



Computer Science and Artificial Intelligence Laboratory
Technical Report

MIT-CSAIL-TR-2014-010

April 27, 2014

**One Clock to Rule Them All: A Primitive
for Distributed Wireless Protocols at the
Physical Layer**

Omid Abari, Hariharan Rahul, and Dina Katabi

One Clock to Rule Them All: A Primitive for Distributed Wireless Protocols at the Physical Layer

Omid Abari Hariharan Rahul Dina Katabi
Massachusetts Institute of Technology
{abari, rahul, dina}@csail.mit.edu

ABSTRACT

Implementing distributed wireless protocols at the physical layer today is challenging because different nodes have different clocks, each of which has slightly different frequencies. This causes the nodes to have frequency offset relative to each other, as a result of which transmitted signals from these nodes do not combine in a predictable manner over time. Past work tackles this challenge and builds distributed PHY layer systems by attempting to address the effects of the frequency offset and compensating for it in the transmitted signals. In this paper, we address this challenge by addressing the root cause - the different clocks with different frequencies on the different nodes. We present AirClock, a new wireless coordination primitive that enables multiple nodes to act as if they are driven by a single clock that they receive wirelessly over the air. AirClock presents a synchronized abstraction to the physical layer, and hence enables direct implementation of diverse kinds of distributed PHY protocols. We illustrate AirClock’s versatility by using it to build three different systems: distributed MIMO, distributed rate adaptation for wireless sensors, and pilotless OFDM, and show that they can provide significant performance benefits over today’s systems.

1. INTRODUCTION

Distributed processing is common at higher layers of the network stack. Yet, at the physical layer, each node typically performs its signal processing independently of other nodes in the network. This architecture is not due to lack of benefits from distributed computation. In fact, the ability to coordinate nodes at the physical layer can bring dramatic benefits. Consider for example distributed MIMO, where multiple wireless nodes coordinate their transmissions to emulate one MIMO transmitter with the sum of antennas on all nodes. Such a design scales wireless throughput with the number of transmitters [28, 7]. In fact, the information theory literature presents many examples of theoretical distributed physical layer designs that could bring significant gains if transformed into practical systems, such as distributed lattice coding [23], noisy network coding [19], transmitter cooperation for cognitive networks [20], etc.

The difficulty in building practical distributed PHY systems is fundamental. Theoretical schemes typically require that the different wireless nodes operate in lockstep with

each other, so that the combined output of the wireless system generated from the computations and actions of individual wireless nodes follows a predictable pattern. However, independent wireless nodes have different clocks with different oscillator frequencies. As a result, different nodes will always have an offset in their carrier frequencies (CFO); the CFO causes two signals to rotate with respect to each other. The output signals formed by the distributed wireless system, therefore, no longer follows the desired pattern; for example, signals that are coordinated to be in-phase and combine constructively could become completely out-of-phase over time, and cancel each other. While having a consistent notion of time is important for distributed protocols both at higher and lower layers, the problem becomes significantly harder at the physical layer, where tiny timing errors of few nanoseconds or even less can cause large errors in the distributed protocol [28].

So, why do different nodes have different clock and oscillator frequencies? Today, each node has a local crystal that generates a low frequency sine wave, which is used as a reference clock. This reference clock is fed to a special circuit called a Phase Locked Loop (PLL), which uses it to generate the desired carrier frequency. For example, if the node is a Wi-Fi radio, the crystal might generate a 10 MHz sine wave, which the PLL then upconverts to a center frequency in the 2.4 GHz or 5 GHz bands. The key problem is that reference clocks on different nodes have slight differences in their frequencies, because different crystals naturally have different properties. Since the PLLs on different nodes lock to reference clocks with different frequencies, their output signals have different frequencies, and this leads to frequency offsets between nodes.

Past work on wireless systems deals with the *effects* of different reference clocks on different nodes by estimating the deviations resulting from the frequency offsets, and compensating for them. For example, practical distributed MIMO systems [28, 7] present accurate techniques for estimating relative signal rotations caused by the CFO and compensating for them before transmission. These systems are however complex, incur overhead, and their details are dependent on the specific protocols and applications.

In contrast, this paper explores a solution that addresses the *root cause* of the problem, *i.e.*, the different reference clocks, by replacing these different clocks with a single ref-

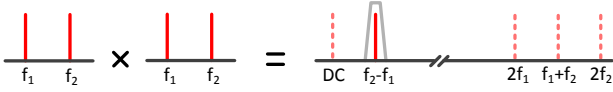


Figure 1—Illustration of AirClock’s Design. Wireless nodes multiply the received signal by itself and extract the desired clock signal by applying a band pass filter centered at f_{ref} .

reference clock that is shared across all the nodes. If wireless nodes were driven by the same reference clock, they would have the same oscillator frequencies, and it would eliminate much of the complexity in building distributed protocols at the physical layer.

One could of course share the reference among multiple wireless nodes by using wires from a single external clock. However, such a system defeats the notion of a wireless network since all nodes are now wired to a central device!

We therefore focus on a design that feeds the shared reference clock to multiple nodes across the wireless medium. At first blush, it might seem that we can simply wirelessly transmit the low frequency wave previously generated by the local crystal on each node, and use it as a shared reference. Each node then feeds this received reference to its PLL. However, such a system is not feasible for two reasons. First, transmitting at such low frequencies (typical crystal generated frequencies are in the range of 10-40 MHz) does not comply with FCC regulations [15]. Second, efficient antennas to capture low frequency signals can be large [8] and, therefore, impractical for many wireless nodes.

We introduce AirClock, a practical system that enables distributed wireless nodes to share a reference clock using the wireless medium. The key idea behind AirClock is to transmit a signal with a specific format such that it can be used to extract a reference clock without any other additional information. AirClock leverages the recently opened white spaces to transmit two single tones separated by the desired clock frequency (e.g., for a clock of 10 MHz, AirClock can send tones at 175 MHz and 185 MHz). To obtain the shared clock, each wireless node multiplies the received signal by itself and applies a band pass filter to extract the desired clock frequency. To see why this system works, recall that the multiplication of two sine waves at different frequencies produces sine waves whose frequencies are the sum and difference of the original frequencies. Specifically, if the two transmitted tones are at frequencies f_1 and f_2 , and the desired clock has a frequency of $f_{ref} = f_1 - f_2$, we have:

$$\begin{aligned} & (\cos(2\pi f_1 t) + \cos(2\pi f_2 t)) \cdot (\cos(2\pi f_1 t) + \cos(2\pi f_2 t)) = \\ & \quad 1 + \mathbf{\cos(2\pi f_{ref} t)} + \cos(2\pi(f_1 + f_2)t) \\ & \quad + \frac{1}{2} \cos(2\pi(f_1 + f_1)t) + \frac{1}{2} \cos(2\pi(f_2 + f_2)t) \end{aligned}$$

Then, by applying a band pass filter centered at f_{ref} , a node can extract the desired clock signal (the term highlighted in bold) and eliminate the undesired frequencies in the signal.¹

¹Note that one cannot share the reference clock by simply unpon-

Fig. 1 illustrates this concept.

Since all nodes extract the reference clock only from the received signal without using any other external signals, they all extract the same reference frequency, thereby eliminating frequency offsets between nodes.

AirClock has several key features. (a) It is **versatile**. Since AirClock transparently provides a synchronized clock abstraction across multiple independent nodes, it enables straightforward implementation of very different kinds of distributed PHY schemes without application or protocol specific mechanisms. (b) Its design is **simple**. Since AirClock only needs to generate a sine wave as a reference clock, it only needs an amplifier, mixer, and filter, and avoids most of the complexity of a typical receiver, such as ADCs, baseband processing such as demodulation, equalization, decoding *etc.* (c) It easily **integrates with standard transceiver architectures**. Specifically, it can be deployed as a standalone module whose output can be plugged in as an input to the PLL, replacing the output of the local crystal.

We demonstrate AirClock’s versatility by showing that its synchronization allows easy implementation of three different kinds of distributed PHY systems.

- **Distributed MIMO without a phase coordination protocol:** Distributed MIMO is a powerful concept that allows multiple distributed wireless transmitters to behave like one huge MIMO transmitter with the number of antennas equal to the sum of antennas on all the cooperating nodes. We show that distributed MIMO systems can use AirClock as a primitive to synchronize the oscillators on the wireless nodes and ensure coherent phase transmission. Such a system eliminates the need for a coherent phase transmission protocol in baseband [28, 7].
- **Distributed Rate Adaptation for Sensors:** Traditional sensor systems such as Zigbee [9] are typically capable of using only a single low transmission rate, and hence cannot exploit good channel conditions to improve wireless throughput. Distributed rate adaptation schemes that allow multiple nodes to transmit simultaneously and thereby exploit good channel conditions, have been designed and implemented for backscatter based networks, such as RFIDs, which face a similar issue [33]. However, such schemes do not work for wireless sensors since, unlike RFIDs, sensors have independent local oscillators, and hence suffer from frequency offsets. By using AirClock, sensor networks can eliminate frequency offsets between nodes, enabling them to leverage backscatter distributed rate adaptation algorithms. We design and implement the first practical distributed rate adaptation system for sensor networks.
- **OFDM without pilots:** OFDM systems like Wi-Fi dedicate some subcarriers for pilots in order to correct for phase drift between the transmitter and the receiver. Since

verting the 10 MHz to a higher frequency at the transmitter and downconverting at the receiver to retrieve the reference clock. This is because downconversion would require the receiver to have a local oscillator to generate the carrier frequency, which would lead to a frequency offset in the retrieved clock signal.

AirClock can synchronize the clocks of transmitters and receivers, we show that it enables us to build an OFDM system that does not need pilots. In addition to the throughput gain from using the pilot frequencies for data, such an architecture also significantly reduces receiver complexity by eliminating the need for phase tracking and compensation algorithms.

We built a prototype of AirClock using off the shelf components and integrated it with USRPs to demonstrate the distributed PHY applications. We evaluated AirClock and its applications in an indoor testbed with line-of-sight and non line-of-sight scenarios. Our results reveal the following findings.

- AirClock can provide tight synchronization across multiple nodes. In particular, in our testbed, the median and 95th percentile CFO between two AirClock nodes operating at 2.45 GHz are 0.38Hz and 1.24Hz respectively.
- Pilotless OFDM using AirClock achieves the same SNR as a traditional pilot based system across the range of operational SNRs for Wi-Fi, thereby providing 8% throughput gain over traditional Wi-Fi.
- AirClock delivers a distributed MIMO system whose throughput scales linearly with the number of transmitters. Specifically, with 5 transmitters, AirClock’s distributed MIMO delivers a median throughput gain of $4.4\times$ over traditional 802.11 style unicast, across the operational range of Wi-Fi SNRs.
- An AirClock based Zigbee system achieves throughput gains over traditional Zigbee by using distributed rate adaptation. Specifically, with 6 Zigbee nodes transmitting jointly to a central node, AirClock provides throughput gains of $1.64 - 3\times$ across a range of SNRs.

Contributions: This work presents the first system that enables multiple wireless nodes to share a clock over the wireless medium, providing a primitive to implement a wide variety of distributed PHY systems. This is in contrast to past work that either shares a clock over the wire, or uses data encoded in a reference signal (e.g., GPS) to extract synchronization information and correct deviations in the local clock signal. In addition to distributed MIMO and pilotless OFDM, we design and implement the first practical distributed rate adaptation scheme for wireless sensors, and realize throughput and reliability gains.

2. RELATED WORK

Related work falls into two categories.

(a) Shared clocks: The most straightforward approach for sharing the clock is to connect the clock input on the nodes to one external clock via wires. In fact, this is how our research community builds MIMO software radios by connecting multiple USRPs to the same clock output. While this approach eliminates any frequency offset or phase noise between the nodes, it effectively transforms the nodes to one device with multiple components connected via wires.

Another traditional solution to synchronize clocks in a wireless network is to use global positioning system (GPS)-disciplined oscillator (GPSDO) to generate the reference signal [17, 11]. A GPSDO is equipped with a high-performance GPS receiver and antenna, a microprocessor and an oven-controlled crystal oscillator (OCXO). Since GPS signals are accurate to the nanoseconds, they provide a good reference to compare the frequency of the crystal oscillator and estimate its offset from its nominal value. Since the frequency of the crystal oscillator is a function of the temperature, this offset can be corrected by precise adjustment of the oven temperature. Due to their complexity, these clock generators cost thousands of dollars which makes them unfeasible to be used in each wireless node. In addition, line of sight to the GPS satellite is required, which make these clocks generator applicable only to outdoor applications. In contrast, AirClock does not require expensive hardware and can also work in indoor applications.

There are also terrestrial time signals such as WWV and WWVB [25, 24], transmitted by the National Institute of Standards and Technology, which can synchronize receivers on the order of seconds. Similarly, traditional clock synchronization schemes like NTP can synchronize nodes to within 10s of milliseconds. However, unlike AirClock, these schemes are too coarse grained for PHY systems, which require synchronization on the order of nanoseconds.

Many communication systems, such as PCI Express and FireWire [10, 4], transmit a shared clock with the data to allow the receiver to synchronize its clock with the transmitter. However, unlike in AirClock, these schemes connect the transmitter to the receiver over a wired link and do not apply to wireless systems, or to multiple transmitters.

(b) Phase tracking and compensation: The typical approach for dealing with frequency offset in wireless nodes is to estimate the resulting phase rotation and compensate for it. In particular, traditional OFDM transmitters send pilots in some subcarriers in the data stream, which OFDM receivers then use to track the phase drift of the transmitted signal and compensate for it. For distributed MIMO, practical schemes that can achieve only diversity [26] or can perform both diversity and multiplexing (e.g. MegaMIMO and AirSync [7, 28]) incorporate sophisticated schemes to track transmitter phase drift and correct signals prior to transmission. These schemes are necessary to ensure that the signals from different transmitters combine coherently at each receiver. In contrast, with AirClock, wireless nodes synchronizes their clocks over the air eliminating the need for phase tracking and compensation either at the receiver or other transmitters and can achieve both diversity and multiplexing gain. Further, AirClock’s system is independent of the protocol, and modulation schemes, and applies to various wireless technology (WiFi, Bluetooth, Zigbee, etc.).

3. BACKGROUND

Wireless signals are transmitted at a particular carrier fre-

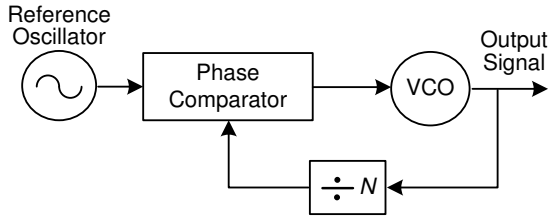


Figure 2—Block diagram of a typical Phase Locked-Loop(PLL). The frequency of the output signal produced by the PLL is N times that of the reference signal.

quency. The signal is up-converted from baseband to the carrier RF frequencies at the transmitter and then down-converted back to the baseband at the receiver. Both up-conversion and down-conversion are performed by a process called mixing, where the signal is multiplied by the carrier frequency. At the transmitter, the mixed signal is transmitted over the air. At the receiver, the received signal is first mixed with the carrier then passed through a low pass filter to obtain the downconverted baseband signal. Both transmitter and receiver therefore need to generate the RF carrier with a precise and stable frequency.

In most communication systems, a phase locked-loop (PLL) synthesizer generates this carrier signal. The PLL takes a low-frequency reference clock signal as input, and produces as output a signal at the desired frequency. The reference signal is typically generated by a crystal resonator. For example, in Wi-Fi systems, the PLL typically uses either a 10 MHz or 40 MHz crystal, and outputs the desired carrier frequency in the 2.4 GHz or 5 GHz bands.

Figure 2 shows the block diagram of a PLL. At a high level, a PLL consist of a voltage-controlled oscillator (VCO), a phase comparator, and a configurable divider. The PLL works as a feedback loop. The VCO generates the desired carrier. The carrier frequency is typically an integer multiple, N , of the reference clock frequency. Thus the output of the VCO is divided N and compared with the reference signal. If there is any phase difference, the PLL adjusts the VCO to keep it around the desired frequency.

In a perfect wireless network, all nodes should use exactly the same carrier frequency. However, today’s wireless networks such as Wi-Fi networks typically have offsets between the carrier frequencies of nodes. For example in today’s Wi-Fi cards, this offset can be tens to hundreds of KHz [1]. This offset is mainly due to the fact that crystals used in different nodes are slightly different. In addition, due to temperature changes, the reference frequencies of different crystal oscillators vary, which results in differences in the PLL output.

4. AIRCLOCK

AirClock synchronizes distributed wireless nodes by driving their oscillators with a single reference signal that they receive over the wireless medium. Each wireless node extracts its reference clock from the reference signal and uses this clock to generate its carrier wave. Since all nodes extract their reference clock from the same signal, their gener-

ated carrier waves do not have any frequency offset relative to each other. These synchronized nodes can then act as one virtual node with many antennas, and can be leveraged to build different kinds of distributed PHY systems.

In the rest of this section, we explain how AirClock works in greater detail. We start with a formal description of the structure of the reference signal and how AirClock can robustly extract a reference clock from it. We then present the actual details of AirClock’s architecture, and describe extensions to it.

4.1 How Does AirClock Work?

AirClock ensures that clocks on different wireless nodes are synchronized by using the same reference signal for the PLLs on the different wireless nodes, and feeding this reference signal over the air. However, one cannot directly feed a reference signal, on the order of 10s of MHz over the air, given the current spectrum allocation. FCC regulations forbid the use of these low frequencies for unlicensed transmission [15]. Besides, transmitting and receiving a signal at this low frequency requires antennas that are many meters long [8], which is impractical for many wireless scenarios.

On the other hand, one cannot transmit the signal simply by upconverting to an unlicensed band at the clock transmitter and downconverting to the reference signal at the clock receiver, as this introduces a chicken-and-egg problem. Up-conversion and downconversion to a band will require independent carrier generation at the transmitter and receiver. Since the reference signals for these independent carriers are generated by different crystals, they will have frequency offset relative to each other, leading to a frequency offset in the downconverted reference signal at different nodes.

AirClock overcomes these challenges by transmitting two single-tones (i.e. sine waves) at frequencies f_1 and f_2 , which are separated by the desired reference clock frequency, f_{ref} , (e.g., 10 MHz), to act as the synchronizing clock signal. AirClock leverages the recently opened white spaces for the frequencies f_1 and f_2 . For instance, for an f_{ref} of 10 MHz, AirClock can use tones at frequencies of, say, 175 MHz and 185 MHz. The transmitted signal can then be written as:

$$S_{tx}(t) = A_1 \cos(2\pi \cdot f_1 \cdot t) + A_2 \cos(2\pi \cdot f_2 \cdot t). \quad (1)$$

This signal passes over the wireless channel before reception, and the received wireless signal can be written as:

$$S_{rx}(t) = B_1 \cdot \cos(2\pi \cdot f_1 \cdot t + \phi_1) + B_2 \cdot \cos(2\pi \cdot f_2 \cdot t + \phi_2), \quad (2)$$

where B_1 , B_2 , ϕ_1 and ϕ_2 capture the channel impact on the signal.

The signal can then be mixed, i.e., multiplied, with itself. This is done by splitting the signal and feeding it to the two inputs of a mixer. This results in:

$$\begin{aligned} S_m(t) &= [B_1 \cdot \cos(2\pi \cdot f_1 \cdot t + \phi_1) + B_2 \cdot \cos(2\pi \cdot f_2 \cdot t + \phi_2)]^2 \\ &= [B_1 \cdot \cos(2\pi \cdot f_1 \cdot t + \phi_1)]^2 + [B_2 \cdot \cos(2\pi \cdot f_2 \cdot t + \phi_2)]^2 \\ &\quad + 2B_1 \cdot B_2 \cdot \cos(2\pi \cdot f_1 \cdot t + \phi_1) \cdot \cos(2\pi \cdot f_2 \cdot t + \phi_2) \end{aligned} \quad (3)$$

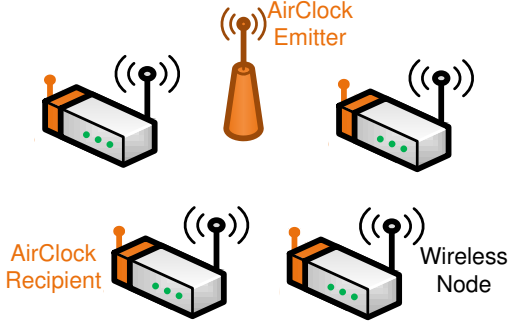


Figure 3—AirClock’s Network Topology. An AirClock emitter transmits the reference signal. Nearby wireless nodes that are equipped with AirClock recipient components (indicated in orange) receive the AirClock signal, extract the reference clock and use it to generate their carrier waves.

Simplifying this equation results in:

$$\begin{aligned}
 S_m(t) &= \mathbf{B_1 B_2 \cos(2\pi \cdot (f_2 - f_1) \cdot t + (\phi_2 - \phi_1))} \\
 &+ B_1 B_2 \cos(2\pi \cdot (f_2 + f_1) \cdot t + (\phi_2 + \phi_1)) \\
 &+ \frac{B_1^2}{2} + \frac{B_2^2}{2} \cdot \cos(2\pi \cdot 2f_1 \cdot t + 2\phi_1) \\
 &+ \frac{B_2^2}{2} + \frac{B_2^2}{2} \cdot \cos(2\pi \cdot 2f_2 \cdot t + 2\phi_2)
 \end{aligned} \quad (4)$$

This signal includes a DC component, some high frequency components at $2f_1$, $2f_2$ and $f_1 + f_2$, and a component at $f_2 - f_1$ (highlighted in bold in the formula above) which is equal to f_{ref} . Hence, a simple band-pass filter centered at the reference clock frequency f_{ref} (e.g., 10 MHz) is used to extract the single-tone reference signal. The signal after the filter will be:

$$\begin{aligned}
 S_{ref}(t) &= B_1 B_2 \cos(2\pi \cdot (f_2 - f_1) \cdot t + (\phi_2 - \phi_1)) \\
 &= B_1 B_2 \cos(2\pi \cdot f_{ref} \cdot t + \delta\phi).
 \end{aligned} \quad (5)$$

This signal is then used as an external reference clock for a wireless node. Since all nodes have access to the same signals, their reference clocks have exactly the same frequency $f_{ref} = f_2 - f_1$ and they will have no CFO with respect to each other. Further, even if the transmitted signal slightly varies, the nodes will still be synchronized since all nodes generate their reference clock using the same signal.

4.2 AirClock’s System Architecture

In this section, we describe how the mathematical idea from the previous section is transformed to a system that can be incorporated into wireless transmitters and receivers.

AirClock has two components: an emitter and a recipient. Fig. 3 demonstrates how AirClock works in a wireless network, at a high level. There is a single emitter in the network which wirelessly transmits a signal, called the AirClock signal. This signal may be used by wireless nodes to generate a shared reference clock. Each wireless node in the network is equipped with an AirClock recipient which receives the AirClock signal from the air, extracts the clock reference and feeds it to the wireless node. Of course, we would like the

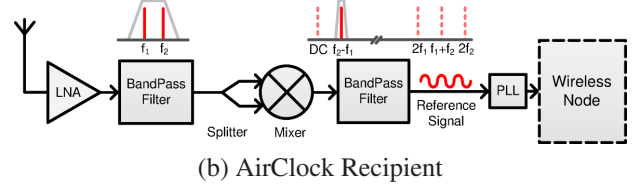
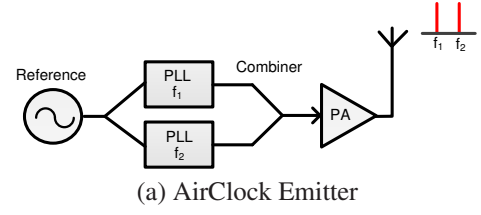


Figure 4—AirClock’s System Architecture. The AirClock emitter includes two PLLs to generate two single tones separated by the desired frequency. The AirClock recipient mixes the received AirClock signal with itself and extracts the reference clock using a band pass filter.

AirClock recipient to be simple and cheap so that it may be used to augment virtually any wireless node including small cheap sensors.

We now describe the architecture of the emitter and recipient components in more detail:

Emitter: Fig. 4(a) presents the system architecture of the AirClock emitter. To transmit the AirClock signal, we start with a local oscillator (*i.e.*, a crystal) that generates a reference signal, and feed its output to two PLLs to generate two tones, f_1 and f_2 , that are separated by the desired clock frequency f_{ref} . The two tones are then amplified using a power amplifier and transmitted on the wireless medium. Note that our signal does not occupy the entire band between f_1 and f_2 , it simply consists of two single-tones which are separated by the desired clock frequency.

Recipient: Fig. 4(b) presents the system architecture of the AirClock recipient. The AirClock recipient receives the emitted signal. As is usual in RF radios, the received signal is passed to a low-noise amplifier (LNA) and the band of interest (from f_1 to f_2) is filtered out using a band pass filter. After filtering, the signal is mixed with itself and the desired reference clock is extracted using a band pass filter centered at the reference frequency (e.g., 10 MHz), and input to a PLL whose output is fed to the wireless node.

The design is simple and cheap. In particular, the receiver circuit does not need any analog to digital converters (ADC) or baseband processing. It is composed of cheap and off-the-shelf components such as amplifiers, band pass filters, a splitter, and a mixer, and can be built in a straightforward manner. Further, the splitter, mixer and filters are all passive components and do not consume any power.

4.3 Scalability

So far, we have explained a situation where all nodes are in the range of a single AirClock emitter. We now discuss how we can scale AirClock to large deployments. Specifically,

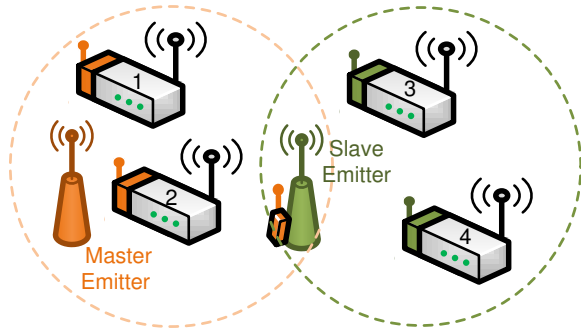


Figure 5— AirClock’s is scalable to large networks. In a scenario where nodes are far from the master AirClock emitter, AirClock uses a slave emitter to regenerate the clock signal. Nodes can synchronize to either the master or slave emitter, and still achieve tight synchronization with each other.

what if not all nodes are within the radio range of a single emitter? Of course, in many applications such as distributed MIMO deployed in a conference room or a big classroom, where all nodes are in the range of an emitter, a single emitter is sufficient to synchronize all wireless nodes. However, in some other applications or scenarios where nodes are far from each other, a single emitter might not be sufficient to cover the whole network and hence we cannot directly synchronize all nodes with a single reference signal.

Here, we explain how we scale AirClock in a larger network. Naively, one might think that we can use multiple emitters in different locations to guarantee that each wireless node receives the AirClock signal. However this solution does not work because each AirClock emitter has a different reference oscillator (i.e., crystal) that it uses to generate its AirClock signal. As a result, signals from different AirClock emitters will not have exactly the same frequency. Consequently, nodes which use AirClock signals transmitted from two different emitters will have CFO with respect to each other, which defeats the purpose of AirClock.

We therefore employ a different design that addresses this issue. Specifically, we guarantee that emitters used in a network are synchronized. We do this by making sure that emitters use the same reference to generate their transmitted AirClock signal. We also ensure that different emitters do not interfere with each other by using different frequencies for their transmitted signals.

To understand how our proposed solution works, consider the scenario shown in Fig. 5. Here, we want to synchronize four wireless nodes, where nodes 3 and 4 are far from nodes 1 and 2. The master emitter generates the AirClock signal from a local oscillator (i.e., a crystal). Nodes 1 and 2 are in the range of this emitter and can use the signal transmitted from the master emitter to generate their reference clocks. However, since nodes 3 and 4 are far, they are not in the range of the master emitter and cannot directly receive the AirClock signal. Hence, we use another emitter called a slave emitter to transmit the AirClock signal to them. The architecture of a slave emitter is similar to that of the master emitter. How-

ever, there is one key difference. A slave emitter does not use a local crystal as its reference. Instead, it uses an AirClock recipient which listens to the master’s signal to generate its reference. The slave then produces its two tones using this recipient generated reference. In order to avoid interference between the master and the slave, the frequencies of the AirClock signal generated by the slave emitter need to be different from the frequencies generated by the master emitter. For instance, the slave emitter can use tones at 620 MHz and 630 MHz, while the master emitter uses tones at 175 MHz and 185 MHz.

Note that tones generated by the slave transmitter are separated by the same amount as the tones generated by the master transmitter. This is due to the fact that the reference for the slave emitter signal is generated by using the master emitter signal. Hence the reference clock extracted from the master emitter signal by wireless node 1 will be synchronized with the reference clock extracted from the slave emitter signal by wireless node 3. In general, Cooperating wireless nodes can pick either the master or the slave signal to extract the shared clock, depending on which of the two are in their radio range. In §7.1, we demonstrate that AirClock provides tight synchronization for nodes that use different emitters, and can thus scale to large deployments.

4.4 Discussion

Channel Impact: Similar to any other wireless system, we need to investigate the impact of the wireless channel on AirClock. First, each AirClock recipient experiences a different wireless channel from its AirClock emitter, i.e., the attenuation and phase experienced by the emitter signal to different recipients is different. However, AirClock is robust to these differing channels as can be seen from Eq. 5, which shows that the frequency of the extracted reference clock is f_{ref} independent of the channel attenuation and phase. Hence, wireless nodes which have different channels to the AirClock emitter will be synchronized. We show this to be the case empirically in §7.1.

Non line-of-sight: AirClock works in non-line-of-sight scenarios as well as line-of-sight scenarios. In a non-line-of-sight scenario, an AirClock recipient receives multiple delayed versions of the signal, each of which encounters a different channel. Note that adding multiple copies of signals at f_1 and f_2 , each with a different attenuation and phase, does not change the output frequency of AirClock. Specifically, similar to Eq. 5, the AirClock recipient will still produce an output reference clock with frequency f_{ref} . We demonstrate empirically in §7.1 that AirClock performs tight synchronization in non-line-of-sight scenarios.

Event Synchronization: In addition to frequency synchronization, distributed PHY systems also need event synchronization, for instance, to ensure that multiple nodes begin their transmissions at the same time. Similar to other protocols, distributed PHY systems using AirClock can achieve event synchronization through standard techniques

such as using a trigger packet, calibrating for propagation delays [27] *etc.*

Power Consumption: As already mentioned components used in the AirClock recipient are simple. The splitter, mixer and filters are passive components and they do not require any power. The only active components are the LNA and the PLL, both of which consume significantly lower power than wireless nodes. For example, a typical Wi-Fi access point consumes tens of watts [12] and Zigbee chips consume about 60-110 mW [32, 30]. In contrast, an LNA that operates in the white space frequency range consumes only 7-10 mW [21], and PLLs consume around 3 mW [16, 6]. Thus, adding an AirClock recipient adds around 10-13 mW to the total power consumption of the node, which is less than 0.1% of the total power consumption of a Wi-Fi access point, and only around 10% of the power consumption of a Zigbee node.

Plug-and-play Integration: AirClock's recipient module is a plug-and-play component, and can be easily integrated into today's wireless nodes. Specifically, the node can be designed so that it can use either the reference clock generated by its local crystal or the reference clock supplied by the AirClock recipient as the input to its onboard PLL. This allows nodes to leverage AirClock to achieve the performance benefits from highly accurate frequency synchronization, while still having the option to use the local crystal during normal operation or in fallback mode.

5. APPLICATIONS OF AIRCLOCK

Now that we have described the architecture of AirClock, we demonstrate its versatility by explaining how it enables distributed PHY schemes for different wireless systems. The wireless systems discussed in this section have significant differences from each other in their associated protocols, applications, modulation schemes, transceiver hardware complexity *etc.*. Despite these differences, however, AirClock enables implementation of distributed PHY schemes in a simple manner because it presents a synchronized clock abstraction across separate wireless nodes.

5.1 Distributed MIMO with AirClock: Coherent Transmission for Free

We explore distributed MIMO transmission in dense wireless LAN deployments, popular in enterprises, stadia, convention centers *etc.* These deployments typically consist of multiple access points connected to a high bandwidth backhaul (say, Gigabit Ethernet). In a distributed MIMO architecture, the multiple different access points act as if they are a single combined access point with a total number of antennas equal to the sum of the number of antennas on all the individual access points. This virtual combined access point can then transmit different packets simultaneously to different receivers, by using multi-user beamforming to eliminate interference between the different transmissions, as illustrated in 6.

The key difficulty in implementing distributed MIMO in practice is that joint multi user beamforming across multiple access points requires that the phases of the transmissions from different access points are precisely aligned. This alignment is necessary to ensure that, at every receiver, the signals destined for other receivers cancel out and only the signal intended for that receiver survives the collision. The receiver can then decode its desired signal without suffering from interference from the other signals. When all the antennas belong to a single transmitter, they are driven by the same oscillator, and hence the phases of the different signals are naturally synchronized with each other. However, when different access points driven by different oscillators transmit simultaneously, the phases of the transmissions from the different access points rotate relative to each other due to the frequency offsets between the transmitters. Hence, the phase alignment between the different transmitters changes over time, leading to interference between the different signals and failure of receivers to decode their desired packets.

Existing practical distributed MIMO techniques such as MegaMIMO and AirSync [28, 7] address the problem of changing phase alignment by using training sequences to estimate the changes in phase alignment at each transmitter, and compensating for the changes prior to transmission.

In contrast, AirClock synchronizes the clocks across all the different transmitters, and hence antennas belonging to different transmitters equipped with AirClock recipients can simply act as if they were different antennas driven by the same oscillator on a large MIMO transmitter. They can then simply perform joint multi-user beamforming similar to existing single transmitter schemes, without performing ongoing phase tracking or compensation.

Similar to existing multi-user beamforming systems [5, 28, 7], Distributed MIMO with AirClock operates in two phases: a channel measurement phase, and a joint data transmission phase which uses the measured channels to perform multi-user beamforming. We implement this system and evaluate its performance for data transmission on the downlink in §7.4.

Additional Benefits

Apart from downlink multiplexing gains, distributed MIMO with AirClock can provide additional benefits.

First, it can enable coherent joint transmission on the uplink from multiple clients. We can do this in a system where all nodes, both access points and clients, are equipped with AirClock recipients. Then, the oscillators on the different clients are synchronized and their transmissions appear at the different access points as if they occurred from multiple antennas on the same device. The access points can then exchange the received samples at the different antennas using their shared backhaul, and jointly decode the received packets, similar to the way a single MIMO receiver can decode multiple different streams from a single MIMO transmitter.

Second, it can enable multiple off-the-shelf access points to exploit MIMO diversity. Specifically, it is typically possi-

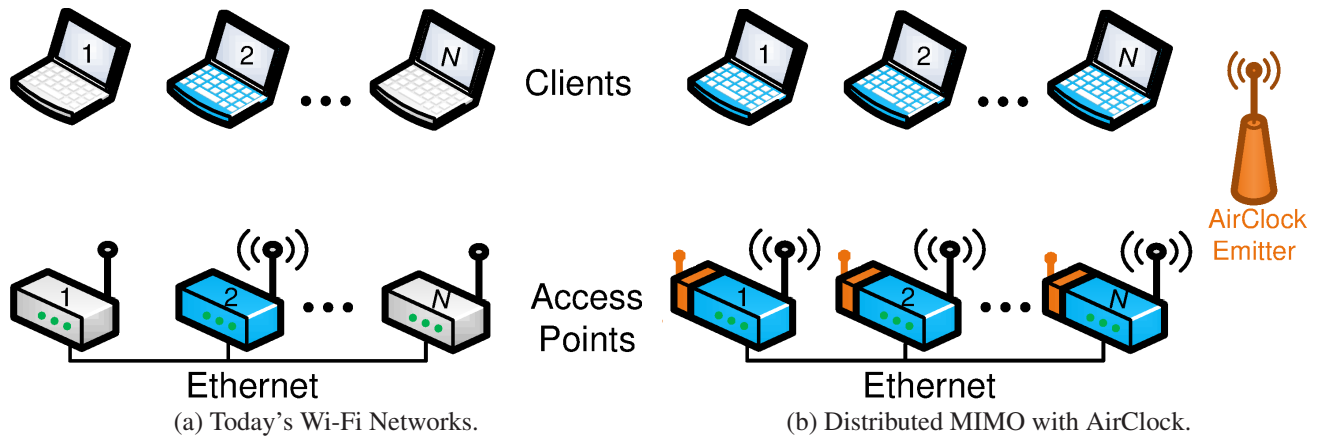


Figure 6—Traditional deployments vs. Distributed MIMO. A blue colored node indicates an active transmitter or receiver. With traditional Wi-Fi, only one access point can transmit at any time in a given frequency. In contrast, with AirClock, the oscillators of all APs equipped with an AirClock recipient are synchronized to the AirClock emitter. Hence, multiple APs can transmit to multiple clients at the same time in the same frequency, thereby scaling network throughput with the number of access points.

ble to feed an external clock to off-the-shelf wireless nodes (e.g., a WiFi access point or a ZigBee sensor). Some devices are designed to have the option of operating with an external clock (e.g., a GPS clock). Even when the device does not directly have a connector for an external clock, it is typically possible to remove the internal crystal and plug in the AirClock circuit. This means that AirClock can synchronize the clocks across multiple off-the-shelf devices. These clocks can then achieve diversity gains by simultaneously transmitting the same data packets. This will enable today’s wireless networks to achieve a higher coverage radius and address dead spots.

5.2 Distributed Rate Adaptation for Wireless Sensors

AirClock enables the first distributed rate adaptation scheme for wireless sensor networks. At a high level, AirClock’s synchronization of multiple sensor nodes eliminates frequency offsets between them, and enables them to leverage distributed rateless codes designed for backscatter networks, which do not have frequency offsets. We describe this idea in greater detail below.

Wireless sensor networks are typically deployed for monitoring environmental and physical conditions [2]. These low power devices periodically collect measurements of some physical quantities (such as temperature, pressure, air quality *etc.*), and transmit them to a sink node.

Today’s sensors typically support only a single modulation scheme, such as on-off keying, BPSK, and OQPSK [13]. The modulations supported are low rate so as to ensure that the sensors can communicate even when channel conditions are adverse. Further, sensors avoid supporting high rate (dense) modulations such as 16-QAM, 64-QAM *etc.*, because these modulations require linear transmitter power amplifiers that consume significant power.

As a result, wireless sensors do not utilize the wireless channel efficiently. In particular, wireless sensors cannot take advantage of a good channel to send at a high rate, *i.e.*,

pack many bits into each transmitted symbol.

One can imagine exploiting channel conditions through distributed rate adaptation across the network to overcome the absence of the ability of any single node to adapt its rate. Prior work [33] has proposed such distributed rate adaptation in the context of backscatter networks like RFIDs. Specifically, multiple RFID nodes in the network can transmit simultaneously and the RFID receiver receives a collided transmission. Consider, for instance, a case where 2 nodes, each using BPSK (*i.e.* +1 or -1), transmit simultaneously. Let the channel between the nodes and the sink be h_1 and h_2 respectively. In such a case, the RFID receiver will receive one of 4 points, h_1+h_2 , h_1-h_2 , $-h_1+h_2$, or $-h_1-h_2$. This can be visualized as a scaled and rotated QPSK constellation, which the RFID receiver can decode if the channel conditions are sufficiently good. If not, it can simply continue to receive additional transmissions and combine these multiple receptions till it can decode the transmitted signals. Such a system will effectively achieve a distributed rateless code across the nodes in the RFID network.

Ideally, one would like to achieve similar distributed rate adaptation for wireless sensor networks. However, there is a key difference between sensor networks and backscatter networks. The nodes in a backscatter network do not have any frequency offset relative to each other. This is because they do not have independent oscillators; instead, they simply reflect, *i.e.*, backscatter, a carrier wave transmitted by the RFID receiver, and communicate data by modifying the way in which they reflect the wave depending on the bit they intend to transmit. In contrast, wireless sensor nodes each have independent oscillators, which drive their RF transmissions. As a result, the transmissions from different sensors rotate in phase relative to each other, and collide in different ways across time. Consequently, the joint constellation formed by the collisions experiences different rotation and scaling over time, preventing the receiver from decoding the transmitted bits from multiple collision receptions.

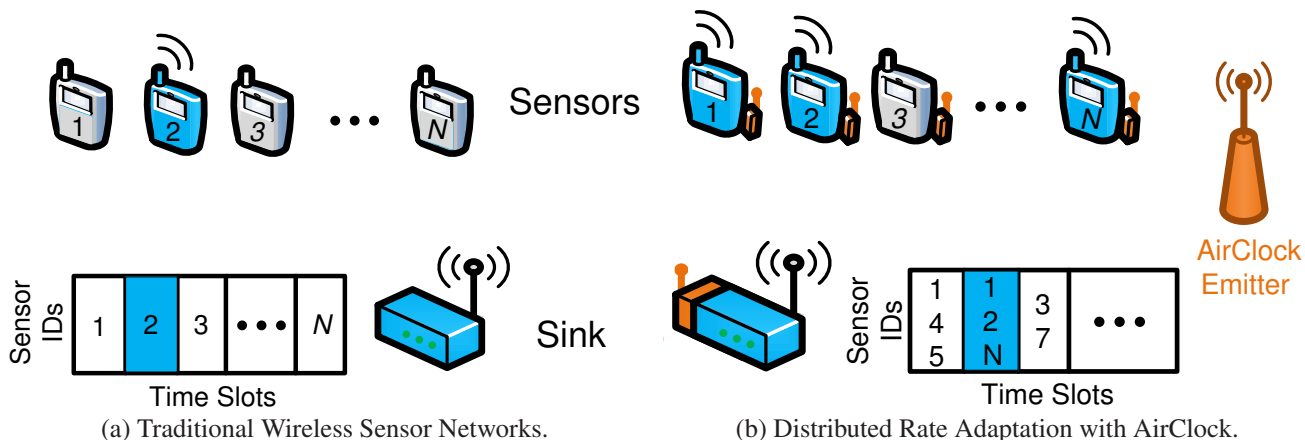


Figure 7—Traditional vs. Distributed Rate Adaptation. Blue indicates active sensors. With traditional sensor networks, only one sensor can transmit at a time, requiring the network to consume N slots to communicate data for N sensors even when channel conditions are good. In contrast, sensors equipped with AirClock can perform distributed rate adaptation allowing multiple sensors to transmit simultaneously (e.g., sensors 1, 2, and N can transmit jointly in the highlighted time slot above). This allows the network to exploit good channel conditions by decoding data from multiple sensors in a single joint transmission, and thereby consuming significantly fewer than N slots to transmit data from all N sensors.

AirClock eliminates the problem of differing frequency offsets across sensors. In particular, when each sensor is equipped with an AirClock recipient, their oscillators are driven by the shared wireless clock and hence they do not have any offset relative to each other. The sensors can therefore perform joint transmission and enable the sink to decode, similar to the case of RFIDs, as shown in Fig. 7.

Note that addition of an AirClock recipient to a sensor has negligible impact on its power consumption. As described in §4.2, the AirClock recipient consumes around 10 mW of power, which is only 10% of the total power consumed by a sensor node during transmission (100 mW).

The full details of the joint transmission algorithm by the nodes, and the receiver’s decoding algorithm are described in [33]. We present an overview here, describing how the transmitters each communicate 1 data bit to the receiver.

The joint transmission algorithm works in multiple phases. The sink initiates joint transmission by sending a data request to all participating sensor nodes. Each sensor receives this command, and in response, generates a random bit (0 or 1) based on its own identifier, and the current phase. If the generated bit is 1, the node participates in the current phase by transmitting its message; if the bit is 0, the node stays silent during the current phase. The sensor nodes repeat this process and continue until they receive an acknowledgment from the sink indicating that it has decoded all the sensor messages.

The receiver receives the collisions, and at each phase, uses all the received collisions so far to jointly decode the transmitted bits. The receiver can determine the participating sensor nodes in each collision, since it knows the identifiers of all the sensors and the shared seed used for the random bit generation. It then uses belief propagation to decode the received collisions. At a high level, the decoder works by starting with an initial guess for the transmitted bits. It then

calculates the expected collided signal for each of the transmissions based on this initial guess, and computes the total error between the expected and the actual received signal. If the initial guess is correct, then the error will be small. If the guess is incorrect, however, the error will be large. If the error is large, the decoder toggles the bits in its initial guess one by one so as to reduce the error. The decoder keeps accumulating collisions till it has decoded all bits.

[33] formally describes this algorithm and provides the pseudocode for the decoder.

5.3 OFDM without Pilots

So far, we have described two applications of AirClock to synchronize multiple distributed transmitters. But AirClock can offer benefits even in the case of a single transmitter, as we describe below. We describe how using AirClock can significantly simplify the design of OFDM based systems while providing performance benefits.

OFDM is the modulation scheme of choice for high data rate communication in modern wireless systems, including WiFi and LTE. The key benefit of OFDM is that it allows wireless nodes to exploit wideband channels and combat frequency selective fading with simple channel equalization algorithms. However, the major challenge that OFDM faces is that it is vulnerable to clock offsets. Today’s OFDM systems address this issue by allocating some subcarriers, called pilots for synchronization. An OFDM receiver uses these pilots to estimate the phase offsets between transmitter and receiver, and correct for them.

Equipping the OFDM transmitter and receiver with AirClock recipients will eliminate the need for an OFDM transmitter to use pilot signals, and for a receiver to implement phase tracking and correction algorithms. Such a system will allow the reuse of pilot subcarriers for data transmission, thereby increasing link throughput. It will also reduce com-

plexity since phase compensation and tracking are among the most complex components in today’s OFDM baseband design.

AirClock can also provide benefits for OFDMA (Orthogonal Frequency Division Multiple Access) schemes, when multiple nodes transmit to a single node. Today’s OFDMA systems require the division of pilots among the multiple transmitters across both frequency and time. In contrast, since AirClock synchronizes clocks of all the participating nodes, it can eliminate the use of these pilots, and the complexity of pilot allocation and scheduling schemes.

6. IMPLEMENTATION

We implement AirClock’s emitter and recipients, and integrate the recipient subsystem with USRPs. We also implement pilotless OFDM, distributed MIMO, and distributed rate adaptation using AirClock with USRPs.

(a) AirClock Emitter and Recipient, and USRP integration: We have built a prototype of the AirClock emitters and recipients as described in 4.2 using the off-the-shelf components. For the emitter, we used the following components: Fury Jackson GPSDO [17] as a local oscillator, two Analog Devices ADF4350 [3] as PLLs, Mini-Circuits TVA-R5-13 [22] as a power amplifier, Laird Technologies EXB-164-BN [18] as an antenna.

For the recipient, we used the following components: Laird Technologies EXB-164-BN [18] as an antenna, Maxim MAX2665 [22] as an amplifier, Mini-Circuits ZLW-1 [22] as a mixer, Mini-Circuits SBP-10.7+ [22] as bandpass filter around 10MHz, Mini Circuits SHP-175+ as a highpass filter and Mini-Circuits SLP-200+ as a lowpass filter to create a bandpass filter around 175-185 MHz, and Silicon Labs Si5328 [29] as a PLL.²

The output of the recipient is connected to the 10 MHz external clock input of the USRP N210 device.

(b) Modulation schemes: For pilotless OFDM and distributed MIMO, we implemented OFDM with the various Wi-Fi modulations (BPSK, 4-QAM, 16-QAM, and 64-QAM) and parameters similar to Wi-Fi (64 total subcarriers with 52 occupied subcarriers) over a bandwidth of 10 MHz. For distributed rate adaptation, we implemented BPSK with a pseudonoise sequence of length 15, similar to Zigbee in the 915 MHz frequency band. We implemented all these schemes using the UHD library supplied for the USRPs.

(c) Distributed MIMO: We implemented Distributed MIMO using software radios. Each AP node is equipped with an RFX 2400 daughterboard, and a USRP N210 [14] integrated with a AirClock recipient as described above. Client nodes consist of USRPs using their internal clocks.

(d) Distributed Rate Adaptation for Wireless Sensors: We

²Note that our prototype uses discrete components for ease of implementation. A production implementation of AirClock would use components and individual chips, and therefore have significantly lower cost and form factor than our prototype.

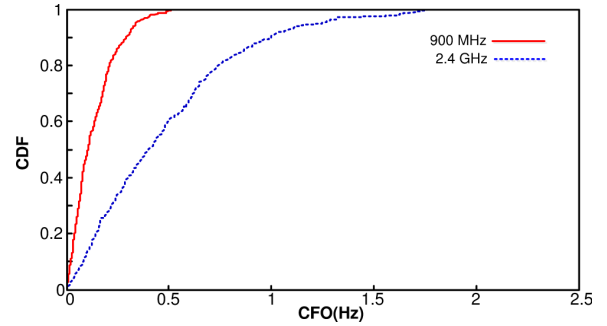


Figure 8—Synchronization Accuracy for 2.4 GHz and 900 MHz carriers. AirClock ensures tight synchronization of oscillators across different nodes. The median CFO is 0.38Hz and the 95th percentile is 1.24Hz for a carrier of 2.4 GHz. For a 900 MHz carrier, the median CFO is 0.11 Hz and the 95th percentile is 0.34 Hz.

used USRP N210s with RFX900 daughterboards to work as wireless sensors and a wireless sink. The sensors transmit over a 2 MHz band, *a la* Zigbee at 915 MHz. Each wireless sensor node is integrated with an AirClock recipient.

7. RESULTS

We evaluate AirClock in an indoor testbed with various line-of-sight and non-line-of-sight scenarios, using both microbenchmarks of its clock synchronization, as well as the performance of applications using AirClock.

7.1 Synchronization

We empirically evaluate the synchronization performance of AirClock in multiple scenarios.

(a) Accuracy of AirClock’s synchronization: The key promise of AirClock is that it can tightly synchronize oscillators on different wireless nodes. We verify if AirClock delivers on this promise.

Method. Our goal in this experiment is to verify the effectiveness of AirClock’s synchronization architecture. To do this, we perform an experiment with all devices in a single room. Specifically, we place an AirClock emitter in one location in the testbed. We place two USRP nodes equipped with AirClock recipients at two node locations in the testbed, with one acting as a transmitter and the other as a receiver. The transmitter transmits packets consisting of a single repeated OFDM symbol. The receiver receives these repeated packets, and computes its CFO with respect to the transmitter using the traditional correlation-based OFDM CFO estimation algorithm [31]. We repeat the experiment for two carrier frequencies: 2.4 GHz and 900 MHz, and for a variety of transmitter receiver locations.

Results. Fig. 8 plots the CDF of the observed CFO for both 2.4 GHz and 900 MHz carriers. It shows that AirClock provides tight synchronization of the oscillators on the transmitter and receiver nodes. Specifically, the median and 95th percentile CFO at 2.4 GHz are 0.4 Hz (0.16 parts per billion) and 1.24 Hz (0.5 parts per billion) respectively. In comparison, the 802.11 standard requires oscillators to have a maxi-

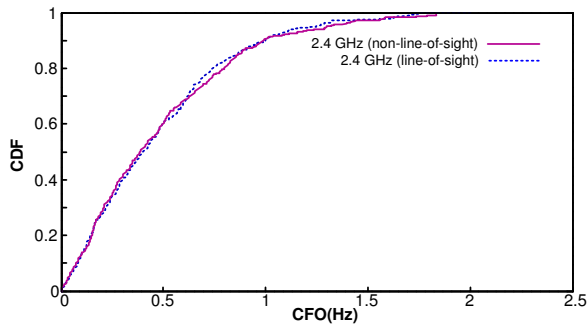


Figure 9—AirClock’s robustness in non-line-of-sight scenarios. CDF of AirClock’s CFO with non-line-of-sight channels. The figure shows that AirClock can achieve tight oscillator synchronization in this scenario, comparable to line-of-sight scenarios.

imum CFO of 40 parts per *million* with respect to each other. AirClock can thus achieve 5 orders of magnitude tighter synchronization than the traditional situation of free running oscillators on independent nodes.

Additionally, the CFO at 900 MHz is less than at 2.4 GHz. In particular, the median and 95th percentile CFOs are 0.11 Hz and 0.34 Hz respectively. This is because the oscillator frequency (2.4 GHz or 900 MHz) is obtained by multiplying the 10 MHz output of the AirClock recipient by an appropriate factor (240 and 90, respectively). Any CFO between the oscillators arises from deviations in the 10 MHz frequency generated by the different AirClock recipients. Therefore, the CFO is higher when the carrier frequency is higher because the errors are scaled up by a larger multiplier.

(b) Robustness of AirClock’s synchronization: We evaluate how robust AirClock’s synchronization in the presence of non-line-of-sight channels.

Method. We place the AirClock emitter such that it has no line of sight to the other nodes in the testbed. We repeat the previous experiment and compute the CFO between the transmitter and receiver at 2.4 GHz.

Results. Fig. 9 shows the CDF of the measured CFO in the presence of non-line-of-sight channels. The median and 95th percentile CFO in this case are 0.39 Hz and 1.3 Hz respectively, showing that AirClock achieves the same tight synchronization performance as line-of-sight channels.

(c) Impact of scaling on AirClock’s synchronization performance: In §4.3, we described how AirClock can scale to large deployments by using slave emitters that generate the 10 MHz clock and repeat it in a different frequency band. In this section, we evaluate the impact of repeating on AirClock’s synchronization performance.

Method. We place an AirClock master emitter and an AirClock slave emitter within radio range of each other. As before, we configure two USRP nodes equipped with AirClock recipients as a transmitter and receiver, respectively. The AirClock recipient on the transmitter receives its synchronization signal from the master emitter, and the AirClock recipient on the receiver receives its synchronization signal from the slave emitter. As before, we compute the CFO between

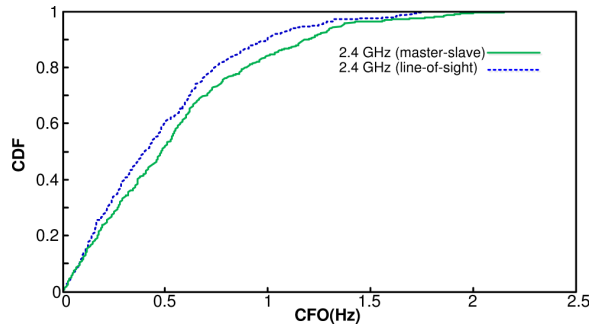


Figure 10—AirClock’s synchronization accuracy with repeaters. CDF of AirClock’s CFO when using slave repeaters. The figure shows that AirClock’s synchronization performance is close to that of a single emitter network, demonstrating that AirClock can scale to deployments larger than the range of a single emitter.

the transmitter and the receiver.

Results. Fig. 10 plots the CDF of the observed CFO. The median and 95th percentile CFO are 0.48 Hz and 1.37 Hz respectively. The 95th percentile is within 10% of the same statistic for a single emitter scenario, demonstrating that AirClock’s architecture can offer tight synchronization even for deployments larger than can be covered by a single emitter.

7.2 Accuracy of Received Signals with AirClock

Having understood AirClock’s oscillator synchronization performance, we now investigate the impact of its synchronization on the accuracy of the received signal.

Method. As before, we place a AirClock emitter and two USRP nodes at random locations in our testbed. One of the USRPs acts as a transmitter and the other as a receiver. We choose a modulation scheme, and transmit OFDM packets carrying random data using that modulation scheme. We then estimate the received signal after channel equalization on the receiver. We perform the experiment with both high and low-density modulation schemes, specifically BPSK and 64-QAM. We repeat the experiment with three different synchronization modes: (a) transmitter and receiver using independent oscillators driven by their local crystals, (b) transmitter and receiver using a single external clock shared over a wire, and (c) transmitter and receiver using AirClock.

Results. Fig. 11 plots the received constellation diagram for BPSK and 64-QAM in the different synchronization scenarios. As can be seen, the transmit and receive oscillators have significant CFO with each other when using their local crystals. As a result, the observed constellation rotates over time. In contrast, the observed constellation points with AirClock are tightly clustered around the expected points, similar to the case when they are both connected to a single external clock over the wire. This provides visual evidence that the receiver can accurately decode the received signal without any frequency offset estimation or compensation, even for dense constellations.

7.3 Pilotless OFDM

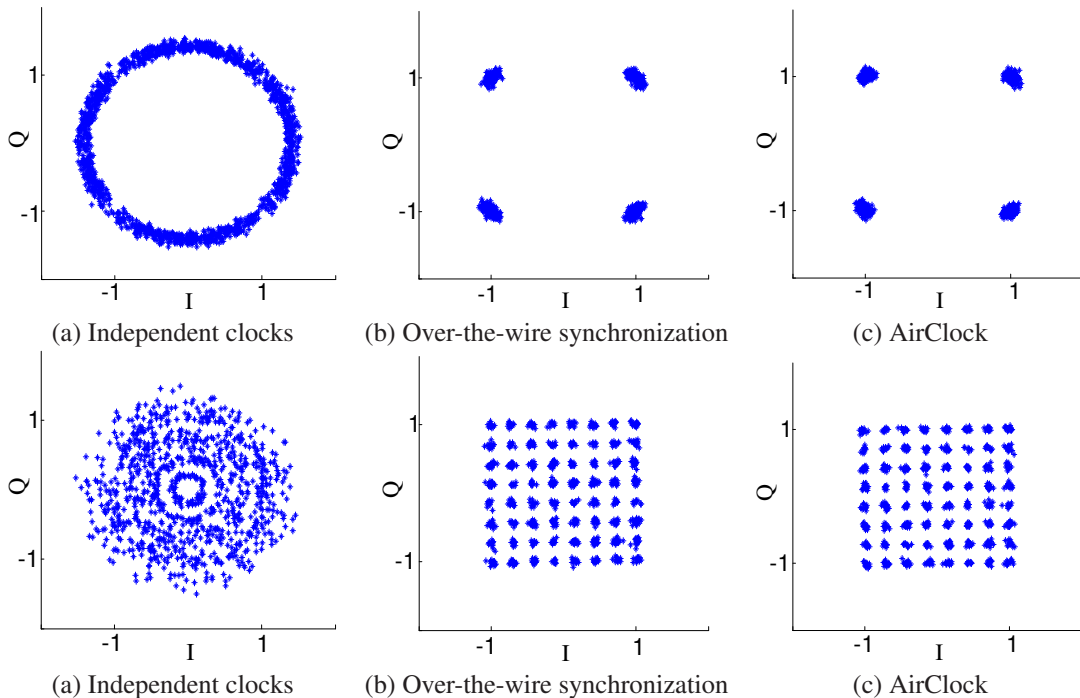


Figure 11—Received constellation points for 4-QAM and 64-QAM in different synchronization scenarios. With different crystals, the transmitter and receiver are not synchronized with each other leading to rotation of the constellation points. Decoding the received points without error would therefore require phase tracking and compensation. In contrast, while using over-the-wire synchronization, the received constellation points after equalization do not rotate relative to each other, showing that the transmitter and receiver oscillators are synchronized. AirClock can provide performance similar to over-the-wire synchronization by using a wireless reference signal.

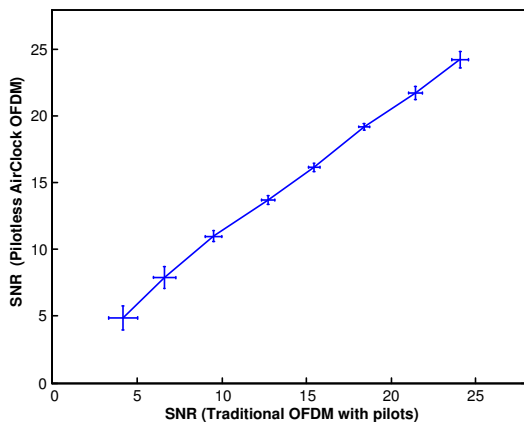


Figure 12—Efficiency of AirClock based pilotless OFDM. For each location, the figure plots the SNR obtained by traditional OFDM performing frequency compensation with pilots, and AirClock based OFDM that does not use pilots. AirClock based OFDM does not require frequency offset estimation and correction unlike traditional OFDM, yet it can achieve the same SNR as traditional OFDM across the entire operational range of Wi-Fi. This allows AirClock to eliminate OFDM complexity, and obtain throughput gain by using the pilots for data transmission.

We investigate the previous result quantitatively in the context of pilotless OFDM.

Method. As before, we place a AirClock emitter and two USRP nodes equipped with AirClock recipients acting as transmitter and receiver in random locations in our testbed.

For each location, we transmit OFDM packets with known random data. Further, at each location, we perform the experiment with the transmitter and receiver in two synchronization scenarios: (a) transmitter and receiver using independent oscillators driven by their local crystals, and (b) transmitter and receiver using AirClock. We repeat the experiment across different locations to achieve the entire range of operational SNRs of Wi-Fi (5–25 dB).

For synchronization scenario (a), we treat subcarriers -21, -7, 7, and 21 as pilots, similar to Wi-Fi, and use them for frequency offset tracking and compensation as in traditional OFDM.

For scenario (b), we treat all subcarriers as data subcarriers and do not perform any frequency offset tracking or compensation during the packet. We simply compute the channel using header estimation symbols as in traditional OFDM, and use this estimated channel for decoding the entire packet. We compute the decoded data in each case, and calculate the SNR achieved in the two scenarios.

Results. Fig. 12 plots the received SNR using pilotless OFDM in synchronization scenario (b) as a function of the received SNR using traditional OFDM with synchronization scenario (a) at each location. As can be seen, AirClock without pilots can achieve the same performance as traditional OFDM with frequency offset tracking and compensation. This enables significant reduction in OFDM system complexity, and provides throughput gains ($\approx 8\%$ for 802.11n)

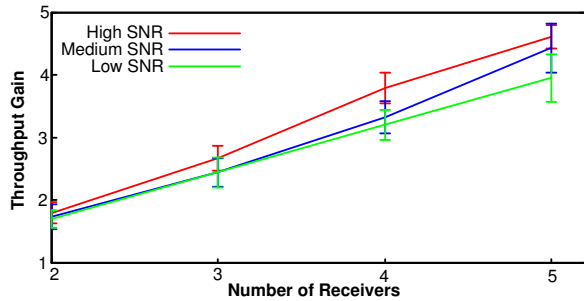


Figure 13—Scaling of network throughput with the number of APs. AirClock’s throughput gain increases linearly with the number of transmitter-receiver pairs in the network. This is because the total network throughput of traditional 802.11 remains constant even as the number of transmitters increases, whereas distributed MIMO allows all transmitters to deliver their packets simultaneously to all receivers.

because AirClock can reuse the pilot subcarriers for data.

7.4 Distributed MIMO with AirClock

In this section, we investigate if AirClock can enable distributed MIMO “for free”, *i.e.*, if it can enable practical distributed MIMO systems without the need for phase tracking and compensation algorithms to synchronize transmitters.

Method. As in all cases, we place a AirClock emitter in our testbed. We then place USRP nodes equipped with AirClock recipients to act as APs and clients in our testbed. Similar to the evaluation scenario in [28], we evaluate distributed MIMO in three different SNR regimes: low (5-10 dB), medium (10-16 dB), and high (> 16 dB). Since USRPs cannot perform carrier sense due to high software latency, we evaluate the performance of traditional 802.11 by scheduling each transmitter so that it gets its fair share of the medium. We repeat the experiment for different node placements and different number of transmitter-receiver pairs.

Results. Fig. 13 plots the throughput gain obtained by distributed MIMO using AirClock as a function of the number of APs, for different SNR ranges. As can be seen, AirClock enables the wireless network throughput to scale with the number of transmitter-receiver pairs, for a gain of $3.95 - 4.61\times$ across the range of SNRs. This is because, with traditional 802.11, only one transmitter-receiver pair is active at any time irrespective of the number of transmitters. In contrast, distributed MIMO enables all transmitters to transmit jointly to their desired receivers without interfering with each other, thus achieving throughput proportional to the number of active transmitters. This shows that, by using AirClock, distributed MIMO systems can achieve results similar to prior work [28] without the need for phase tracking and compensation algorithms.

7.5 Distributed Rate Adaptation for Wireless Sensors

In §5.2, we describe how AirClock’s synchronization primitive enables the implementation of distributed rate adaptation for sensor networks, similar to distributed rate

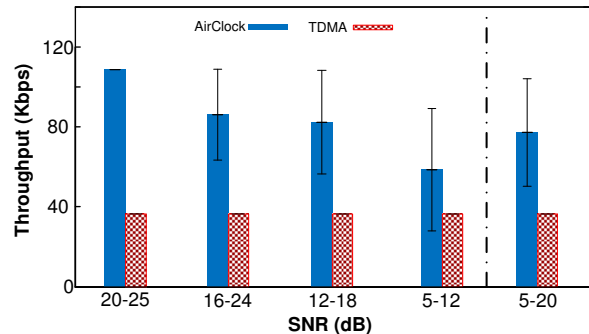


Figure 14—Channel Quality versus Throughput for Distributed Rate Adaptation. AirClock enables distributed rate adaptation for wireless sensors. it delivers a throughput gain between $1.64 - 3\times$ for a network of 6 wireless sensors

adaptation for backscatter networks. We investigate the performance of such distributed rate adaptation.

Method. As in other experiments, we deploy an AirClock emitter in the testbed. We deploy 6 USRP nodes equipped with AirClock implementing the Zigbee protocol and acting as sensors, and one USRP node with AirClock acting as a sink. We compare distributed rate adaptation with AirClock with a TDMA scheme, where only one sensor transmits at a time, and the different sensors transmit one after the other. We evaluate both schemes in various SNR ranges.

Results. Fig. 14 plots the throughput of distributed rate adaptation using AirClock, and of traditional TDMA, for different SNR ranges. Distributed rate adaptation can achieve $1.64 - 3\times$ the throughput of traditional TDMA. Since traditional TDMA cannot take advantage of good channel conditions to increase its transmission rate, the throughput of traditional TDMA is constant independent of the SNR. In contrast, distributed rate adaptation can exploit good channel conditions by allowing the receiver to decode multiple simultaneous transmitters from a single collision. Since the receiver can decode more simultaneous transmitters as the SNR increases, the throughput gain of distributed rate adaptation increases with the average SNR of the network, similar to how the throughput of traditional rate adaptation increases with increasing link SNR.

8. CONCLUSION

This paper presents AirClock, a system that can synchronize multiple nodes using a reference signal that they receive over the wireless medium. AirClock has a simple, low-cost, low-power architecture that easily integrates with today’s wireless transceivers. AirClock provides an abstraction that multiple different wireless nodes are driven by the same reference clock, and hence enables straightforward implementation of a variety of distributed PHY algorithms, including distributed MIMO and distributed rate adaptation for sensor networks. We believe that AirClock’s synchronization primitive can serve as a building block that brings a large body of distributed information theoretic schemes closer to practice.

REFERENCES

- [1] IEEE standard for information technology-specific requirements part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications amendment 5: Enhancements for higher throughput. *IEEE*, pages c1–502, 29 2009.
- [2] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer networks*, 38(4):393–422, 2002.
- [3] Analog Devices, “Analog Devices PLL”. <http://www.analog.com/>.
- [4] D. Anderson, T. Shanley, and R. Budruk. *PCI express system architecture*. Addison-Wesley Professional, 2004.
- [5] E. Aryafar, N. Anand, T. Salonidis, and E. Knightly. Design and experimental evaluation of multi-user beamforming in wireless LANs. In *Mobicom 2010*.
- [6] N. August, H.-J. Lee, M. Vandepas, and R. Parker. A TDC-less ADPLL with 200-to-3200 MHz range and 3 mw power dissipation for mobile SOC clocking in 22nm CMOS. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pages 246–248. IEEE, 2012.
- [7] H. V. Balan, R. Rogalin, A. Michaloliakos, K. Psounis, and G. Caire. Airsync: Enabling distributed multiuser mimo with full spatial multiplexing. 2012.
- [8] C. A. Balanis. *Antenna theory: analysis and design*. John Wiley & Sons, 2012.
- [9] P. Baronti, P. Pillai, V. W. Chook, S. Chessa, A. Gotta, and Y. F. Hu. Wireless sensor networks: A survey on the state of the art and the 802.15. 4 and ZigBee standards. *Computer communications*, 30(7):1655–1695, 2007.
- [10] R. H. Bloks. The IEEE-1394 high speed serial bus. *Philips Journal of Research*, 50(1):209–216, 1996.
- [11] C.-L. Cheng, F.-R. Chang, and K.-Y. Tu. Highly accurate real-time GPS carrier phase-disciplined oscillator. *Instrumentation and Measurement, IEEE Transactions on*, 54(2):819–824, 2005.
- [12] Cisco, “Cisco Aironet 2600 Access Point”. http://www.cisco.com/en/US/prod/collateral/wireless/ps5678/ps12534/data_sheet_c78-709514.pdf.
- [13] S. C. Ergen. ZigBee/IEEE 802.15.4 summary. <http://pages.cs.wisc.edu/~suman/courses/838/papers/zigbee.pdf>, 2004.
- [14] Ettus, Universal Software Radio Peripheral. <http://www.ettus.com>.
- [15] FCC, “FCC online table of frequency allocation, April 16, 2013”.
- [16] Y. Hiraku, I. Hayashi, H. Chung, T. Kuroda, and H. Ishikuro. A 0.5 V 10MHz-to-100MHz 0.47 μ W/MHz Power Scalable AD-PLL in 40nm CMOS.
- [17] Jackson Labs, Fury GPSDO. <http://jackson-labs.com/>.
- [18] Laird Technologies, “Laird VHF antenna antenna”. <https://www.tessco.com/products/displayProductInfo.do?sku=59459>.
- [19] S. H. Lim, Y.-H. Kim, A. El Gamal, and S.-Y. Chung. Noisy network coding. *IEEE Transactions on Information Theory*, 57(5):3132–3152, 2011.
- [20] I. Maric, N. Liu, and A. Goldsmith. Encoding against an interferer’s codebook. In *46th Annual Allerton Conference on Communication, Control, and Computing, 2008*, pages 523–530. IEEE, 2008.
- [21] Maxim Integrated, “VHF/UHF Low-Noise Amplifiers.”. <http://datasheets.maximintegrated.com/en/ds/MAX2664-MAX2665.pdf>.
- [22] MiniCircuits. <http://www.minicircuits.com>.
- [23] B. Nazer and M. Gastpar. The case for structured random codes in network capacity theorems. *European Transactions on Telecommunications*, 19(4):455–474, 2008.
- [24] NIST. NIST Radio Station WWVB. <http://www.nist.gov/pml/div688/grp40/wwvb.cfm>.
- [25] NIST. Radio Station WWV. <http://www.nist.gov/pml/div688/grp40/wwv.cfm/>.
- [26] F. Quitin, M. M. U. Rahman, R. Mudumbai, and U. Madhow. A scalable architecture for distributed transmit beamforming with commodity radios: Design and proof of concept. *Wireless Communications, IEEE Transactions on*, 12(3):1418–1428, March 2013.
- [27] H. Rahul, H. Hassanieh, and D. Katabi. SourceSync: A Distributed Wireless Architecture for Exploiting Sender Diversity. In *ACM SIGCOMM 2010*, New Delhi, India, August 2010.
- [28] H. Rahul, S. S. Kumar, and D. Katabi. Megamimo: Scaling wireless capacity with user demand. In *SIGCOMM*, 2012.
- [29] Silicon Labs, “Jitter Attenuater”. <http://www.silabs.com/Support%20Documents/TechnicalDocs/Si5328.pdf>.
- [30] Silicon Labs, “High-Performance, Integrated ZigBee/802.15.4 System-on-Chip.”. <http://www.silabs.com/SupportDocuments/TechnicalDocs/EM35x.pdf>.
- [31] M. Speth et al. Optimum receiver design for wireless broad-band systems using OFDM- I. *Comm, IEEE Trans. on*, 47(11):1668–1677, Nov. 1999.
- [32] Texas Instruments, “True System-on-Chip Solution for 2.4-GHz IEEE 802.15.4 and ZigBee Applications.”. <http://www.ti.com/lit/ds/symlink/cc2530.pdf>.
- [33] J. Wang, H. Hassanieh, D. Katabi, and P. Indyk. Efficient and reliable low-power backscatter networks. *ACM SIGCOMM Computer Communication Review*, 42(4):61–72, 2012.

