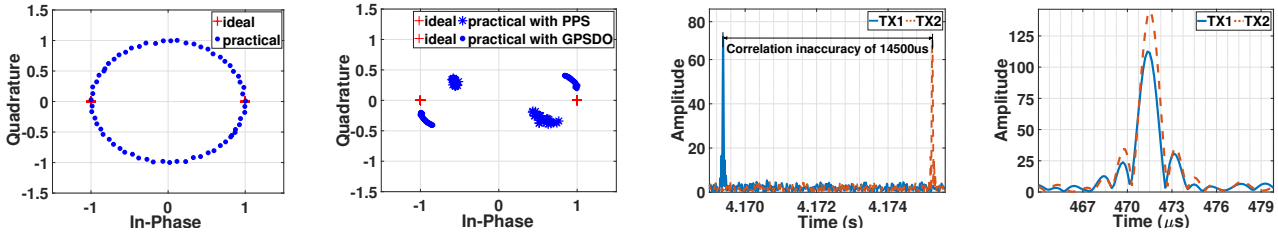
Practical clock oscillators exhibit deviations from their nominal frequency of operation due to imperfections in the manufacturing process, variations in supply voltage, and ambient temperature. Using variables ϕ and f to denote phase and frequency, respectively, the relative frequency offset $\Delta f_{ij} = (f_i - f_j)$ is a simple function of oscillator frequency in the two radios i and j, with angular frequency $\omega_{ij} = 2\pi(f_i - f_j)$. The instantaneous phase relationship between clocks for these two different radios can now be expressed as:

$$\phi_j(t) = \Delta\theta_{ij} + \Delta\omega_{ij}t + \phi_i(t) \quad (1)$$

where $\Delta\theta_{ij}$ is the relative phase difference and $\Delta\omega_{ij}t$ is the phase rotation over time t. Thus, when $\Delta\theta_{ij} = 0$ and $\Delta f_{ij} = 0$, there is perfect synchronization in phase and frequency between clocks. Furthermore, let ψ_i and ψ_j represent the deviation from the nominal operating frequency f_n, with the relationship $f_i = f_n + \psi_i$ and $f_j = f_n + \psi_j$. Therefore, relative frequency offset becomes $\Delta f_{ij} = \psi_i - \psi_j$.

We use (1) to derive a deeper insight on how clocks introduce errors during RF carrier generation. For transmitter i, let the carrier frequency f^i_{clk} be obtained from the local oscillator, as shown in Fig. 3. Consider a PLL with frequency divider elements N and R, which influence the carrier frequency $f^i_{clk} = \kappa \cdot f_i$, where $\kappa = N/R$ scales the reference clock appropriately to generate the carrier. Components within the PLL architecture, such as phase detector, voltage controlled oscillator (VCO), amplifier and power supply generate phase noise (ψ_{PLL}) that contribute noise side-bands in the power spectrum. Hence, for a given radio i, the carrier frequency becomes the summation $f^i_{clk} = \kappa \cdot f_i + \psi^i_{PLL}$. Therefore, a more accurate model of RF carrier clock is:

$$\phi^j_{clk}(t) = \Delta\theta^{ij}_{clk} + 2\pi\Delta\psi^{ij}_{clk}t + \Delta\psi^{ij}_{PLL}(t) + \phi^i_{clk}(t) \quad (2)$$



(a) (b) (c) (d)

Figure 5: Effect of CFO and timing error: (a) impact of CFO on IQ Samples; (b) Received IQ samples with software-based PPS alignment and GPSDO-based PPS; (c) Correlation inaccuracy due to PPS mis-alignment; (d) Accurate cross-correlation with time synchronized PPS.

where the relative CFO is $\Delta\psi^{ij}_{clk} = \kappa \cdot (\psi_i - \psi_j)$ and phase noise is $\Delta\psi^{ij}_{PLL}(t) = \psi^{i}_{PLL}(t) - \psi^{j}_{PLL}(t)$, respectively. From (2), we see that the oscillator phase noise is transferred to the RF carrier.

3.1 Clock Synchronization on SDR Platforms

To visualize the impact of the clock synchronization, we deploy a testbed of four Ettus B210 SDRs connected to a common host computer, three as receivers and one as transmitter. The transmitter sends an unmodulated signal over the air, which is received by the three receiver SDRs. We aim to validate the clock model described in (2) by analyzing the instantaneous unwrapped phase of received signals at different receivers. An Ettus Octoclock is used as the external reference clock (10 MHz) for all the radios as we observe the clock drift between different radio oscillators. We see from Fig. 4 that $\Delta\psi^{ij}_{clk} = 0$. This is because the received signals exhibit a constant phase when connected to a common external clock. The relative CFO ($\Delta\psi^{ij}_{clk}$) for each transmitter-receiver pair can be found from the slope of the signals. The calculated CFOs, $\Delta\psi^{ij}_{clk}$, for each receiver w.r.t. to the common transmitter is 136.9Hz, 170.3Hz and 207.6Hz, respectively. When inserting these values in (2) for 10ms of signal duration, we see that each pair of radios have phase rotation w.r.t each other of around 10.1rad, implying more than 180°change in phase. In this particular study, the random phase noise $\Delta\psi^{ij}_{PLL}(t)$ is negligible because the SDR effectively minimizes it with help of a low-noise clock generator ADF4001 [10] and AD9361 integrated frequency synthesizer [11].

3.2 Clock Synchronization in Distributed SDRs

If each SDR in a distributed antenna system generates its RF carrier signal from a separate LO, the receiver is exposed to multiple CFOs. In addition, multiple signal streams arrive at the receive antenna from the transmit antennas at different instances, giving timing misalignment. We explore these effects in a testbed using an Orthogonal Frequency Division Multiplexing (OFDM) modulated waveform, which is commonly used in 802.11 a/g/n, WiMax, and LTE. OFDM is very sensitive to frequency offset and timing errors that cause inter-symbol interference (ISI) and inter-channel interference (ICI) [6].

3.2.1 CFO Estimation: For a single input single output (SISO) link, the relationship between received signal *y(t)* and transmitted signal *x(t)* is $y(t) = h(t)x(t) + z(t)$ where received symbol *y(t)* is impacted by the channel *h(t)* and additive Gaussian noise *z(t)*. The received symbol becomes $y(t+\tau)e^{j(\phi^{j}_{clk}(t))}$ due to lack of synchronization,

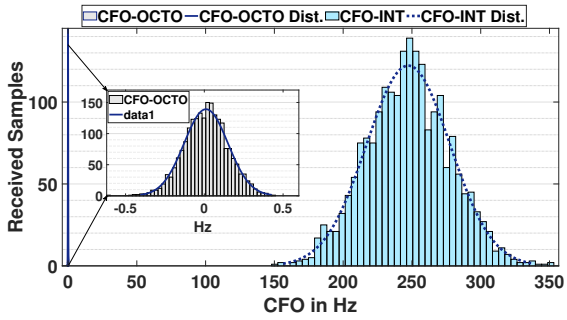


Figure 6: Impact of Octoclock (inset) and only using internal oscillator on the CFO for a transceiver-receiver pair.

where τ represents timing misalignment and $\phi^{j}_{clk}(t)$ the instantaneous phase difference derived earlier in (2). We ignore sampling clock phase offset and sampling clock frequency offset during one symbol period as their effect is not significant.

To obtain the CFO between pairwise SDRs, we generate 802.11a-compliant OFDM frames in MATLAB and transmit them over the air. We use the method described in [44] that performs coarse and fine frequency offset estimation using short (STS) and long training sequences (LTS). Fig. 6 shows the observed CFO, where the main plot describes the case of only using internal clocks without any CFO correction. The plot in the inset contrasts this with the case where the Octoclock is the external shared clock. Thus, when the B210 SDRs operate with their internal clock, their CFO is in the range 150-350 Hz, whereas the CFO with Octoclock is in the range 0-0.5 Hz. This corresponds to an average ratio of 0.2 ppb (parts per billion). Due to the CFO arising from the internal oscillators of SDRs, the points on the constellation diagram keep moving along the unit circle in the same direction, as shown in Fig. 5a. This is a marked deviation from the expected and ideal BPSK constellation.

3.2.2 Effect of Timing Error in DCB Application: Apart from the external clock, SDRs require a PPS signal for their operation. The clock signal (10 MHz in Ettus B210 SDRs) is used to drive the digital and analog circuits of the RF front-end and the PPS signal is used to control the synchronized operation. Note that due to variable latency in the link between RF front-end and the host computer, software-only synchronization is not precise. Thus, RFClock is designed as a stand-alone hardware solution to achieve time synchronization in the order of nanoseconds. To observe the effect of timing error between multiple radios in a DCB application, we conduct an experiment with 3 USRP B210 radios, with two of them as transmitter and one as the receiver. Each transmitter generates a

frame with predefined Gold sequence as training symbol, followed by the same OFDM blocks encapsulating BPSK modulated symbols with proper zero-padding and cyclic prefix (CP) insertion. All SDRs are connected to an Octoclock that provides the 10 MHz reference and eliminates CFO error. However, these disconnected SDRs must perform processing tasks on samples aligned in time, i.e., at the same sample clock edge for correct DCB. The DCB implementation is straightforward: We introduce a channel state feedback process that exploits statistical knowledge of channel characteristics by (i) correlating the incoming samples against the stored Gold sequences to detect an individual transmitter, and then (ii) performing Least Squares (LS) estimation to estimate the channel. The receiver updates the transmitters with the beamforming weight vector $\mathbf{w}$ every 50ms, a limitation posed by GNURadio as it must pause for this time to avoid buffer overflow. The transmitted symbols $s[m]$ are multiplied by the beamforming weights to construct the new signal $x[m] = \sqrt{E_s}\mathbf{w}^H s[m]$, where E_s is the average energy of the transmitted signal $x[m]$ with normalized constellation symbols at any instant m. Even with perfect software-based time co-ordination among transmitters, the starting point of two copies of same OFDM symbol from different transmitters may not coincide with the exact timing of receiver FFT window. This affects the correlation of training symbols from distributed transmitters, an example of which is shown in Fig. 5c. We observe an inaccuracy of 14500µs between the cross-correlation peaks from the two transmitters. This results in the PPS edges being misaligned, causing a mismatch in the phase synchronization. The resulting rotation in the constellation points at the receiver is shown in Fig. 5b. With PPS alignment, this issue can be resolved, the result of which can be seen in Fig. 5d, where, with accurate cross-correlation of training signals, the correlation peaks get aligned within 1µs.

We also studied issues with GPS synchronization by repeating the experiment outdoors with two USRP B210 SDRs paired with the Ettus GPS disciplined oscillator (GPSDO) that provides the 10 MHz clock and PPS signal. We observe that the relative time error (TE) between two GPSDO-sourced PPS is ±500ns. However, the relative phase drift between two clock outputs of GPSDOs is not stable, varying between 0-100ns, whose adverse effect is seen in the constellation diagram at the receiver-side (see Fig. 5b).

4 RFCLOCK STEP 1. FREQUENCY SYNCHRONIZATION

We describe RFClock design by separately considering the (i) *frequency* and (ii) *time and phase* synchronization. We discuss the former in this section and the latter in Sec. 5.

Our key idea is that by combining two tones that are separated by the frequency of reference clock, we can produce an envelope signal at the intended reference clock frequency. Then, the receiver can extract this output envelope with additional processing steps to obtain the reference clock. We formally explain this process next.

4.1 Extracting Reference Clock Signal

Let $x_1(t) = Ae^{j2\pi f_1 t}$ and $x_2(t) = Ae^{j2\pi f_2 t}$ be two single tone sinusoidal signals of amplitude A at frequencies f_1 and f_2, respectively, which are combined by the RFClock leader. Therefore, the transmitted two-tone signal is $S_{tx}(t) = A\sum_{k=1}^{2} e^{j2\pi f_k t}$. This superposition

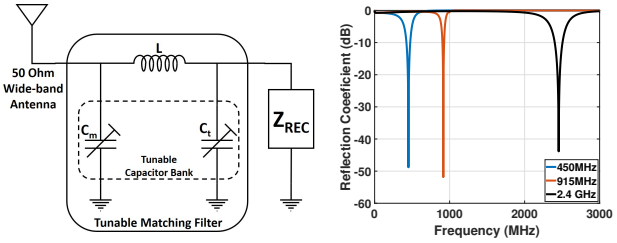


Figure 7: Tunable filter design showing Π filter network with tunable capacitors and inductors as in (a) and optimized filter frequencies in (b).

of waves can be written as a function of the sum and difference of the frequencies f_1 and f_2,

$$S_{tx}(t) = 2cos(2\pi((f_1 - f_2)/2)t)e^{j2\pi(\frac{f_1+f_2}{2})t} \quad (3)$$

One part is a quadrature signal which oscillates with the average frequency $f_{avg} = \frac{f_1+f_2}{2}$. The other part is a cosine wave which oscillates with the difference frequency $f_{diff} = \frac{f_1-f_2}{2}$, as if it were the *modulator signal* controlling the *envelope* of the resulting wave. As the envelope crosses the zero mark twice in every period, the envelope frequency is twice the difference frequency. This is given by the magnitude of the difference of the two frequencies as $f_{env} = |f_1 - f_2|$. This is the reference clock signal f_{clk} that is extracted at the RFClock follower through a suitably designed envelope detector. As the two-tone signal $S_{tx}(t)$ propagates over the wireless channel, the received version at the RFClock follower $S_{rx}(t)$ can be expressed as:

$$S_{rx}(t) = A[\alpha_1 e^{j(2\pi f_1 t + \phi_1^{ch})} + \alpha_2 e^{j(2\pi f_2 t + \phi_2^{ch})}] \quad (4)$$

where α_1 and α_2 are signal attenuation constants, and ϕ_1^{ch} and ϕ_2^{ch} represent phase change of the signals due to the wireless channel. This received signal is given to an envelope detector that outputs the full-wave signal $S_{env}(t)$ at frequency f_{clk}, which is the envelope of the quadrature signal as discussed earlier.

$$S_{env}(t) = A|\alpha_1 e^{j(2\pi f_1 t + \phi_1^{ch})} + \alpha_2 e^{j(2\pi f_2 t + \phi_2^{ch})}| \quad (5)$$

We re-write (5) as a voltage-shifted version of the modulator signal at frequency $f_{clk} = |f_1 - f_2|$ as $A\{2(\alpha_1^2 + \alpha_2^2) + 2\alpha_1\alpha_2 \cos(2\pi(f_1 - f_2)t + (\phi_1^{ch} - \phi_2^{ch}))\}^{1/2}$. The first term is a DC component that we filter out with a band-pass filter centered at f_{clk}. All the RFClock followers now have the same drift as they are locked to a common RFClock leader.

4.2 Effect of Multipath and Motion

If there are L independent propagation paths for the reference tone, with the first arriving signal taking the direct path, then the received signal at a given RFClock follower is the summation: $S_m(t) = |A\sum_{m=1}^{L}\alpha_m\{\sum_{k=1}^{2} e^{j(2\pi f_k t)}\}e^{j\phi_m^{ch}}|$. Here α_m and ϕ_m^{ch} are the attenuation and phase shift for the m^{th} path, respectively. For simplicity, we assume that tones have 0 initial phase, then the corresponding envelope of this signal is obtained from (5) as:

$$S_m(t) = A\Bigg\{2\sum_{m=1}^{L}\alpha_m^2\big[1 + cos(2\pi(f_1 - f_2)t)\big] + \sum_{\substack{m=1 \\ m\neq n}}^{L}\sum_{n=1}^{L}\alpha_m\alpha_n\big[cos(2\pi(f_1 - f_2)t - \Delta\phi_{mn}^{ch}) + cos(\Delta\phi_{mn}^{ch})\big]\Bigg\}^{1/2} \quad (6)$$

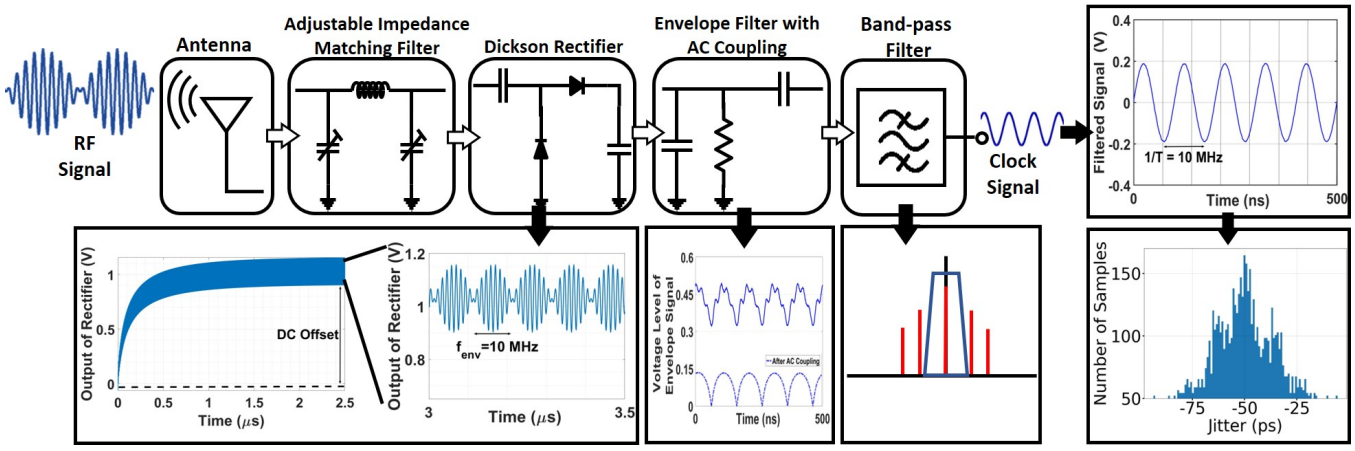


Figure 8: Circuit chain of RFClock front-end performs reference clock extraction: each plot demonstrates the output of the each cascaded unit obtained from real test-bed.

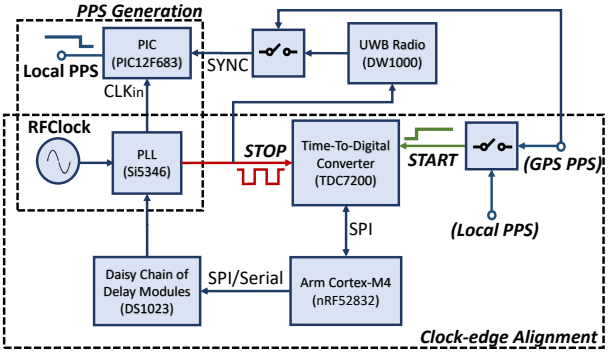


Figure 9: PPS generation and clock-edge alignment.

In a multipath environment, the extracted beat frequency $|f_2 - f_1|$ remains the same, which we validate later in Sec.6.3. However, in dynamic environments or due to relative motion, the extracted frequency at the RFClock follower may shift due to change in the phase offset $\Delta\phi_{mn}^{ch}$ between, say, the m^{th} and n^{th} path. This introduces random perturbations called *jitter* in the clock's signal edges. The PLL in the RFClock follower has a digitally-controlled loop filter that increases the amount of jitter attenuation at such times by reducing the loop filter bandwidth. We investigate these effects of dynamic environments on the followers in Sec. 6.3.

4.3 Frequency Agility

RFClock's operation can be impacted by an interfering RF signal, with the resulting link disruption causing (i) an increase in CFO at the follower, or (ii) the follower PLL to lose its lock with the leader. To mitigate this, we design a tunable matching filter that allows RFClock to switch between frequency bands. Fig. 7a shows the filter topology consisting of digital tunable capacitors and inductors arranged in a Π network. We optimize the tunable filter for different center frequencies with 10MHz bandwidth, as shown in Fig. 7b. We adjust center frequency by changing capacitance C_m while inductor L is kept constant. Capacitance C_t is used for matching the filter to load and source impedance, which are the RF front-end input and antenna impedance, respectively. We optimize this filter for 900-910MHz and 1800-1810MHz, since the antenna supports dual bands.

5 RFCLOCK STEP 2. TIME AND PHASE SYNCHRONIZATION

The frequency synchronization in Sec. 4 compensates for *clock drift*, allowing all followers to track the frequency of the leader. However, there may still be a phase difference between their respective clock edges because signal propagation time for leader-follower pairs may vary. RFClock includes a PPS module, which generates a uniform stream of pulses derived from the RFClock's front-end output (i.e every 100ns at the reference frequency of 10MHz). However, the PPS generation must have a common origin for all followers, and in absence of such a synchronized start or correction, the difference between PPS trigger instants can grow rapidly. Thus, we need to (i) compensate the phase offset (ΔT_{PPS}) between PPS edges of different nodes at a hardware level (called as phase synchronization) (see Sec. 3.2.2); (ii) establish a common origin to start the PPS generation (called as time synchronization); (iii) include a recovery mechanism when synchronization fails. In this section, we describe the hardware and software components designed to overcome the above challenges. Our approach uses the six components as shown in Fig. 9: 1) RFClock front-end, 2) PLL, 3) Time-to-Digital (TDC) converter, 4) PIC (PIC12F683) microcontroller, 5) ARM (Cortex M4) processor, and 6) a daisy chain of delay modules. The end-output of this block is a 1PPS signal, phase locked to the 10 MHz clock and also phase synchronized across the entire network of nodes. We have a two step process through which we (i) first generate the PPS signal, and then (ii) align it among all the followers.

5.1 Base PPS generation

To obtain a stand-alone PPS signal, we implement a digital frequency divider (DFD) within the PIC controller. It takes as input the signal generated by the PLL port, which in turn is phase locked to the RFClock envelope detector output signal at 10MHz (see Fig. 9). The PIC then executes the DFD code custom written in assembly language, where each instruction requires one processor clock cycle to execute. Thus, it takes 2.5M instruction cycles to derive an exact measure of 1 second. This generates a stream of pulses with a fixed pulse width of 200ms corresponding to %20 duty cycle, which is standard for most commercial PPS generators, such as the Octoclock. The dedicated microcontroller avoids time jitter as it is

not interrupted by other real time operations of the ARM processor. To measure the precision of the PPS spacing in time, we export $10K$ pulses and calculate the time difference between successive rising edges. The maximum and minimum values of the period jitter are within 80ps, and the RMS (root-mean square) of the period jitter is 20ps. This remarkably low jitter is visually depicted in Fig. 8.

5.2 PPS Alignment

We next implement a simple approach to correct the phase difference ΔT_{PPS} between two PPS signals. This first method in this section uses a global PPS that is used for correction, such as the PPS obtained from GPS signals, while the second method uses UWB-based ranging to calculate relative offsets among followers.

5.2.1 PPS alignment with GPS. This method involves an ARM processor to perform these tasks: *i*) as part of an initialization step, it resets and syncs the system PPS with a global source PPS, *ii*) it tracks phase differences between the internally generated and external triggers at each PPS edge, and *iii*) it adjusts the clock edges according to, ΔT_{pps}, through a delay chain. Specifically, the clock-edge alignment module has a synchronization line (SYNC) that resets the time-base for PPS generation. The SYNC pin is only read at the next rising edge of the 10MHz clock. This introduces up to 100ns of error unless the source driving the SYNC line is phase aligned with the 10MHz clock. Therefore, this method only allows an offset correction within a maximum 100ns error margin. To estimate the residual error, we measure the phase error between the GPS PPS input and the next positive edge of the 10MHz clock by using a time-to-digital converter (TDC). The TDC measures the phase between the PLL-generated pulse derived from the 10MHz envelope detector that arrives at its STOP pin and the other 1PPS pulse stream arriving on the START pin, with a resolution of picoseconds (see Fig. 9). The output of the TDC is used by the ARM processor to activate the delay chain with the estimated phase error to align the clock edge to the global PPS edge. This feedback loop controls the PLL (and in turn the PIC's PPS output) by adjusting the delay element using the output from TDC measurements.

5.2.2 PPS Alignment with UWB Ranging. Since GPS requires line-of-sight to satellites and works best outdoors, RFClock incorporates an auxiliary UWB-based message exchange protocol for GPS-denied environments described in this section. As shown in Fig. 10, we start from the point where the RFClock leader and follower have their respective rising edges of the PPS separated by ΔT_{PPS}, although they have no relative CFO. The UWB module in the follower sends a POLL message to the leader and records the local time T_s. The leader records the reception time of POLL message at t_{rx}^{poll} local time, takes an additional t_ϵ to initiate the reply, and then sends out a RESPONSE (RES) message at local time t_{tx}^{res}. Both the times t_{rx}^{poll} and t_{tx}^{res} are included in the payload of the RES message. These timing relationships can be expressed as:

$$\begin{aligned} t_{rx}^{poll} &= (T_s - \Delta T_{PPS}) + t_{tof} \\ t_{tx}^{res} &= (T_s - \Delta T_{PPS} + t_{tof}) + t_\epsilon \\ t_{rx}^{res} &= T_s + 2t_{tof} + t_\epsilon \end{aligned} \tag{7}$$

where, t_{tof} is the unidirectional time of flight. We calculate ΔT_{PPS} from (7) as follows,

$$\Delta T_{PPS} = \frac{(T_s - t_{rx}^{poll}) + (t_{rx}^{res} - t_{tx}^{res})}{2} \tag{8}$$

Here, the controller orchestrates the ranging instructions and gathers error measurements caused by the phase ambiguity between UWB reference clock 38.4 MHz and PPS through TDC and processing delay that changes with each ranging cycle. The processing delay is compensated within maximum of 5 ranging iterations. The residual error ($\Delta\zeta$) that remains after resetting the local PPS with the estimated phase offset ΔT_{pps} is finally applied to the delay chain to remove any remaining offset between PPS edges.

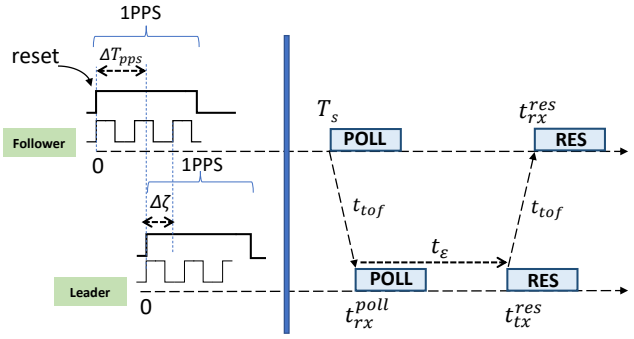


Figure 10: Timing diagram for UWB-based ranging.

5.2.3 Coordinating Start Time for Transmission. The timestamps obtained from ranging can be further exploited to provide a common notion of time across the network for simultaneous start of transmissions by all RFClock followers. After all the nodes are phase synchronized and have PPS aligned with respect to the leader PPS, the latter broadcasts a UWB POLL message at time T_s. This message helps each follower to estimate the time difference ΔT_{offset} between leader and itself by using (9). Each SDR uses this offset provided by RFClock to appropriately schedule the starting instant of its transmission for an application like DCB.

$$\Delta T_{offset} = (T_s - t_{rx}^{poll}) - t_{tof} \tag{9}$$

where t_{tof} is the time of flight that is estimated by utilizing timestamps as $[(t_{rx}^{res} - T_s) - (t_{tx}^{res} - t_{rx}^{poll})]/2$.

6 SYSTEM EVALUATION

The block diagram for the implementation of RFClock leader and follower is shown in Fig. 11a and 11b, respectively. The complete schematics, bill of materials, component specifications can be downloaded from the anonymized GitHub respository [41]. We evaluate RFClock, both in terms of its synchronization capability with respect to wired and GPS solutions, as well as overall performance when used with COTS B210 SDRs for DCB. In this section, we perform RFClock's experiments in indoor and outdoor settings. Experiments are performed in a 96ftx124ft crowded office environment (e.g many desks, metallic equipment and other types of reflectors in close proximity) as shown in Fig. 13a. We also conduct experiments outdoors, amidst low/moderate-height buildings with maximum 100ft leader-follower separation, and also approximately 6ft inter-follower separation when placed in a linear array.

6.1 Implementation Summary

The RFClock leader consists of a i) reference oscillator Ettus GPSDO (e.g 10 MHz), ii) RF frequency synthesizer ADF4350 that locks to reference oscillator to produce two-tone signal at desired frequency

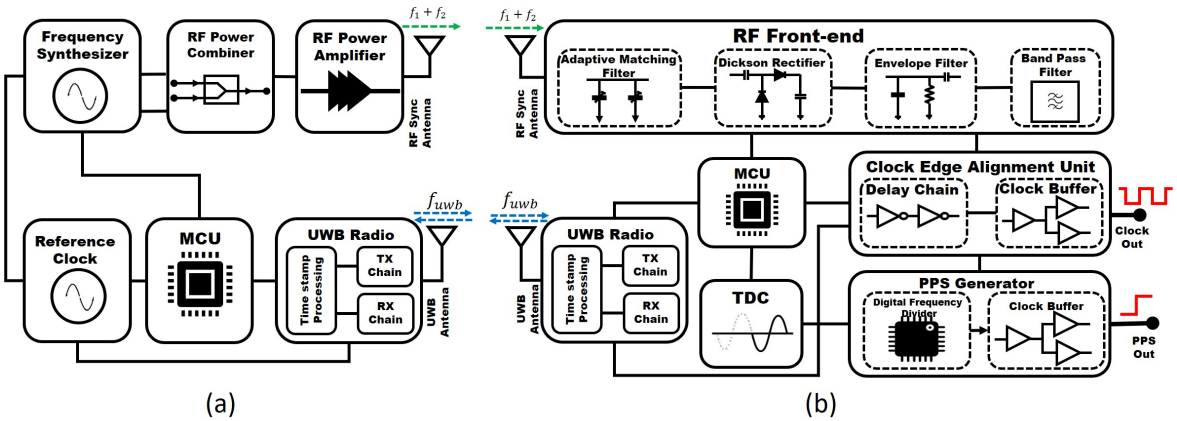


Figure 11: (a) RFClock leader schematic; (b) RFClock follower schematic. GPS integration is not shown for clarity.

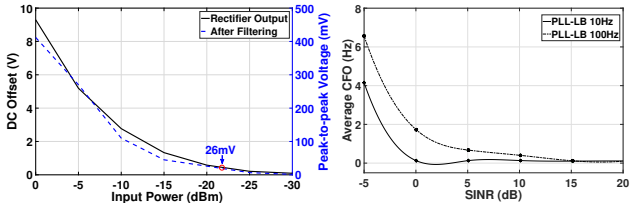


Figure 12: (a) RFClock front-end performance based on input power (see also Fig. 8); (b) RFClock enables frequency synchronization under a wide range of SINR values.

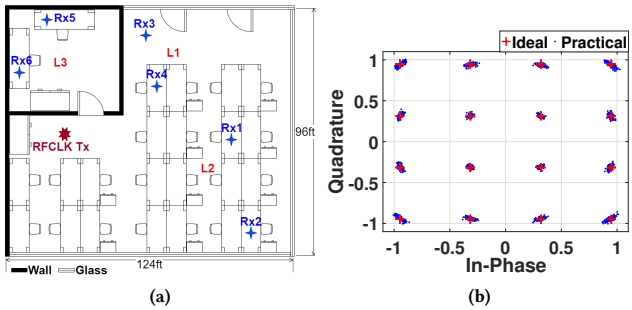


Figure 13: (a) Crowded office environment for experimental evaluation; (b) Constellation diagram of 16-QAM with RFClock synchronization in DCB. The constellation points (in blue) were superimposed over 100 iterations.

$f_1 + f_2$, iii) RF power combiner and amplifier used to combine the two-tone signal and transmit over-the-air, iv) a DECAWAVE DW1000 UWB radio IC, and v) an ARM-Cortex microcontroller nRF52832, that orchestrates all RFClock functions.

RFClock follower consists of following design units; i) RFClock front-end, ii) clock-edge alignment, iii) PPS generation and iv) phase/time estimation (see also Fig. 2 for the fabricated design). Our front-end design (see Fig. 8) consists of passive resistors, capacitors and diodes. An adjustable impedance matching filter composed of wiSpry WS1040 digital capacitor array allows flexible tuning of desired frequencies. A 4-stage rectifier composed of HSMS285C Schottky diodes extracts the envelope. Last, the extracted clock signal passes through a band pass filter with center frequency at 10MHz. The envelope output drives a low jitter PLL Si5346 from Silicon Lab. One output of the PLL is connected to the PIC microcontroller PIC12F683 to produce 1PPS signal. The phase difference between 10MHz/38.4MHz and the local PPS is measured using the time to digital converter (TDC). The UWB Decawave DW1000 [7] module (from Sec. 5.2.2) is responsible of estimating ΔT_{pps}, which has capability of time-stamping the transmission and reception of packets with a resolution of 15.65ps. The Cortex-M ARM microcontroller synchronizes clock edges with estimated phase offset through the delay chain composed of cascaded multiple DS1023 timing elements that allows delays up to 100ns. Finally, power consumption of front-end design is 6.6µW, while the energy consumed for a single UWB ranging operation is 0.159µJ. As seen from Table 2, UWB radio's current consumption changes over each operation state. When we consider all system components such as RFClock front-end, clock edge alignment and phase/time estimation, the power consumption range changes between $[170 - 390]mW$.

Deep Sleep	Sleep	Idle	Tx	Rx
$50nA$	$1\mu A$	$14mA$	$59mA$	$75mA$

Table 2: Current consumption of UWB operations

6.2 RFClock Coverage Range

We next obtain the maximum coverage range of the system using conventional *free space path loss (FSPL)* [19] between leader and follower. We do so by measuring follower RF sensitivity in terms of front-end design and UWB ranging performance, and determining leader's transmitted power along with other parameters. Fig. 12a shows the measured peak-to-peak voltage of follower's extracted clock signal versus input power (antenna gain not included). We pick −22dBm as our front-end sensitivity, which is the minimum required power for our clock recovery mechanism to perform recovery and enable the rest of the system. In addition, we adjust the leader's transmission power to the maximum permissible level based on FCC's limitation of 36dBm per transmitter in the 900MHz ISM band FCC [16]. Since the leader transmits two different single tones over the air, the total transmitted power is 39dBm. This gives the maximum allowable coverage radius as 164ft (with additional increase possible with a multi-antenna transmitter at the leader). Similarly, the regulatory limit for UWB is 41.3dBm/MHz if frame transmission time is less than 1ms, which corresponds to total channel power of −14.3dBm/500MHz. We adjust the UWB parameters to increase this upper bound such that by using the highest data rate

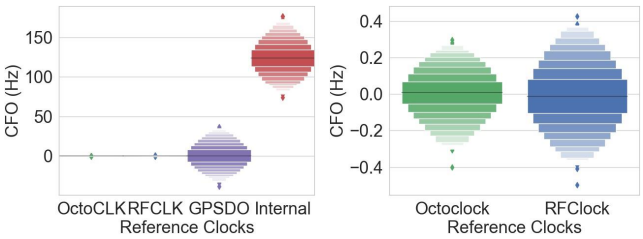


Figure 14: Minimizing CFO: Performance comparison of different reference clocks used with SDRs (left); Zoomed in view of performance of RFClock vs. Octoclock from left Fig. at 915MHz with 5MHz channel bandwidth (right).

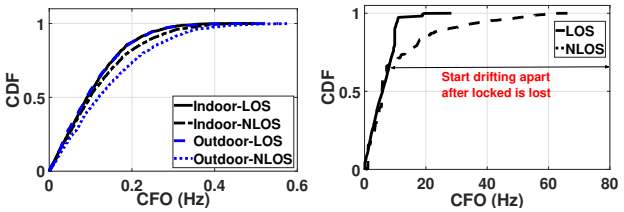


Figure 15: Behaviour of RFClock and GPSDO in LOS/NLOS environment with indoor and outdoor settings: (a) RFClock; (b) GPSDO.

of 6.8Mbps and a preamble length of 128, we can contain the total frame transmission time to 180µs. This reduction allows boosting transmitter power to 6.9dB. The minimum permissible receiver sensitivity of UWB chip is −94dBm, which can be improved by decreasing clock offset between paired radios. Since our system eliminates this offset by extracting reference clock from the leader's transmission (see also Fig. 18a), we decrease this lower bound to −106dBm. Considering a frequency of 3993.6MHz, the coverage distance of this UWB chip is around 656ft, which also serves as the range for successful exchanging of probe packets.

6.3 Testing Frequency Synchronization

Method: We integrate six SDRs with RFClock followers and place them at random locations in a 96 ft X 124 ft indoor area with maximum leader-follower separation of 80 ft. We use 802.11n OFDM frames to estimate CFO between these SDRs driven by the RFClock, as explained in 3.2.1. Each such OFDM packet is of length 1024 bytes with QPSK/64QAM modulated random data as payloads, and is transmitted in a 5MHz channel bandwidth at 915MHz and 40MHz channel bandwidth at 2.4GHz. The frame structure consists of two training sequences, STS and LTS. STS occurs at the beginning of the OFDM packet and is used to detect the start of the packet along with coarse frequency offset estimation. After this, LTS is used for channel estimation and fine frequency offset. Thus, the overall CFO is summation of these two individual offsets. However, CFO estimation is affected by SNR level of 802.11n WiFi signals, inducing extra estimation noise as error into CFO during this estimation, which does not fully capture the real accuracy of the RFClock. To eliminate multipath effect on CFO estimation and providing high SNR conditions, we move one SDR radio attached with RFClock follower to the locations of other radios, who will transmit and receive WiFi frames, and connect them via cables at RX and TX ports to enable ground-truth error floor of the system while RFClock

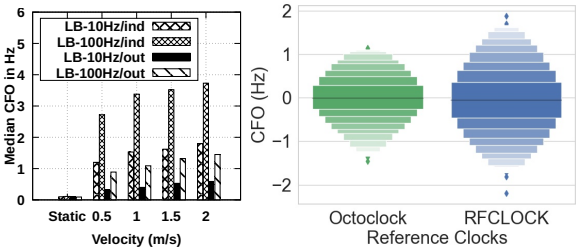


Figure 16: (a) Impact of mobility on CFO; (b) CFO estimation at 2.4GHz with 40MHz channel bandwidth.

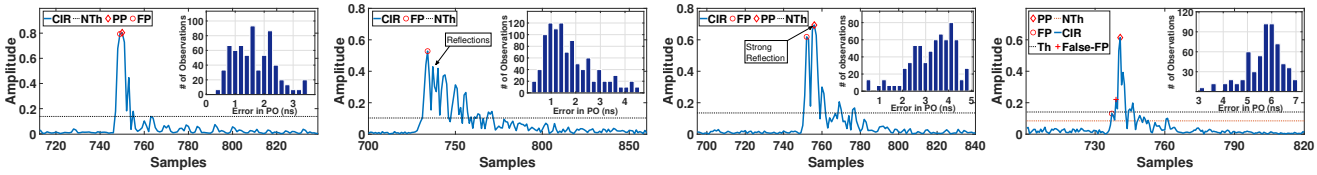
leader and followers remained as in Fig. 13a, except the follower attached to transmitter SDR radio. Then, we obtain results with omni-directional antennas with the transmitter SDR attached to RFClock leader. This setup allows us to extract estimation error w.r.t ground-truth data and estimate CFO for each of the radios.

Performance Comparison: We repeat the above test with (i) the Ettus Octoclock and (ii) Ettus GPSDO and average estimated CFO of over 2000 packets transmission for all set of radios. We first observe that CFO varies between 75−180Hz, with a median value of 123.6Hz at 915MHz carrier frequency for different SDR units from the same Ettus B210 family, when an internal oscillator is used. From Fig. 14 the deviation in the CFO of these different synchronization methods can be observed. RFClock is superior to the GPSDO as the median CFO is 0.094Hz and 7.58Hz, respectively. Moreover, RFClock performance approaches the wired Octoclock, which has a median CFO of 0.059Hz. Fig. 16b shows the CFO comparison when WiFi packets are transmitted at 2.4GHz with 64QAM modulated data in a 40MHz channel bandwidth. The median CFOs are 0.263Hz and 0.401Hz for Octoclock and RFClock, respectively.

Multipath and NLOS performance: We next study the impact of multipath and NLOS on synchronization of the SDRs in a rich indoor multipath environment located in L1 and L3 as in Fig. 13a and outdoor settings where RFClocks coordinate with/without LOS. The same experiment is repeated outdoors with/without LOS to satellites for GPSDO-mounted SDRs. From Fig. 15a, we see that RFClock is not significantly impacted by NLOS indoors, and slightly degrades outdoors while maintaining CFO within permissible range. However, GPSDO's clock starts drifting resulting in increased CFO error, as observed in Fig. 15b.

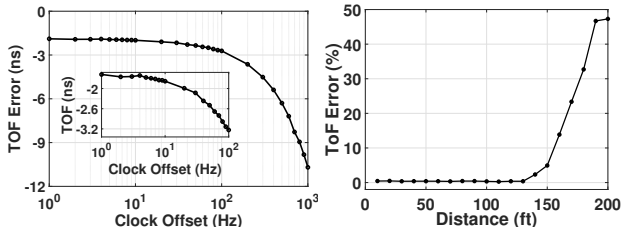
Impact of mobility: We consider moderate human mobility, ranging from typical walking speed of 0.5m/s to running speed of 2m/s in indoor and outdoor settings. From Sec. 4.2, we recall that mobility introduces jitter in the received clock signal. We mitigate this effect by optimizing the digital loop bandwidth of the PLL to increase jitter attenuation. Fig. 16a demonstrates that mobility induces CFO error up to 3.73Hz indoors when the PLL loop bandwidth is 100Hz. This error is reduced by decreasing loop bandwidth to 10Hz, which reduces error down to 1.8Hz at running speed. Also, we observe that multipath fading impacts CFO more in indoor settings.

Phase Misalignment: Here, we study how much phase drift is induced due to CFO error within a single OFDM packet. With coding rate as 1/2 and data packet length and modulation, the legacy preamble STS consumes 32µs and LTS takes 32µs[30]. Hence, the total packet duration is around 2.8ms. 95% of phase misalignment during one packet duration is 0.0037rad. According to [34], 99%



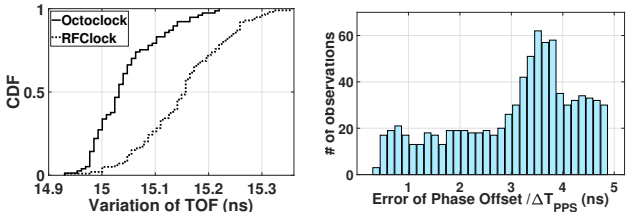
(a) (b) (c) (d)

Figure 17: Estimated UWB Channel Impulse Response (CIR) obtained from RFClock followers in different channel conditions along with distribution of Phase Offset (PO) error: (a) Outdoor LOS. The peak corresponding to first direct path is clearly distinguishable; (b) Indoor LOS (follower RX1 in Fig. 13a). The peaks of multiple reflections follow the first direct path; (c) Indoor NLOS (follower Rx4 in Fig. 13a). The peak corresponding first direct path is not the strongest peak; (d) Indoor NLOS (follower Rx5 in Fig. 13a). LDE is not able to detect the first direct path resulting in increased PO error.



(a) (b)

Figure 18: (a)ToF versus Frequency offset. Absolute ToF error increases with the frequency offset; (b) ToF error vs distance.



(a) (b)

Figure 19: RFClock Accuracy: (a) RFClock's ToF variation compared to Octoclock at a distance of 20ft; (b) Error in PO estimation after JBSF.

beamforming gain is achieved when phase misalignment of the received signals is in the margin of 15°(or 0.261rad). Therefore, we conclude that RFClock ensures phase coherence between radios. RFClock is capable of coherent transmission in our mobility scenarios, since the maximum misalignment in a packet duration is 0.623°(or 0.011rad).

Impact of Interference: To evaluate the performance of RFClock under interference conditions, we intentionally introduce another SDR transmitter within the coverage of the earlier setup. We measure the average signal-to-interference-plus-noise-ratio(SINR) for every leader-follower pair to demonstrate the effect of interference on estimated CFO accuracy. Low SINR levels introduce jitter at recovered clock signals that induces increased CFO error during communication, as shown in Fig. 12b. This figure also shows how CFO error caused by low SINR level drops by reducing loop filter bandwidth of the PLL and CFO accuracy. The outcome is as good as operating in an interference-free condition (see 14b) when SINR ≥ 0. For SINR < 0, phase misalignment during packet duration is 4.18° which is in permissible range.

6.4 Testing Time/Phase Synchronization

Method: We use the phase/time estimation unit of RFClock follower and RFClock leader that are mainly controlled by the UWB module and the ARM controller, as described in Sec. 6.1 and shown in the schematic Fig. 11b. All evaluations are carried out UWB radio's highest data rate of 6.8Mbps with preamble length of 128 symbols and a pulse repetition frequency of 64MHz.

Time of Arrival Estimation: We need to ensure accurate timestamps of received messages (POLL/RESPONSE) to estimate phase offset, ΔT_{pps}, which depends on time of arrival estimation (TOA). There are several techniques in literature to estimate TOA in different channel conditions (LOS/NLOS) and/or real multipath environments [7]. The key idea is to first detect the direct path of the incoming signal and thereby estimate arrival time as exactly as possible. For this purpose, we use the *leading edge detection (LDE) algorithm* embedded in the DW1000 to detect the direct path of the incoming signal that is extracted from channel impulse response (CIR) measurements stored in a large buffer (4096B) with roughly 1ns sampling time [8]. LDE is a threshold-based algorithm that detects the first direct path (FP) when the first stored CIR sample is above the dynamically adjusted threshold. The threshold is calculated based on standard deviation, σ_η, and peak value of the estimated noise. To decrease false FP detection due to error in threshold estimation, we implement a similar approach to *jump back and search forward (JBSF)* [21] that searches whether there is another leading edge that exceeds the new calculated threshold in a pre-determined window (W_n) after determining the FP by LDE. We calculate the new threshold level by $(NTMx\sigma_\eta)xc$, where NTM is noise threshold multiplier set by DW1000 and c is empirically obtained. Keeping c in the range [0.4,0.6] decreases false FP detection, especially in NLOS conditions.

ToF metric vs Clock Offset: Our first study investigates the effect of clock offset on ToF estimation. To eliminate clock offset between UWB radios, We enable external synchronization of these radios by generating two 38.4MHz clock signals from the same PLL and attaching them to our custom-designed boards, which provides access to UWB radio's clock input (in system level practice, this clock input is fed by output of the RFClock's front-end). We control the clock offset between these two 38.4MHz clock signals through the PLL, and this allows us to generate a known frequency offset between two devices. The ToF is measured by UWB ranging as shown in Fig. 10. We increase the frequency of the one radio's reference clock signal (38.4MHz) in steps of 1Hz, up to 1KHz. From Fig. 18a we observe that the ToF error escalates significantly through increase of clock offset. Consequently, this error decreases synchronization accuracy while inducing error in ranging.

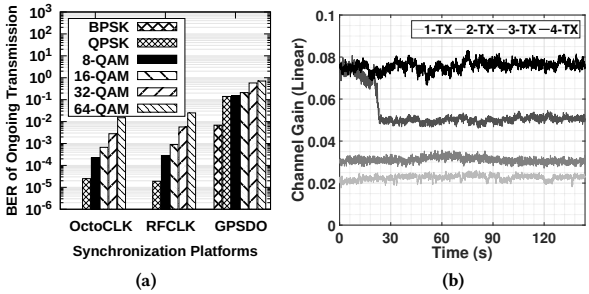


Figure 20: (a) BER performance in different modulation schemes and synchronization methods; (b) Channel gain with increasing number of transmitters.

RFClock ToF estimation: Accurately estimating the time offset requires high precision ToF estimation. Using the steps explained in Sec. 5.2.2, we conduct ranging experiments at different distances. The ground-truth data is collected using a wired Octoclock driving the PLL (we generate 38.4MHz from Octoclock output). In Sec. 3.2.1 for 915MHz, we have validated the Octoclock frequency offset for 38.4MHz clock as 0.0077Hz. This gives pico-second level error in ranging. The variation of ToF, indicated in Fig. 19a, is 477ps, which is nearly equal to Octoclock. Therefore, the effect of RFClock's clock offset is negligible (see Fig. 18a). We also repeat experiments to study the distance versus RFClock performance in terms of ToF error, noting that the any degradation starts only after 150ft, that is a limitation imposed by RFClock front-end, also explained in 6.2. After this point, the clock recovery starts degrading and begins to impact ToF estimation, as shown in Fig. 18b.

Phase Offset Estimation: We next evaluate the pairwise synchronization performance of RFClock leader and RFClock follower. Our aim is to estimate phase offset of PPS w.r.t leader's PPS. The phase offset (see Sec. 5.2.2) ΔT_{pps} is computed using timestamps recorded through POLL and RESPONSE messages. The interrupt processing delay, while resetting the local PPS, is compensated through several rounds of ranging between nodes. The offset between 38.4MHz and local 1PPS is measured by TDC module. DW1000 first enables a coarse RX timestamp estimation that records the first pulse of PHR (Physical Layer Header) after the SFD (Start of Frame Delimiter) [24] and adjusts this value based on the RX antenna delay and the first path (FP) index in CIR estimation (ToA) detected by LDE. We demonstrate the relationship between estimating ToA of the transmitted signal and error in phase offset estimation of the PPS in different channel conditions in Fig. 17. Here, we present the multipath propagation characteristics of the UWB channel between leader and followers, which is obtained from CIR measurements. Fig. 17d showcases the scenario where the first sampled amplitude exceeds the threshold (Th) detected as FP. This false FP detection results from the wrong estimation of the noise threshold. Moreover, the leader-follower error distribution calculated from 500 observations from six followers (see scenario Fig. 13a) is shown in Fig. 19b. The synchronization error is below 5ns, when we enable the approach with the estimated new dynamic threshold value (NTh).

7 USING RFCLOCK FOR DCB

We setup four transmitter B210 SDRs in a linear array located in L1 location of the scenario illustrated in 13a and one receiver B210 SDR to demonstrate DCB with single user MISO (multiple input single output) and the resulting N^2 increase in the received power, where N is the number of available transmit antennas [25]. We adapt the DCB approach from Sec. 3.2.2 by integrating the RFClock follower with all the SDRs for frequency, phase and start time synchronization. We also combine BPSK, QPSK, 8-QAM, 16-QAM, 32-QAM and 64-QAM modulated symbols with the receiver-generated beamweights to study the impact on higher order modulations and evaluate DCB with RFClock in moderate SNR regime (10-15dB). Using the same setup, we replace the RFClock synchronization module with Octoclock and GPSDO for comparison with RFClock.

7.1 Experimental Evaluation

The synchronization accuracy of RFClock in DCB is showcased in Fig. 13b, resulting in near-zero phase and frequency offsets on the received I/Q symbols over time throughout the duration of the experiment. The impact of RFClock on BER for different modulation schemes, when compared with Octoclock and GPSDO, is shown in Fig. 20a. We see that the BER performance of RFClock is similar to the wired setup of Octoclock for modulation schemes up to 8-QAM (10^{-6} for BPSK and QPSK), but degrades slightly for 16-QAM, 32-QAM and 64-QAM. The BER performance with GPSDO fares worse in comparison, with the BER staying near 10^{-2} for BPSK and rising to 10^{-1} for higher modulation schemes. Fig. 20b showcases the expected effect of increasing channel gain due to DCB, as we increase the number of transmitters. This improvement in channel gain is a result of in-phase arriving signals from the transmitters, which in turn improves BER at the receiver.

Our results demonstrate that RFClock's performance is close to current wired synchronization approaches used in the industry, such as Octoclock, and performs better than the state-of-the art, GPSDO. For example, the requirements for 802.11.ax/ac to achieve MU-MIMO is 350Hz relative clock offset between transmitters with ±0.4μs timing constraints, while the requirement for realizing MIMO with transmitter diversity in 5G is ±65ns. Both these applications can be supported by RFClock.

8 CONCLUSION

RFClock enables highly accurate time, phase and frequency synchronization as a stand-alone hardware solution, and can be interfaced with distributed COTS SDRs. We develop the theory for such precision synchronization and implement it in a custom-design, which we release as an open-source community resource. We compare the performance of RFClock with popular wired as well as GPS-based hardware solutions, both in terms of clock performance as well as impact on distributed beamforming. Our experimental studies reveal RFClock shows a phase offset of less than 5ns and frequency offset of less than 0.1Hz, which is at par with wired solutions. In addition, when used for DCB with four transmitters, RFClock results in a BER of less than 10^{-5} for QPSK modulation and close to 10^{-4} for 8/16-QAM, respectively. Our next steps will focus on (i) demonstrating RFClock over km-long separation, and (ii) increasing resiliency in highly mobile, terrestrial and airborne scenarios.

Acknowledgments. This work was supported in part by US National Science Foundation (NSF) under research grant CNS1452628 and Defense Advanced Research Projects Agency (DARPA) under grant N66001-17-1-4042.

REFERENCES

[1] Omid Abari, Hariharan Rahul, and Dina Katabi. 2015. AirShare: Distributed Coherent Transmission Made Seamless. *IEEE INFOCOM* (2015).

[2] Horia Vlad Balan, Ryan Rogalin, Antonios Michaloliakos, Konstantinos Psounis, and Giuseppe Caire. 2013. AirSync: Enabling distributed multiuser MIMO with full spatial multiplexing. *IEEE/ACM ToN* 21, 6 (2013), 1681–1695.

[3] Ertugrul Basar, Marco D. Renzo, Julien de Rosny, Merouane Debbah, Mohamed-Slim Alouini, and Rui Zhang. 2019. Wireless Communications Through Reconfigurable Intelligent Surfaces. *IEEE Access* 7 (2019), 116753–116773.

[4] Patrick Bidigare, Miguel Oyarzyn, David Raeman, Dan Chang, Dave Cousins, and D. Richard Brown. 2012. Implementation and Demonstration of Receiver Coordinated Distributed Transmit Beamforming across an Ad-hoc Radio Network. In *Proc. of Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*. Pacific Grove, CA.

[5] Carlos Bocanegra, Kubra Alemdar, Sara Garcia, Chetna Singhal, and Kaushik R Chowdhury. 2019. NetBeam: Networked and Distributed 3-D Beamforming for Multi-user Heterogeneous Traffic. In *2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN)*. IEEE, 1–10.

[6] Yong Soo Cho, Jaekwon Kim, Won Young Yang, and Chung G. Kang. 2010. *MIMO-OFDM Wireless Communications with MATLAB, Chapter 5.* Wiley Publishing.

[7] D. Dardari, A. Conti, U. Ferner, A. Giorgetti, and M. Z. Win. 2009. Ranging With Ultrawide Bandwidth Signals in Multipath Environments. *Proc. IEEE* 97, 2 (2009), 404–426. https://doi.org/10.1109/JPROC.2008.2008846

[8] DECAWARE. [n.d.]. DW1000 User Manual. https://www.decawave.com/sites/default/files/resources/dw1000_user_manual_2.11.pdf/

[9] Jean-Daniel Deschênes, Laura C Sinclair, Fabrizio R Giorgetta, William C Swann, Esther Baumann, Ian Coddington, and Nathan R Newbury. 2015. Optical two-way time synchronization at the femtosecond level over a 4-km free space link. In *Applications of Lasers for Sensing and Free Space Communications*. Optical Society of America, LTh1C–3.

[10] Analog Devices. [n.d.]. ADF4001 Clock Generator. https://www.analog.com/en/products/adf4001.html/

[11] Analog Devices. [n.d.]. ADF4001 RF Transceiver. https://www.analog.com/en/products/ad9361.html#product-overview/

[12] Adwait Dongare, Patrick Lazik, Niranjini Rajagopal, and Anthony Rowe. 2017. Pulsar: A wireless propagation-aware clock synchronization platform. In *IEEE RTAS*. 283–292.

[13] Jeremy Elson, Lewis Girod, and Deborah Estrin. 2002. Fine-grained network time synchronization using reference broadcasts. *ACM SIGOPS Operating Systems Review* 36, SI (2002), 147–163.

[14] Ettus. [n.d.]. GPSDO. https://kb.ettus.com/GPSDO/

[15] Ettus. [n.d.]. Octoclock. https://www.ettus.com/all-products/octoclock/

[16] FCC. [n.d.]. Part 15 of the FCC rules. https://www.ecfr.gov/cgi-bin/text-idx?SID=eed706a2c49fd9271106c3228b0615f3&mc=true&node=pt47.1.15&rgn=div5#se47.1.15_124/

[17] J-L Ferrant, Mike Gilson, Sebastien Jobert, Michael Mayer, Michel Ouellette, Laurent Montini, Silvana Rodrigues, and Stefano Ruffini. 2008. Synchronous Ethernet: A method to transport synchronization. *IEEE Communications Magazine* 46, 9 (2008), 126–134.

[18] Federico Ferrari, Marco Zimmerling, Lothar Thiele, and Olga Saukh. 2011. Efficient network flooding and time synchronization with glossy. In *Proceedings of the 10th ACM/IEEE International Conference on Information Processing in Sensor Networks*. IEEE, 73–84.

[19] H. T. Friis. 1946. A Note on a Simple Transmission Formula. *Proceedings of the IRE* 34, 5 (1946), 254–256. https://doi.org/10.1109/JRPROC.1946.234568

[20] 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*. 138–149.

[21] I. Guvenc and Z. Sahinoglu. 2005. Threshold-based TOA estimation for impulse radio UWB systems. In *2005 IEEE International Conference on Ultra-Wideband*. 420–425. https://doi.org/10.1109/ICU.2005.1570024

[22] Jiho Han and Deog-Kyoon Jeong. 2009. A practical implementation of IEEE 1588-2008 transparent clock for distributed measurement and control systems. *IEEE transactions on instrumentation and measurement* 59, 2 (2009), 433–439.

[23] Sa Hu, Krishna Chitti, Fredrik Rusek, and Ove Edfors. 2018. User Assignment with Distributed Large Intelligent Surface (LIS) Systems. In *IEEE PIMRC*. 1–6.

[24] IEEE. [n.d.]. IEEE 802.15.4x-2019 - IEEE Standard for Low-Rate Wireless Networks. https://standards.ieee.org/standard/802_15_4x-2019.html/

[25] Suhanya Jayaprakasam, Sharul Kamal Abdul Rahim, and Chee Yen Leow. 2017. Distributed and collaborative beamforming in wireless sensor networks: Classifications, trends, and research directions. *IEEE Communications Surveys & Tutorials* 19, 4 (2017), 2092–2116.

[26] Christoph Lenzen, Philipp Sommer, and Roger Wattenhofer. 2014. PulseSync: An efficient and scalable clock synchronization protocol. *IEEE/ACM Transactions on Networking* 23, 3 (2014), 717–727.

[27] Zhuqi Li, Yaxiong Xie, Longfei Shangguan, R. Ivan Zelaya, Jeremy Gummeson, Wenjun Hu, and Kyle Jamieson. 2019. Towards Programming the Radio Environment with Large Arrays of Inexpensive Antennas. In *Proceedings of the 16th USENIX Conference on Networked Systems Design and Implementation* (Boston, MA, USA) *(NSDI'19)*. USENIX Association, USA, 285–299.

[28] Maciej Lipiński, Tomasz Włostowski, Javier Serrano, and Pablo Alvarez. 2011. White rabbit: A PTP application for robust sub-nanosecond synchronization. In *2011 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*. IEEE, 25–30.

[29] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. 2004. The flooding time synchronization protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*. 39–49.

[30] MATLAB. [n.d.]. MATLAB WLAN Toolbox, 802.11a Procedure Protocol Data Unit. https://www.mathworks.com/help/wlan/ug/wlan-ppdu-structure.html/

[31] Microsemi. [n.d.]. CSAC. https://www.microsemi.com/product-directory/clocks-frequency-references/3824-chip-scale-atomic-clock-csac/

[32] Microsemi. [n.d.]. OCXO. https://www.microsemi.com/product-directory/high-reliability-rugged-oscillators/4847-ocxo/

[33] David L Mills. 1991. Internet time synchronization: the network time protocol. *IEEE Transactions on communications* 39, 10 (1991), 1482–1493.

[34] Raghuraman Mudumbai, D. Richard Brown III, Upamanyu Madhow, and H. Vincent Poor. 2009. Distributed transmit beamforming: challenges and recent progress. *IEEE Communications Magazine* 47, 2 (2009), 102–110.

[35] Danh H Nguyen, Anton Paatelma, Harri Saarnisaari, Nagarajan Kandasamy, and Kapil R Dandekar. 2017. Sub-Microsecond Network Synchronization for Distributed Wireless PHY Protocols. In *Proceedings of the 9th ACM Workshop on Wireless of the Students, by the Students, and for the Students*. 3–5.

[36] NIST. [n.d.]. Radio Station WWVB. https://www.nist.gov/pml/time-and-frequency-division/radio-stations/wwvb/help-wwvb-radio-controlled-clocks/

[37] Françis Quitin, Muhammad M. U. Rahman, Raghuraman Mudumbai, and Upamanyu Madhow. 2013. A Scalable Architecture for Distributed Transmit Beamforming with Commodity Radios: Design and Proof of Concept. *IEEE TWC* 12, 3 (March 2013), 1418–1423.

[38] Hariharan Rahul, Haitham Hassanieh, and Dina Katabi. 2010. SourceSync: a distributed wireless architecture for exploiting sender diversity. *ACM SIGCOMM Computer Communication Review* 40, 4 (2010), 171–182.

[39] Hariharan Rahul, Swarun Kumar, and Dina Katabi. 2012. MegaMIMO: Scaling Wireless Capacity with User Demands. In *ACM SIGCOMM 2012*. Helsinki, Finland.

[40] S. Mohammad Razavizadeh and Tommy Svensson. 2020. 3D Beamforming in Reconfigurable Intelligent Surfaces-assisted Wireless Communication Networks. (2020). arXiv:cs.IT/arXiv:2001.06653v1

[41] RFClock. [n.d.]. Github Repository, "RFCLOCK Design and Implementation Files". https://github.com/dDreamCatcher/RFCLOCK

[42] Bashar Romanous, Naim Bitar, Ali Imran, and Hazem Refai. 2015. Network densification: Challenges and opportunities in enabling 5G. In *2015 IEEE 20th International Workshop on Computer Aided Modelling and Design of Communication Links and Networks (CAMAD)*. 129–134.

[43] Vaibhav Singh, Susnata Mondal, Akshay Gadre, Milind Srivastava, Jeyanandh Paramesh, and Swarun Kumar. 2020. Millimeter-Wave Full Duplex Radios. In *Proceedings of the 26th Annual International Conference on Mobile Computing and Networking* (London, United Kingdom) *(MobiCom '20)*. Association for Computing Machinery, New York, NY, USA, Article 2, 14 pages. https://doi.org/10.1145/3372224.3380879

[44] Essam Sourour, Hussein El-Ghoroury, and Dale McNeill. 2004. Frequency offset estimation and correction in the ieee 802.11 a wlan. *Vehicular Technology Conference* 7, SI (2004), 4923–4927.

[45] O. Tervo, H. Pennanen, D. Christopoulos, S. Chatzinotas, and B. Ottersten. 2018. Distributed Optimization for Coordinated Beamforming in Multicell Multigroup Multicast Systems: Power Minimization and SINR Balancing. *IEEE Transactions on Signal Processing* 66, 1 (2018), 171–185. https://doi.org/10.1109/TSP.2017.2762289

[46] Vivek Yenamandra and Kannan Srinivasan. 2014. Vidyut: exploiting power line infrastructure for enterprise wireless networks. *ACM SIGCOMM Computer Communication Review* 44, 4 (2014), 595–606.