DigiScatter: Efficiently Prototyping Large-Scale OFDMA Backscatter Networks

Fengyuan Zhu
Shanghai Jiao Tong University
jsqdzhufengyuan@sjtu.edu.cn

Yuda Feng
Shanghai Jiao Tong University
fydsjtu15@sjtu.edu.cn

Qianru Li
Shanghai Jiao Tong University
liqianruu@sjtu.edu.cn

Xiaohua Tian
Shanghai Jiao Tong University
xtian@sjtu.edu.cn

Xinbing Wang
Shanghai Jiao Tong University
xwang8@sjtu.edu.cn

ABSTRACT

Recently proposed OFDMA backscatter could improve both concurrency and spectrum allocation flexibility for backscatter systems based on OFDM. However, we find that it is remarkably inefficient for the existing design to scale up in prototyping: it requires one-by-one offline computation to obtain tags' operating parameters, in order to ensure orthogonality among subcarriers in the system; moreover, the tag hardware has to be dedicatedly modified offline before being assigned multiple subcarriers. The inefficiency is caused by the current analog frequency synthesis design for the tag. This paper proposes DigiScatter, an OFDMA backscatter system realizing digital frequency synthesis, which provides an efficient prototyping approach for large-scale OFDMA backscatter networks. In DigiScatter, we for the first time integrate IDFT into the tag design; such a simple but effective improvement enables the system to support high concurrency and flexible spectrum resource allocation through pure software configurations in an online manner. We build a prototype and conduct comprehensive experiments to validate our design. DigiScatter physically realizes 100 and 300 concurrent OFDMA backscatter transmissions in 2.4GHz and 900MHz respectively, and provides frequency synthesis capability for supporting 1019 concurrent transmissions.

CCS CONCEPTS

• **Networks → Home networks**; **Cyber-physical networks**; **Sensor networks**.

KEYWORDS

OFDMA, Backscatter Communication

ACM Reference Format:

Fengyuan Zhu, Yuda Feng, Qianru Li, Xiaohua Tian, and Xinbing Wang. 2020. DigiScatter: Efficiently Prototyping Large-Scale OFDMA Backscatter Networks. In *The 18th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '20), June 15–19, 2020, Toronto, ON, Canada.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3386901.3388914

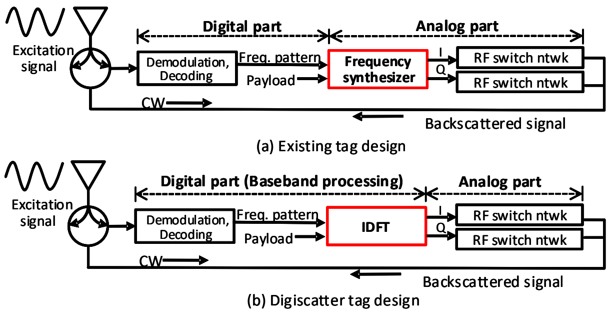


Figure 1: DigiScatter overview.

1 INTRODUCTION

Backscatter communication can provide μW power level backhaul to Internet-of-Things (IoT) devices, which usually have very limited energy budgets [2, 3, 5, 6, 9–13]. With proliferation of IoT widgets featured by short bursts of data into human activities, scalability of the backhaul becomes critical, where the crux is to support as many concurrent backscatter transmissions as possible. The groundbreaking work of *NetScatter* presents the *distributed chirp spread spectrum* (DCSS) mechanism [3], which allows 256 concurrent backscatter transmissions. With DCSS, each of the concurrent backscattering devices is assigned a unique cyclic shift of the chirp signal, and the device then conveys bits with OOK (ON-OFF Keying) modulation.

While novel and promising, the DCSS design is based on chirp spread spectrum modulation technique, which is known as the core of LoRa [2]; in contrast, most of the modern wireless systems including 802.11 a/g/n/ac, 4G and 5G adopt *orthogonal frequency division multiplexing* (OFDM) as the modulation scheme [32], based on which *orthogonal frequency division multiple access* (OFDMA) can be realized for concurrency enhancement. Zhao *et al.* for the first time enables OFDMA in the Wi-Fi backscatter network [5], which realizes 48 concurrent backscatter transmissions leveraging the OFDM framework of 802.11g. With OFDMA backscatter design, tags perform *frequency shifting* to the excitation signal, in order to generate backscattered signals filling bands of those OFDM subcarriers. This is equivalent to assigning each tag a subcarrier, thus simultaneous backscatter transmissions incur no interference with each other due to orthogonality among those subcarriers.

Concurrency of the existing OFDMA backscatter design theoretically depends on the number of subcarriers the OFDM framework defines: Imagine that the next-generation Wi-Fi 802.11ax could contain 996 subcarriers in 2.4GHz and 5GHz band respectively

[23–26], then it should be possible to realize thousands of concurrent transmissions. Moreover, the OFDMA backscatter can provide heterogeneous connectivities to tags: it is possible to assign 4 subcarriers to a particular tag, which increases the tag's data rate by 4 times [5].

However, we find that it is remarkably inefficient for the proof-of-concept OFDMA backscatter network to scale up in prototyping: it requires one-by-one offline computation to obtain tags' operating parameters, in order to ensure orthogonality among subcarriers in the system; moreover, the tag hardware has to be dedicatedly modified offline before being assigned multiple subcarriers. The inefficiency is caused by the current analog frequency synthesis design for the tag. In particular, the tag design is shown in Fig. 1(a), where the excitation signal is split into two ways. Along the upper path, the signal is demodulated and decoded to obtain the control information, which notifies the tag the frequency shift supposed to realize. The *frequency synthesizer* generates an analog single tone with selectable phases and frequency Δf, which is for not only payload information modulation but also making the tag work in desired subcarrier frequency. Then the I/Q output of the modulated signal controls the RF switch network to perform mixing with the continuous wave (CW) coming along the lower path.

The frequency synthesizer is realized by a voltage-controlled oscillator (VCO) based phase locked loop (PLL) using FPGA [5]. In order to ensure strict frequency orthogonality among backscattered signals, it is recommended that operating parameters of the PLL on each tag are carefully computed offline with dedicated software such as clock wizard [37]. It is time-consuming to obtain those parameters when there are hundreds to thousands of tags, and sending those parameters to tags through the system downlink incurs non-trivial messaging overhead, considering that the downlink of backscatter systems is usually with limited capacity under simple OOK modulation scheme. Moreover, the tag has to be modified by installing an 8 way splitter/combiner before it can work with 4 subcarriers for higher data rate [5]; it is impossible to allocate multiple subcarriers to uniformly designed tags in the runtime. Such issues hinder OFDMA backscatter prototype from efficiently scaling up, and deteriorate the desired flexibility.

In this paper, we present **DigiScatter**, an OFDMA backscatter system realizing digital frequency synthesis, which provides an efficient prototyping approach for large-scale OFDMA backscatter networks. In DigiScatter, we for the first time integrate *inverse discrete Fourier transform* (IDFT) into the tag design; such a simple but effective improvement enables the system to support high concurrency and flexible spectrum resource allocation through pure software configurations in an online manner. Our technical contributions are as follows.

First, we show how to integrate IDFT into the tag design, which realizes all-digital baseband processing as shown in Fig. 1(b): the I/Q output directly controls the RF switch network for mixing with the CW without the analog frequency synthesizer (§3.2), in contrast to the classic tag design adopted by a number of existing backscatter systems as shown in Fig. 1(a) [5, 9–11, 13].

Second, we present new protocols for the OFDMA backscatter system under the IDFT based frequency synthesizer, which allows the system to flexibly allocate multiple subcarriers to any uniformly designed tag in an online manner without hardware modification. Moreover, the system is able to adjust the concurrency and data rate through flexibly configuring IDFT size in the runtime, which yields efficient spectrum resource utilization (§3.2).

Third, we construct a prototype for DigiScatter, which physically realizes 100 and 300 concurrent OFDMA backscatter transmissions in 2.4GHz and 900MHz respectively, and provides frequency synthesis capability for supporting 1019 concurrent transmissions (§5). We conduct comprehensive experiments with the prototype to evaluate our proposed design.

2 PRELIMINARIES

2.1 OFDMA Backscatter Design

OFDMA backscatter [5] is based on OFDM, which contains multiple orthogonal subcarriers in the given spectrum. OFDMA backscatter system assigns each tag a subcarrier through frequency shifting: each tag shifts the backscattered signal's frequency to an OFDM subcarrier's frequency and convey the local information using phase modulation. In particular, the excitation signal contains control information instructing the amount of spectrum spacing (Δf) each tag should carry out for frequency shifting, which guarantees that the backscattered signals are orthogonal to each other in the frequency domain, thus concurrent backscatter transmissions can be supported. The prototype implementation as presented in [5] is based on 802.11g's OFDM framework containing only 48 data subcarriers (the other 12 null subcarriers and 4 pilot subcarriers for special purposes); therefore, 48 concurrent backscatter transmissions can be supported.

Generally, given the spectrum bandwidth BW, the number of subcarriers can be supported in an OFDMA system is determined by the design of the spectrum spacing between neighboring subcarriers (Δf). Theoretically, Δf can be arbitrarily small, making infinite subcarriers available, which in turn making infinite concurrent backscatter transmissions possible. However, smaller spacing requires higher frequency resolution of hardware in practical system implementation. For OFDMA backscatter, the tag must be able to distinguish neighboring subcarriers under the spectrum spacing of Δf, which is actually determined by the stability and accuracy of the frequency-shifting clock in the tag.

The typical crystal oscillator's frequency error tolerance can be up to 100ppm. If we directly use this clock to generate the shift frequency, then the maximum frequency error can be $BW \cdot \frac{100}{1\times10^6}$. If the resulted frequency error is equal to or greater than Δf, then it is impossible for the tag to distinguish a subcarrier and the ones adjacent to it. Consequently, we must guarantee that the maximum frequency error tolerance is no greater than $\Delta f = \frac{BW}{N}$, where N is the concurrency. That is

$$BW \cdot \frac{100}{1 \times 10^6} \le BW \cdot \frac{1}{N} \tag{1}$$

thus

$$N \le 1 \times 10^4. \tag{2}$$

This means that the maximum concurrency of the OFDMA backscatter system can be up to 1×10^4. The potential of OFDMA backscatter design to support concurrency is huge.

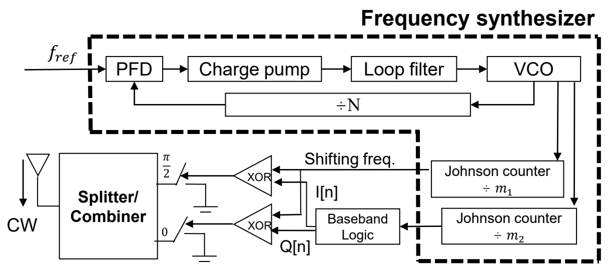


Figure 2: Tag design with analog frequency synthesizer.

2.2 Issues for Network Scaling

■ **Laborious offline configuration process.** The tag's circuit design of OFDMA backscatter is shown in Fig.2, where the part contained in the dashed frame is the frequency synthesizer used for generating the shifting frequency. In fact, a number of existing backscatter systems [5, 9–11, 13] adopt the similar design. The frequency synthesizer is factually a VCO based PLL, and the VCO itself is a regular analog component in FPGA. It requires a precise external clock as a reference clock, which is usually a crystal oscillator; the output of VCO is multiple of the reference clock, which goes through two Johnson counters to generate the shifting frequency and the baseband frequency respectively. The baseband logic outputs the local information in the form of square wave, and the two RF switches act as mixers, which "moves" the information to the carrier with desired frequency.

In order to generate square waves with appropriate shifting frequency and baseband frequency, values of parameters **N**, m_1 and m_2 need to be carefully assigned after f_{ref} is given. For example, passive Wi-Fi [9] uses $f_{ref} = 12.375kHz$ to generate $49.5MHz$ VCO frequency, which is 4000× the reference frequency. The VCO frequency is then divided by 4 and 4.5 to generate the shifting frequency and baseband frequency, which are $12.375MHz$ and $11MHz$, respectively.

The frequency generated by a PLL is precise if the mathematical relationship among f_{ref}, f_{vco} and f_{out} is correctly modeled. The model is then written into divider registers to take effect. A number of factors have to be taken into account to construct the relationship model, such as the divider range, VCO operating range and PFD constraints. More constraints can be found in FPGA datasheets [57–60]. For example, most VCOs on FPGAs can only work in an interval of $[600, 1200]MHz$ for Xilinx 7-series FPGAs, and it is a typical rule for the PLL to maximize VCO frequency while using smallest dividers when there exist multiple solutions to generate the target frequency. Due to such implicity between target frequency and divider settings, the frequency configuration process is usually manually computed via IDE provided by the FPGA manufacturer. Consider that in 802.11ax [23], the number of OFDM subcarriers grows to 256 and the subcarrier spacing becomes $78.125kHz$. All 256 tags need to be manually configured and programmed for its PLL configuration. To prototype a backscatter network in this scenario is time-consuming.

It is worth mentioning that an alternative could be using the COTS programmable oscillator to directly generate shifting frequency, which however still can not avoid configuring the divider settings in the prototyping process. We investigate the programmable oscillators from Ti, Silicon Labs and Linear Technology[43–45], which are suitable in frequency range and frequency step for the purpose of scaling up the OFDMA backscatter prototype; however, we find that they are called 'programmable' because their dividers can be digitally configured, similar to PLL modules on FPGAs. As a result, the same problem will still exsit even if we use COTS programmable oscillators.

■ **Inflexibility in supporting heterogeneous connectivities.** An advantageous capability of the OFDMA backscatter design is to provide heterogeneous connectivities to tags [5]. In particular, some tags can be allocated with more than one subcarrier, thus supporting faster uplink. Physically, all the tags can be regarded as mainly consisting of a splitter/combiner and a single side band (SSB) module as shown in [5], where the SSB module manages to yield two way backscatter signals with $\frac{\pi}{2}$ frequency offset. Consequently, if we install an 8-way splitter/combiner and 4 SSB modules in the tag, then the tag can work with 4 subcarriers thus the data rate can increase by 4 times. Each SSB module will also need to implement an independent analog synthesizer for the specific subcarrier generation. Such design requires the tag's hardware to be modified offline, and the system is unable to adjust the subcarriers allocation in the runtime, which deteriorates the flexibility of spectrum resource allocation in the OFDMA backscatter system.

3 CORE DESIGN OF DIGISCATTER

Based on the analysis above, we can see that the analog frequency synthesizer is the main reason that hinders prototyping large-scale OFDMA backscatter networks. The natural idea is to design a digital synthesizer to realize the frequency shifting operation. This section present the core design of DigiScatter. We first present the basic idea of digital synthesis for OFDMA backscatter, and then show how to implement the design in practice. We are to show that our simple but effective improvement can resolve the issues for prototyping large-scale OFDMA backscatter networks, and can help improve spectrum efficiency of the system.

3.1 Basic Idea

The frequency shifting operation essentially is the process of generating a time-domain signal that fills a desired frequency band. This process in the regular OFDMA system is factually accomplished by the digital IFFT module of the transmitter. IFFT module has two important parameters: 1) size or the number of bins denoted by N; 2) output sampling frequency f_s. The output sampling frequency determines baseband bandwidth; the number of bins is factually the number of pieces the baseband is sliced, or the number of subcarriers. Each input bin is a frequency-domain entry to a subcarrier and each output bin is a sampling point of the time-domain signal to be transmitted. We can see that there is no need to consider divider settings in IFFT in contrast to the current design as mentioned.

Along this vein, we propose the tag design with IFFT as shown in Fig. 3. The first step is to put a symbol from the input bit stream into an input bin of the IFFT module, in order to assign the symbol a

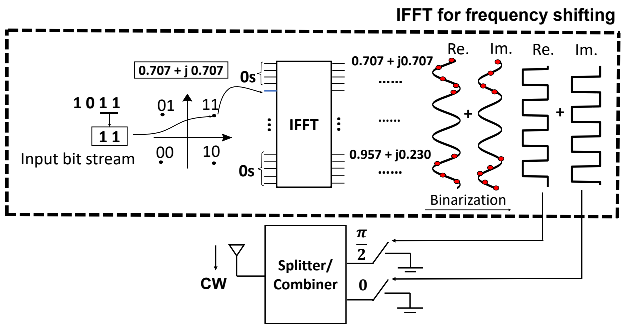
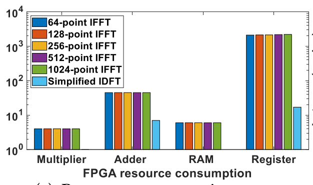


Figure 3: Tag design with IFFT for Frequency shifting.

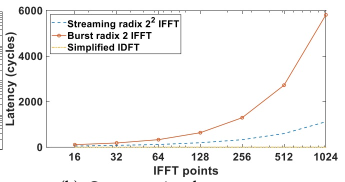
subcarrier with desired frequency. After performing IFFT, both the real and the imaginary part of the output sequence are thresholded to directly control those RF switches for mixing. The key observation here is that the process directly generates the symbol-level discrete sequence just like in the regular OFDM transmitter. Supporting heterogeneous connectivities requires generating multiple shifting frequencies simultaneously, for which the symbols could be first interleaved and then fed to corresponding input bins. After performing IFFT, the remaining process is the same.

Note that the high-resolution DAC is needed in regular active OFDM transmitters to convert IFFT outputs into the analog domain with acceptable distortion. After that, the output will be further mixed by local carrier and then converted to RF domain. For the backscatter device, even if the DAC is realized, there is no way for the output to be directly modulated on the carrier that is actually provided by the excitation signal transmitter, which is too weak to drive a MOSFET based mixer. Our key insight here is that we can use local IFFT digital outputs to drive RF switches to mix the carrier signal for the tag. To make the IFFT outputs to be accepted by RF switches, we need to first binarize the signal in both I and Q path. Such binarization would distort the IFFT output waveform, which however is acceptable for backscatter devices expected to utilize limited number of subcarriers simultaneously.

Compared with the design as shown in Fig. 2, the new design as shown in Fig. 3 can dramatically improve efficiency and flexibility for large-scale prototyping. Consider that the bandwidth of the frequency spectrum BW is determined by the sampling frequency, the subcarrier spacing is $\frac{BW}{N}$. If we choose an arbitrarily great value of N, there will be an arbitrarily large number of subcarriers, which is convenient for the prototype to scale up for higher concurrency. In particular, we can generate a time-domain sequence for frequency shifting in any spectrum resolution if we choose an appropriate N, feed the symbol into input bins and fix f_s. For example, to generate 256 concurrency with $20MHz$ band, we only have to use a sampling clock of $20MHz$, and set $N = 256$. The tag working in the x^{th} subcarrier only needs to feed the information to the input bin x and sets other input bins to be zeros. Such a method can generate multiple subcarriers simultaneously with the similar workflow, because IFFT naturally supports parallel modulation on different subcarriers. It is flexible to generate 4 subcarriers without any need for hardware modification compared with the design in [5]. No matter how high concurrency needs to be supported, IFFT



(a) Resource consumption.



(b) Computation latency.

Figure 4: FPGA resource consumption and latency: Regular IFFT vs streamlined IDFT.

solution only needs the tag to generate the frequency equal to the baseband bandwidth (e.g., $20MHz$ in 802.11g), which is easy for most of FPGAs.

3.2 IDFT based Design

To implement the design as shown in Fig. 3 with COTS IFFT module in the FPGA is feasible but may not be optimal for the following two reasons:

- **Resource consumption**. The IFFT module consumes FPGA resources like adders, RAMs, multipliers and registers. We take the open-access burst radix 2 IFFT design provided by Simulink [46] as an example, where the resource consumption is illustrated in Fig.4(a). We can see that over two thousands bit registers are used, and we still leaves the resource consumption of interface logic, output buffer and timing control logic unconsidered.
- **Latency**. IFFT operations are sequential and incur latency. To perform IFFT, a frequency-domain input vector should be first loaded, which constitutes a 'frame'. It then takes several clock cycles to finish the butterfly operation for the 'frame'. We note that an IFFT 'frame' only contains one symbol. To transmit all symbols in a frame, we need to obtain IFFT 'frames' one after another. The latency is dependent on the IFFT architecture and the number of IFFT points. We test the latency of two open-access IFFT designs from Simulink[46] to illustrate the latency of a 'frame' in Fig.4(b).

We propose an IDFT based design to realize the basic idea as shown in Fig. 3. It is well known that IFFT is a fast algorithm for IDFT, which is to transform a frequency-domain sequence $X[k]$ into the time-domain counterpart $x[n]$:

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{-kn}, \tag{3}$$

where W_N equals $e^{-j2\pi/N}$. IFFT is designed for general purposes with many inputs and high-resolution fixed-point outputs, which however is far more than what is required for performing frequency shifting in backscatter. We have the following observations for IFFT based frequency shifting as shown in Fig. 3.

- **Sparse input.** Each tag is normally allocated one bin (subcarrier), making all other elements in the input vector zeros except the allocated bin.
- **One-bit output** The tag modulates CW through impedance matching and the impedance has only two values: 0 and infinity, which corresponds to the reflection ratio of -1 and 1, respectively. Consequently, the result of IFFT is not necessarily an accurate fixed-point number, and could be an one-bit or logic number.

Consequently, we can rewrite Eq. (3) for the particular frequency shifting scenario. Assume that a tag is assigned a subcarrier corresponding to IFFT bin k_0, then rest of the bins are set to be zeros:

$$x[n] = \frac{1}{N}X[k_0]W_N^{-k_0 n}.$$

Take PSK for example, $X[k_0] = e^{j2\pi p_0}$ with $2\pi p_0$ being the complex angle of a PSK symbol. Using Euler formula to expand the equation, we have

$$x[n] = \frac{1}{N}cos(\frac{2\pi}{N}k_0 n + 2\pi p_0) + j\frac{1}{N}sin(\frac{2\pi}{N}k_0 n + 2\pi p_0),$$

where n ranges from 0 to $N-1$ and k_0 is a constant. It can be seen that the complexity of IDFT in this case is only $O(N)$, in contrast to the computational complexity of IFFT $O(Nlog(N))$. Directly computing IDFT brings reduced computational complexity in the particular case.

We set the threshold to be zero for both the real and imaginary part of $x[n]$ to fit the two states of RF switches. For simplicity, the real part one-bit output of $x[n]$ is denoted by $R[n]$, and the imaginary part output of $x[n]$ is denoted by $I[n]$. We thus obtain

$$R[n] = NOT((\frac{k_0 n}{N} + p_0 \geq 0.25 + m)\&\&(\frac{k_0 n}{N} + p_0 < 0.75 + m)),$$

$$I[n] = (\frac{k_0 n}{N} + p_0 \geq 0 + m)\&\&(\frac{k_0 n}{N} + p_0 < 0.5 + m),$$

where

$$m = [\frac{k_0 n}{N} + p_0].$$

Apparently, using the above IDFT design would save FPGA resource in prototypes. To quantize it, we realize the above simplified IDFT modules with different sizes using FPGA; and the corresponding resource consumption and latency cost are shown in Fig.4. It can be seen that IDFT significantly saves all aspects of FPGA resources compared to burst radix 2 IFFT. The simplified IDFT consumes 1 multiplier, 7 adders, 17 registers and no RAM. As the size increases, the resource consumption of both IDFT and IFFT does not change much. For IFFT, this characteristic is at the cost of longer latency, while IDFT always has no latency.

3.3 Dynamic Spectrum Allocation

Our simple but effective improvement described above can resolve the issues for prototyping large-scale OFDMA backscatter networks as mentioned in Section 2.2. We here show that the new design can realize dynamic spectrum allocation for the system in the runtime.

Practical deployment of the large scale backscatter network may encounter the scenario that the number of tags online is dynamic. In particular, the tag just online may want to connect to a network, but the network has reached the capacity of its own; a number of tags may leave a network, leaving most of the spectrum resource idling. In order to support the use case and fully utilize the spectrum resource, we must be able to adjust the spectrum spacing in the runtime to rearrange subcarriers, which essentially requires changing the IDFT size in the runtime.

The size of IDFT is $N = 2^M$, and division by N can be realized through right shifting M bits. The process utilizes the property of fixed-point number therefore can be easily implemented on a FPGA. If the subcarriers are insufficient, we could instruct the IDFT module on the tag to double N by increasing M by 1. If many idling subcarriers appear, we could instruct the tag to decrease M by 1. The subcarrier occupation information is recorded by the excitation signal transmitter, which takes charge of sending control messages according to the network condition. Note that either in increasing M or deceasing M, the tag would not change k_0. Once k_0 exceeds the new value of N, the device would return to unassociation state and join the network again; then it will be assigned with a new valid k_0.

In order to support the functionality, we need to modify the receiver design of the OFDMA backscatter system. A generic FFT module is needed on the receiver to support dynamic-sized IDFT operations on tags. We could set FFT size of the receiver to be N_0, the maximum size designed for the receiver, and always let the receiver to perform N_0-point FFT. For example, an 1024-point FFT module can handle situations with the IDFT sizes ranging from 16 to 1024. Recall Eq. (3), the term W_N^{-kn} distinguishes different subcarriers. Consider a concurrency condition $N_1 = 2^{M_1}$, a bin k_1 is mapped to bin $2^{10-M_1} \cdot k_1$ in 1024 concurrency. We consider W_N^{-kn} under $N = 128$ and $k = 4$, which can be written as $W_N^{-kn} = e^{-j2\pi \cdot 4n/128}$, or $e^{-j2\pi \cdot 32n/1024}$. The latter one is also the expression of *bin* 32 in the 1024. In other words, *bin* 4 in the 128-point IFFT case is equivalent to *bin* 32 in the 1024-point case.

For decoding a frame under the concurrency situation that is lower than 1024, we first perform zero-padding to increase the length of the original symbol to fit the length of 1024-point FFT. We then perfom 1024-point FFT, and down sample the output sequence to fit the concurrency N_0 before we perform channel equalization and direction-directed PLL. Note that the key steps zero-padding and down sampling are for FFT-size matching and maintaining orthogonality. Based on the design, we can use a fixed-size FFT to decode dynamic concurrency situations.

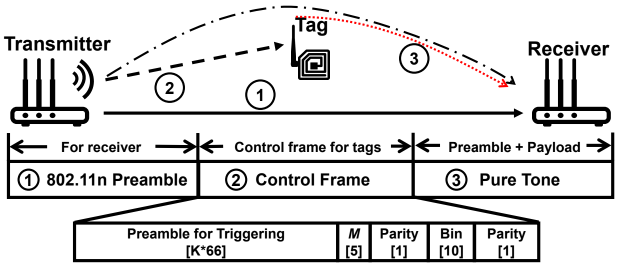


Figure 5: Transmitter frame structure.

4 PROTOCOL DESIGN

We now show how the excitation signal transmitter, tags and the receiver interact with each other in the DigiScatter system, so that those advantages mentioned can be fully exerted. We use Fig. 5 to illustrate how new-coming tags can associate with the system, and use Fig. 6 to show how the associated tag and the newcomer can coexist.

In Fig. 5, tx and rx have the same capability of decoding backscattered signals from tags. Tx continuously broadcast the data frame as shown in the figure, which basically contains three parts:

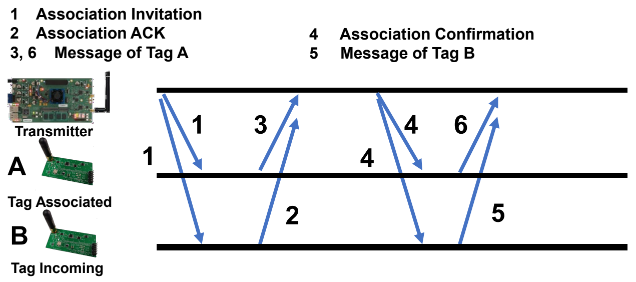


Figure 6: DigiScatter association process: tag B as new-comer and tag A has been associated.

- **802.11n preamble.** This is for tx-rx synchronization in both time and frequency domain.
- **Control frame.** The frame consists of several fields: The Preamble is for triggering the tag, M followed by a *Parity* is to guarantee that the regulated system concurrency configuration is correctly received. *Bin* is for assigning the tag the corresponding subcarrier, and the *Parity* field is with the same purpose as in the case of M field. Preamble consists of a variable number of triggering sequences; and each sequence is embedded with unique clock information for syncrhonization. The length of Preamble is adjusted to fit practical deployment.
- **Pure tone.** Tx starts to send a pure tone with bin 0. The tag starts to convey local bits upon receiving the pure tone. The backscattered signal to be transmitted by the tag consists of a preamble and payload. The tag-to-rx preamble is for channel estimation and modulation mode indication. In particular, the preamble is transformed into the frequency domain at the receiver to estimate the corresponding channel response $H(k)$ for channel equalization; we regulate that the preamble uses BPSK modulation, while the payload can be in either BPSK or QPSK, for which the last symbol of the preamble indicates which modulation is used. The payload contains a cyclic suffix (CS) addressing ICI and ISI, which has the same function as the cyclic prefix (CP) design as in [5]. We mention that static phase offsets will be calibrated during the channel estimation for tag-to-rx preamble; and the dynamic phase offsets mentioned in [5] is addressed by DD-PLL algorithm [36].

Joining process. After slicing the bandwidth, Tx has 2^M subcarriers available. Tx first selects one of the subcarriers and asks if there is a tag wants to join in the system using Association Invitation, which is realized by putting the index of the subcarrier in *Bin* field of the broadcasting frame. The new coming tag receives the frame and joins in the system by conveying an Association ACK using the subcarrier. After tx decodes the request, it confirms that the new coming tag is assigned the subcarrier just broadcast by sending an Association Confirmation consisting of an invalid N (≥ 2048) and the ID of the tag. Tx then will broadcast the next subcarrier to be assigned. This process is shown in tag B's case of Fig. 6.

If there is a tag to be allocated 8 subcarriers tries to join in the network, then the tag needs to execute the regular joining process for 8 times till all its subcarriers are successfully allocated. The association ACK is modified correspondingly so that the receiver can correctly decode the messages using the deinterleaver.

Collision settlement in joining process. If multiple new-coming tags claim the same subcarrier, the tx is unable to decode the ACK. Then Tx will fill a special value (1023) into the *Bin* field and broadcast again. Upon receiving this frame, tags will start the back-off process, so that they will finally get associated. In detail, each tag has a 4-bit counter driven by the sampling clock. These four bits, which are virtually random, will be used as the initial value of a frame-level count down clock once a back-off command is received. In practical deployment scenarios, tags can also be pre-configured the joining order if the system need to associate a large number of tags from the start, which could save much time for networking.

Concurrency configuration process. After associated with the system, tags record the value of M field in the broadcast frame. If the newly broadcasting frame has a new value in M field, this means that the system is adjusting the concurrency configuration. If the concurrency is enhanced, associated tags do not need to rejoin; if the concurrency is reduced, the tags assigned a subcarrier with index greater than 2^M must rejoin.

Triggering synchronization. Large-scale concurrent transmissions require all the tags to be triggered at the same time. Existing OFDMA backscatter design utilizes OOK signal with 11-bit message to trigger tags generating backscattered signals; however, such triggering scheme is unreliable especially when there are a large number of tags. It could occur that only part of the tags are successfully triggered in one round of excitation signal broadcasting; those failed tags have to wait for the next round, thus the overall system latency increases. To resolve the issue, a natural idea is to use coding schemes to improve the reliability of triggering, for example, existing OFDMA backscatter design uses 11-bit barker code [5]; however, simply using longer codes still can not guarantee triggering synchronization among a large number of tags. The low energy budget of the tag's receiver requires simple but effective design to achieve the goal. Our solution is as follows.

The control frame contains up to 8 triggering sequences in order to provide more triggering opportunities, so that all the tags can be triggered at the same time. Each triggering sequence is with an index, which indicates the period of time it is from the coming CW. No matter which sequence works for the tag, the tag can estimate the starting point of the CW and executing the backscatter process at that time. Using 8 sequences provides highest probability to trigger all tags but results in the longest version of the control frame, which in turn incurs highest downlink overhead/latency. We need to appropriately choose the number of triggering sequences in the control frame. Suppose that the system is in the state with observed triggering rate p_0, current number of triggering sequences is K and there are n tags. The latency cost for all backscatter tags C is:

$$C = (1 - p_0) \cdot n \cdot 1/p_0 \cdot T_{frame} + K \cdot T_{trigger} \cdot n;$$

where T_{frame} and $T_{trigger}$ denote the duration of the frame and the single triggering sequence, respectively. The first term refers to the estimated latency of retransmission for the untriggered tags, and the second refers to the latency cost for triggering sequence itself for all tags. In this case, Tx would adapt K by 1 if it can decrease the total cost C or $\Delta C < 0$, equivalently. Here ΔC is given by

$$\Delta C = -\widetilde{\Delta p} \cdot 1/p_0 \cdot T_{frame} + T_{trigger} \cdot n;$$

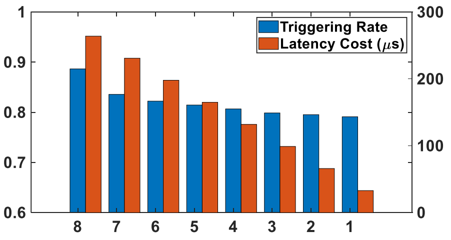


Figure 7: Trade-off of triggering rate.

where $\widetilde{\Delta p}$ is the empirical difference of triggering rate and should be measured through experiments. We present the empirical triggering rate $\widetilde{p}$ under different K configurations in Fig.7. We note that the triggering rate can vary from time to time in a day dependent on channel condition. In our experiments, we can oberseve 100% triggering rate for consecutive 1000 frames when the channel is good, while sometimes the triggering rate can be as low as 72%.

Coexistence Scenario. In Fig. 6, A and B are associated and new-coming tags respectively. Tx broadcasts the frame as shown in Fig. 5 and received by both of the tags. Tag A has associated with the system thus send the local message back, but Tag B is a new comer thus it has to accomplish the joining process as described above. Note that the association process related messages will also be received by tags that have been in the system, but will cause no influence as long as the concurrency configuration is not changed. Note that all the semantics can be delivered with the syntax of the frame as shown in In Fig. 6. Such transmitter-initiated association process saves one round of handshake in the joining process, in contrast to the tag-initiated process presented in NetScatter.

5 IMPLEMENTATION AND IC DESIGN

■ **Prototype construction.** The hardware and deployment of the DigiScatter testbed are shown in Fig. 8. We use WARP V3 board [49] to realize the transmitter and receiver in 2.4GHz. The transmission power is measured to be $15dBm$. We attach the same $2dBi$ antenna to the RF front when conducting experiments. We implement the tag following the new design described in previous sections of the paper. The RF part of the tag consists of an envelope detector and a reflective RF switch for backscattering as in [5], which is attached with a $2dBi$ antenna working in $2.4GHz$. The IDFT algorithm and downlink design are all written in Verilog and deployed on the CMOD S7 [55] development board, which is based on Spartan 7 FPGA. The RF part of the tag and CMOD S7 is connected through pins on the S7 board. RF part passes the result of envelope detection to S7, which then outputs in phase logic signal to the RF part to realize backscatter modulation.

■ **IC design.** We simulate IC power consumption for the DigiScatter tag design. A typical IC design for the backscatter system contains three main components: digital baseband, backscatter modulator and frequency synthesizer, which are the main sources of power consumption [2, 5–7, 9–11, 13, 30]. The backscatter modulator refers to any assistant control logic that helps realize backscatter modulation, including but not limited to single-side band (SSB) logic and data multiplexer. However, only the first two digital parts can be functionally verified in IC design; frequency synthesizer is a functional verification gap between prototyping and IC implementation. Some of the previous works [5, 10, 11] adopt models of the IC-implemented frequency synthesizer [12, 22] to estimate the corresponding power consumption, and some others did not provide detailed functional performance in IC design [2, 9, 13].

We resolves this issue by integrating the dedicated synthesizer for frequency shifting and backscatter modulator into a single IDFT module which can be described using Verilog/VHDL in both FPGA and IC design. We note that the output signal of IDFT makes a direct contact with the RF switches.Thus we eliminate the uncertainty in IC implementation. Our IC design uses the Verilog code of the baseband verified in FPGA, the Synopsis Design Compiler and IC Complier with 40 nm LP process (voltage = 1.0 V). To give a reasonable power estimation, VCD files containing signal behaviors are used. The baseband mainly contains link-layer and physical layer part. For the link layer including downlink decoding and state configurations, the IC power consumption is $36.8\mu W$. And for the physical layer (channel coding and IDFT) directly controlling RF switches, performing frequency shifting and modulation (which is a natural SSB output), the IC power consumption under $20MHz$ baseband can be as low as $14.7\mu W$. The analog part refers to the RF front, including power detector and RF switches, where the power consumption is less than 3 μW. In total, power consumption of a backscatter device under single subcarrier configuration is around $54.5\mu W$.

■ **Advantage of all-digital baseband design in prototyping.** Comparing the analog frequency synthesis design with our proposed all-digital baseband design as shown in Fig. 1, the latter has the advantage: it makes prototyping and IC simulation more consistent.

The prototyping process adopting the PLL based analog frequency synthesizer design incurs uncertainty in IC simulation for the tag's processing logic. The frequency shift produced by the FPGA is normally stable and accurate; however, since the PLL has the analog VCO, it can hardly be described using hardware description languages (HDLs) in IC simulation. Consequently, the simulation usually adopts a modeling based approach to estimate the power consumption. That is, the PLL is modeled as a ultra-low power synthesizer that can generate the clock required for frequency shifting [2, 5, 10, 11]. The overall power consumption is the sum of the power consumption of the modeled frequency synthesizer and other IC-realized submodules such as the downlink decoding and baseband processing [2, 3, 5, 9–12]. The hidden uncertainty in this process is: Although the ultra-low power synthesizer model can generate the desired frequency, the corresponding reliability and accuracy are lower than the FPGA PLL. If the IC is indeed manufactured following the simulation scheme [5, 12], the generated frequency shifts might jitter, which could jeopardize the frequency orthogonality thus making the backscatter system unpredictable in practice.

The all-digital processing logic totally can be described using the HDL with no need for modeling, which makes prototyping and IC simulation more consistent.

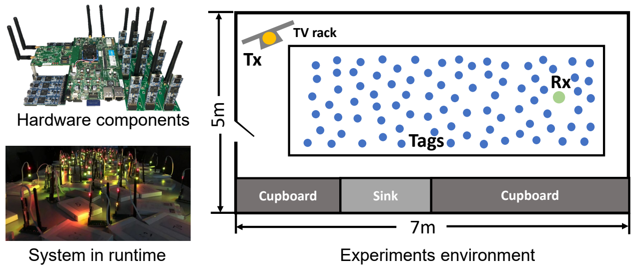
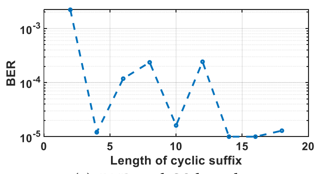
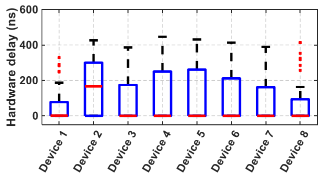


Figure 8: Testbed and experiments environment.



(a) BER and CS length.



(b) Hardware delay.

Figure 9: Timing mismatch.

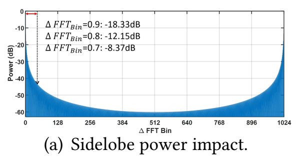
6 EVALUATION

6.1 Timing mismatch

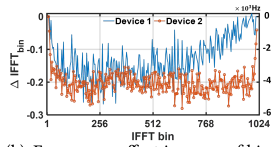
We first examine if the CS design can neutralize the timing mismatch, which is incurred by the hardware response delay and signal propagation delay. Serious time mismatch could incur ISI and ICI as in regular OFDMA system. We conduct the following two experiments to evaluate impact of timing mismatch.

■ **Overall timing mismatch.** We choose a series of CS lengths and measure the BER under those CS configurations. Changing CS lengths will simultaneously modify symbol rates, and hence the receiver algorithm correspondingly adapts in this test. The tx-to-tag and tag-to-rx distance are set to be both $5m$, close to the maximum distance DigiScatter can support. Note that each point of CS length can be translated into timing tolerance of $50ns$. Results are shown in Fig.9(a). It can be seen that the length of CS becomes critical to system performance when CS length is less than 4 sampling points; and the BER becomes stable after then. Our choice of CS length (16 samples) corresponds to the minimum BER and can provide sufficient tolerance for time delay.

■ **Hardware delay.** We randomly select a number of DigiScatter tags in the experiment and measure the response time after being triggered by the control frame from the transmitter. We connect two channels of oscilloscope to the envelope detector output and FPGA logic signal output respectively, and measure the time difference between the end of control frame and the start of backscatter modulation by comparing the waveforms of two oscilloscope channels. The oscilloscope [48] has a sampling frequency of $5GSa/s$. We conduct the same experiments for the randomly chosen devices to reveal the variance. The hardware delay results are depicted in Fig.9(b). We can see that the hardware delay is within $500ns$ in thousands times of tests, and the red median lines show that the hardware delay can be as low as zero, within the time resolution of oscilloscope $0.2ns$.



(a) Sidelobe power impact.



(b) Frequency offset in num. of bins.

Figure 10: Frequency mismatch.

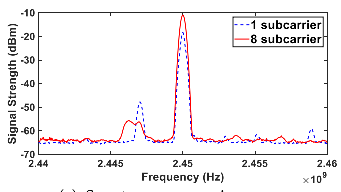
6.2 Frequency Mismatch

The frequency mismatch includes SFO and CFO. Although CFO can be coarsely removed at the beginning of receiver algorithm, residual CFO is still the major factor impacting the system performance, which could be higher than SFO in 1-2 order of magnitude. DigiScatter tags work in fixed frequencies after link-layer configuration, thus timing mismatch has no impact on frequency mismatch. We note that in NetScatter, time delay would incur frequency mismatch because chirp signal's frequency is time-varying, which makes is more difficult to synchronize in time domain. In our experiment, we test frequency offsets caused by SFO under 1024 concurrency. We choose 2 different devices and let them traverse 1024 IFFT bins. In their communication progress, the frequency offset is estimated by the receiver DSP algorithm. To be specific, we use DD-PLL (decision-directed PLL) algorithm [36] to record the phase error and further obtain their frequency offset. DD-PLL is a classic digital-domain carrier recovery algorithm which alternatively estimates the phase deviation and makes demodulation decisions. We record this phase-correction process and extract the phase deviations estimated by DD-PLL, which is further translated to the final frequency offset. The frequency offset can be counted by the number of IFFT bins.

We present these results in Fig.10(b). It shows that the frequency deviation can vary between $10Hz$ and $10kHz$ over the spectrum, which can be totally handled by the DD-PLL. Besides, the minimum frequency spacing between two devices in bin level is 0.75 bin when they work in adjacent subcarriers. The power influence corresponding to the frequency deviation is presented in Fig.10(a). We choose an ideal OFDMA symbol and perform zero-padding to achieve sub-bin spectrum impact similar to [3]. We pad the original symbol to 100×2^{10} points to obtain 0.01 bin resolution. We can see that 0.75 bin deviation's corresponding power influence is $-10.05dB$ according to the result in Fig.10(a). Frequency mismatch can be well handled using DD-PLL with a feedback factor of 0.5.

6.3 Harmonic Interference

DigiScatter adopts 1-bit quantization, that is, two impedance states, which could incur 3rd and 5th harmonics like existing backscatter systems [5, 6, 9–11]. In this experiment, we first assign two devices bins k_0 and $3k_0$ respectively, which makes the second device subject to impact of the first device's 3rd harmonic. We then measure the BER of the $3k_0$-bin device and record results when both devices are under decoding. This result could reflect the 3rd harmonic impact. Similarly, we then assign the second device bin $5k_0$ and $N - k_0$ to measure the 5th harmonic. The baseline are BERs when there is no harmonic interference for the tag (e.g. only one tag operates in the channel). From Fig.12, we can see that the 3rd harmonic has a more serious impact compared to 5th harmonic due to higher power level,


(a) Spectrum comparison.

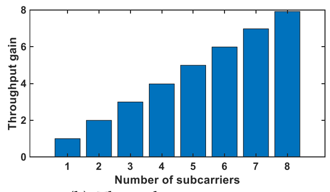

(b) Throughput gain.

Figure 11: Heterogeneous access.

but the latter results will show that all those interferences still can be harnessed.

6.4 Heterogeneous Connectivities

Assigning multiple subcarriers to the tag could possibly incur the following two kinds of negative influences:

■ **Power spreading.** The backscattered signal power is spread to all subcarriers assigned to the tag, which results in lower transmission power for each subcarrier.

■ **Power leakage.** The tag could leak power to impact subcarriers that are not assigned to it, due to the distortion effect brought by quantization (binarization).

To measure spectrum of the multi-subcarrier tag, we first modify the tag logic to send data continuously bypassing triggering scheme and link layer logic, and then put the tag between an analog signal generator (Keysight N5171B) and a spectrum analyzer (Keysight N9322C) which are 1 meter away. The signal generator sends a 2.45GHz pure tone in 20dBm; and the spectrum analyzer has a 20dB input attenuation. Since the downlink is disabled, we configure subcarrier settings via digital pins. We find that if the number of subcarriers assigned to a single tag is no greater than 1/16 total subcarriers, the impacts mentioned above can be neglected as shown in Fig.11(a), which illustrates the spectrum of a single subcarrier in a tag, when the tag is assigned one and 8 subcarriers respectively.

■ **Throughput gain.** The main benefit for the multi-subcarrier tag is the improvement of throughput compared with the single-subcarrier tag. This is verified by the throughput gain ratio of a multi-subcarrier tag under different subcarrier assignment configurations over the single-subcarrier case as shown in Fig.11(b). This performance is obtained under $N \geq 128$.

6.5 Worst-Case LOS and NLOS Range

We set the tx-tag distance d_1 and tag-rx distance d_2 the same and increase both of them simultaneously to find the maximum communication range DigiScatter can support. Such a deployment is the worst-case setting according to the property of the backscatter signal propagation as pointed in [9]. We fix the distance between the transmitter and receiver. In the NLOS scenario, the tag is put inside a cardboard box. The x-axis in Fig.13 shows the value of d_1 (note that $d_1 = d_2$). In the worst case of $d_1 = d_2$, DigiScatter can work in an area with a diameter of 7m with d_1 (or d_2) being at least 5m under different concurrency configurations. Normally, higher concurrency requires higher accuracy for the system operation, thus the BER is higher in the same range setting. The NLOS performance is similar to LOS performance with $N = 256$ in terms of BER but the range is shorter, because the weak signal could hardly trigger the tag in longer distance.

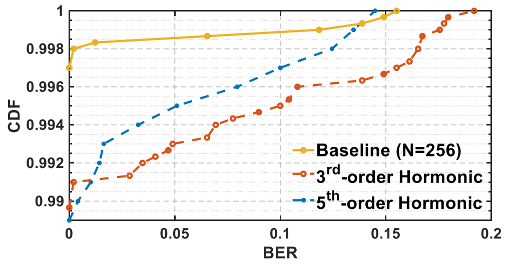
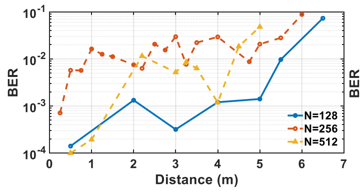


Figure 12: Harmonic interference.


(a) LOS.

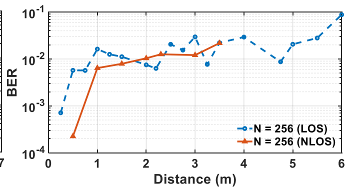
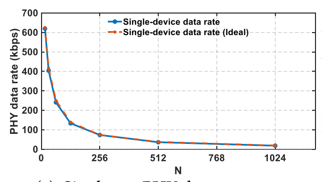

(b) NLOS.

Figure 13: Communication range.


(a) Single tag PHY data rate.

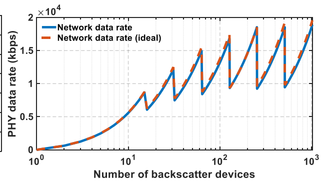

(b) Network PHY data rate.

Figure 14: Data rates.

6.6 Network Performance

We now start to evaluate the DigiScatter system's overall capability for data delivery in the following aspects:

■ **Physical-layer and Link-layer data rate.** The formmer is the raw data rate achieved in the payload part, and the latter measures the data rate of useful payload after considering overheads including preamble, control frame, tag's preamble and channel coding. We evaluate these two data rates for both the single device and the whole network under different concurrency configurations. In particular, we examine situations with concurrency 32, 64, 128, 256, 512 and 1024, respectively, where we adopt BPSK modulation and $f_s = 20MHz$. We use 100 DigiScatter tags for the experiments. For the situation that concurrency is greater than 100, we divide all the subcarriers into groups with each containing up to 100 subcarriers. Such subcarriers are assigned to those 100 tags in several rounds, so that all the bins are traversed. We conduct the experiments in a meeting room as shown in Fig. 8, with all the tags randomly placed on the table. Wi-Fi signals from multiple pre-installed APs can be detected.

The results in Fig.14(a) shows that the PHY data rate of a single tag decreases as the concurrency N increases, because the spectrum resource, i.e. the bandwidth of each tag halves when paramter N doubles. Similarly, in Fig.14(b) the curves are in a jagged shape because the network doubles the number of subcarriers by squeezing current users' spectrum when there are not enough subcarriers. This operation is realized by broadcasting a new parameter M in

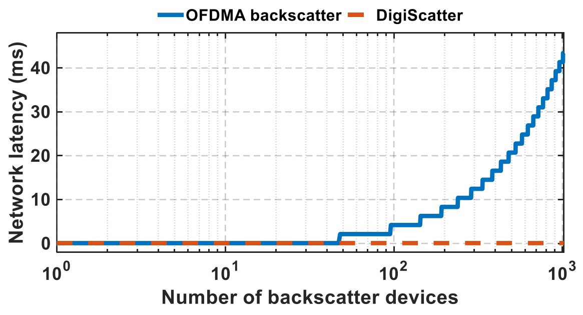


Figure 15: Network latency.

the downlink which doubles parameter N of all the backscatter devices in the network. A drawback of this method is that spectrum utilization would decrease to around 50% everytime the spectrum saturates. However, this is a tradeoff between power efficiency and performance. We note that for the simplicity of IDFT calculation in Section 3.2, we set $N = 2^M$. If we want to realize a smooth data rate, we should be capable of changing parameter N arbitrarily (e.g. 17,18,19...), thus we not only need an extra divider in IDFT module, but also have to increase the overhead brought by downlink control frame since N can be no more represented by M.

■ **Latency.** In Fig.15, we present that DigiScatter provide a significantly lower network latency ($80\mu s$) compared to OFDMA backscatter when there are more than 48 tags. Even when the number of devices is smaller than 48, their latency is at the same level ($80\mu s$ vs. $72\mu s$). Their latency differences lie in three aspects: a) DigiScatter adopts 802.11n preamble, which is shorter than 802.11g preamble used in OFDMA backscatter; b) DigiScatter has a downlink symbol rate of 2MHz, which is 8× OFDMA backscatter ($250kHz$), but has a longer control frame; c) DigiScatter supports concurrency up to 1024 while OFDMA backscatter supports only 48.

■ **Concurrency examination in bin level.** We present the PHY bitrate of each subcarrier under different concurrency configurations. We examine the bin-level data of 100 tags in $2.4GHz$ band [62] with WARP and 300 tags in $900MHz$ with USRP E312 [47] as shown in Fig.16(a) and Fig.16(b). We also use 100 $2.4GHz$ tags to sweep the spectrum under different configurations, which are shown in Fig.16(c) and Fig.16(d). We calculate the theoretical value of PHY data rate, which is then compared with the data rate in the practical situation. Figure 16 shows that subcarriers around index 0 are unavailable because the transceiver IC we use (MAX2829) would filter out any signal within 100 kHz in baseband when receiving backscattered signals. This is a typical transceiver design to avoid saturation caused by high DC power, making several subcarriers unavailable in DigiScatter. Those subcarriers account for within 0.5% of all the subcarriers. For example, in 1024 concurrency configuration, there are 5 subcarriers unavailable, thus the number of concurrent backscatter transmissions is 1019.

7 DISCUSSIONS: OFDMA VS DCSS

Compared with the proof-of-concept OFDMA backscatter [5], DigiScatter emphasizes on practical large-scale prototyping of OFDMA backscatter networks, which provides more efficiency and flexibility. However, OFDMA backscatter design is not the only way for realizing large-scale backscatter networks. The DCSS can also help achieve the goal as mentioned. This section compares the OFDMA with DCSS based backscatter design in technical details, in order to make the big picture more complete.

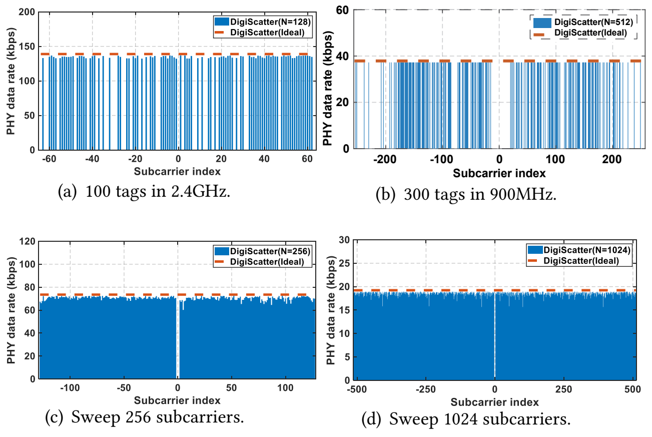


Figure 16: Concurrency examination in bin level.

■ **Bottlenecks of concurrency.** Comparing OFDM and DCSS based design, we can see that both of them essentially assign unique FFT bin to each tag. Their potentials for supporting large-scale concurrency are restricted by the hardware capability for tolerating the frequency mismatch. However, the main sources of the frequency mismatch for the two systems are different. For DigiScatter, oscillator jitter is the main source, thus the concurrency limitation can be calculated from the ppm value of the crystal oscillator, as analyzed in Section 2.1. For NetScatter, besides the oscillator jitter, the concurrency is also influenced by the hardware delay. In particular, NetScatter reports that the frequency offset is equivalent to 0.15 bin under $SF = 9$ [3], while the hardware delay can be as high as $3.5\mu s$, which is in fact more than 1 bin (2 μs) and close to 2 bins. This is caused by the characteristic of CSS modulation: timing mismatch can also incur frequency mismatch.

In the perspective of spetrum utilization, we can see that NetScatter realizes 256 concurrency with $500kHz$, while DigiScatter needs $20MHz$ to potentially realize 1019 concurrency. This is because concurrency in practice is more dependent on the hardware restriction; simply increasing bandwidth cannot provide higher concurrency for backscatter, because the hardware restriction scales up when increasing the bandwidth:

1) For DCSS based design, the hardware delay scales up with bandwidth increasing, thus doubling the bandwidth does not necessarily double the concurrency. NetScatter examines what will happen if BW increased from $500KHz$ to $1MHz$ and SF increased from 9 to 10; the result is that the MCU hardware delay can be up to $3.5\mu s$, which is equivalent to x bins where $x = BW \cdot t_{Delay} = 3.5$ [3]. This indicates that increasing the BW can incur more serious frequency mismatch. NetScatter uses guarding cyclic shifts to overcome the frequency mismatch, which means that the system will use only one out of every $SKIP + 1$ cyclic shifts among all availalbe cyclic shifts. If $x = 3.5$, then NetScatter must set $SKIP = 3$ at least, which makes the concurrency still $2^{SF}/(SKIP + 1) = 256$.

2) For both DCSS and OFDMA design, the clock frequency variation scales up with BW increases. The frequency synthesizer normally has a reference clock usually made of a crystal oscillator, where the error of the oscillator is profiled by the number of errors could occur in every 1 million oscillations. Increasing BW is equivalent to increasing sampling frequency f_s, which increases the frequency offset in Hz. For example, in DigiScatter, frequency offset caused by oscillator can be up to 0.25 bin under $f_s = 20MHz$ and the concurrency can be supported is 1024 theoretically. If we let $f_s = 40MHz$, with the bandwidth of the subcarrier unchanged, theoretically we could have the number of bins available doubled; however, the frequency offset will be also doubled (up to 0.5 bin), which can impact more bins in this case. Consequently, the concurrency does not necessarily double.

Moreover, the concurrency of the backscatter system is also influenced by the system's working frequency and transmitting power of the excitation signal transmitter in practice. In particular, NetScatter's working frequency is $900MHz$ and the excitation signal transmitting power is $30dBm$, while DigiScatter's working frequency is $2.4GHz$ and the transmitting power is $15dB$ less than the NetScatter case, which is due to the limitation of the WARP V3 platform. This makes NetScatter able to cover larger areas thus contain more concurrent tags. In contrast, DigiScatter's coverage is smaller, and it is difficult to successfully trigger tags located at the rim of the coverage.

■ **Modulation techniques.** DigiScatter adopts PSK (BPSK or QPSK) modulation, while NetScatter utilizes OOK and CSS. 1) Under the same transmission power, BPSK is 3-dB better than OOK based on communication theory. However, PSK needs to recover phases, requiring dedicated DSP; therefore, DigiScatter performs equalization and DD-PLL processing while NetScatter only needs to judge peaks in the spectrum. 2) Through using CSS, NetScatter can achieve high processing gain, which provides opportunity for long-range communication below the noise floor; however, using CSS means that NetScatter can only work with limited symbol rate without further optimization. As a result, the highest PHY data rate of a DigiScatter device is $620 \times$ the data rate of a NetScatter device.

■ **Dynamic data rate adaptation.** DigiScatter enables dynamic rate adaptation, which in fact is supported by the IDFT. The dynamic adaptation scheme is triggered by the transmitter, and the backscatter tags only provide an interface for downlink control. To the best of our knowledge, DigiScatter is the first backscatter system with such feature.

■ **Suitable deployment scene.** In contrast to NetScatter suitable for wider-area deployment, DigiScatter is unable to communicate below the noise floor due to the modulation characteristic and carrier frequency. Recall Section 6.2, we reveal that tags working in adjacent subcarriers under $N = 1024$ may have a maximum power impact of $-10.05dB$ to each other; therefore, DigiScatter suits dense deployment in indoor environment. This will avoid the near-far problem in contrast to NetScatter, because $-10.05dB$ means $25m+$ propagation [33] but the typical deployment diameter of DigiScatter is about $10m$, which means that two devices located at different places would not interfere with each other.

Summary: OFDMA and DCSS based backscatter design in principle can both support high concurrency, but restrained by various factors in the practical implementation. We would like to emphasize that enabling digital frequency synthesis is only one step forward for constructing large scale OFDMA backscatter systems. Many practical issues need to be resolved, which will be our future work.

8 RELATED WORK

NetScatter [3] and OFDMA backscatter design [5] are all for enhancing concurrency of backscatter systems. In fact, such trials have been made over the past decade. A number of approaches including TDMA, FDMA, SDMA and CDMA have all been tried for backscatter concurrency enhancement, where the design philosophy is to improve efficiency of collision avoidance [1, 14–21]. Recent works can decode collided parallel transmissions through analyzing the RF-powered tags' IQ features [1, 17, 19–21]. Early Wi-Fi backscatter tags are also purely powered by RF signals [7, 8]. While consuming energy in the microwatts level, the inherent low-efficient power modulation scheme [7] and self-interference with the excitation signal [8] incur limited data rate and backscatter range of the system, which can hardly support concurrency. OFDMA backscatter design is first presented in [5]. Besides LoRa backscatter [2] and NetScatter [3], PLoRa [6] is also a system utilizing CSS modulation and realizes long-range communication. Different from the previous two CSS based designs, PLoRa does not generate chirps on the tag; instead, it performs *blind chirp modulation* based on ambient LoRa packets using two different square waves for frequency shifting.

9 CONCLUSIONS

We have proposed DigiScatter, an OFDMA backscatter system enabling digital frequency synthesis. In DigiScatter, we for the first time integrate IDFT into the tag design; such a simple but effective improvement enables the system to support high concurrency and flexible spectrum resource allocation through pure software configurations in an online manner. We have built a testbed and conduct comprehensive experiments to validate our design. DigiScatter physically realizes 100 and 300 concurrent OFDMA backscatter transmissions in $2.4GHz$ and $900MHz$ respectively, and provides frequency synthesis capability for supporting 1019 concurrent transmissions.

10 ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their valuable comments and suggestions. The work in this paper is supported by the National Natural Science Foundation of China (No. 61922055, 61872233, 61829201, 61532012, 61325012, 61428205).

REFERENCES

[1] M. Jin, Y. He, X. Meng, D. Fang, and X. Chen. "Parallel Backscatter in the Wild: When Burstiness and Randomness Play with You," in *Proc. ACM MobiCom*, 2018.

[2] V. Talla, M. Hessar, B. Kellogg, A. Najafi, J. Smith and S. Gollakota, "LoRa backscatter: Enabling the vision of ubiquitous connectivity," in *Proc. Ubicomp*, 2017.

[3] M. Hessar , A. Najafi, S. Gollakota. "Netscatter: Enabling large-scale backscatter networks," in *Proc. NSDI*, 2019.

[4] R. Eletreby, D. Zhang, S. Kumar, and O. YaÄ§an. "Empowering low-power wide area networks in urban settings," in *Proc. ACM SIGCOMM*, 2017.

[5] R. Zhao, F. Zhu, Y. Feng, S. Peng, X. Tian, H. Yu and X. Wang, "OFDMA-Enabled Wi-Fi Backscatter,âĂİ in *Proc. ACM MobiCom*, 2019.

[6] Y. Peng, L. Shangguan, Y. Hu, Y. Qian, X. Lin, X. Chen, D. Fang, and K. Jamieson. "PLoRa: Passive long-range data networks from ambient lora transmissions," in *Proc. ACM SIGCOMM*, 2018.

[7] B. Kellogg, A. Parks, S. Gollakota, J. R. Smith and D. Wetherall, "Wi-Fi backscatter: Internet connectivity for rf-powered devices," in *Proc. ACM SIGCOMM*, 2014.

[8] D. Bharadia, K. Joshi, M. Kotaru and S. Katti, "BackFi: High throughput WiFi backscatter," in *Proc. ACM SIGCOMM*, 2015.

[9] B. Kellogg, V. Talla, S. Gollakota and J. R. Smith, "Passive Wi-Fi: Bringing low power to Wi-Fi transmissions," in *Proc. Usenix NSDI*, 2016.

[10] P. Zhang, D. Bharadia, K. Joshi, S. Katti1, "HitchHike: Practical backscatter using commodity WiFi," in *Proc. ACM SenSys*, 2016.

[11] P. Zhang, C. Josephson, D. Bharadia and S. Katti, "FreeRider: Backscatter communication using commodity radios," in *Proc. ACM CoNEXT*, 2017.

[12] P. Zhang, M. Rostami, P. Hu and D. Ganesan, "Enabling practical backscatter communication for on-body sensors," in *Proc. ACM SIGCOMM*, 2016.

[13] V. Iyer, V. Talla, B. Kellogg, S. Gollakota and J. R. Smith, "Inter-technology backscatter: Towards Internet connectivity for implanted devices," in *Proc. ACM SIGCOMM*, 2016.

[14] C. Mutti and C. Floerkemeier, "CDMA-based RFID systems in dense scenarios: Concepts and challenges," in *Proc. IEEE RFID*, 2008.

[15] J. Wang, H. Hassanieh, D. Katabi and P. Indyk, "Efficient and reliable low-power backscatter networks," in *Proc. ACM SIGCOMM*, 2012.

[16] L. Kong, L. He, Y. Gu, M. Wu and T. He, "A parallel identification protocol for RFID systems," in *Proc. IEEE INFOCOM*, 2012.

[17] P. Hu, P. Zhang, D. Ganesan, "Leveraging interleaved signal edges for concurrent backscatter," in *Proc. ACM HotWireless*, 2014

[18] P. Hu, P. Zhang, D. Ganesan, "Laissez-faire : Fully asymmetric backscatter communication," in *Proc. ACM SIGCOMM*, 2015.

[19] O. Abari, D. Vasisht, D. Katabi and A. Chandrakasan, "Caraoke: An E-toll transponder network for smart cities," in *Proc. ACM SIGCOMM*, 2015.

[20] J. Ou, M. Li, and Y. Zheng, "Come and be served: Parallel decoding for COTS RFID tags," in *Proc. ACM MobiCom*, 2015.

[21] M. Jin, Y. He, X. Meng, Y. Zheng, D. Fang and X. Chen, "FlipTracer: Practical parallel decoding for backscatter communication," in *Proc. ACM MobiCom*, 2017.

[22] Y. Hiraku, I. Hayashi, H. Chung, T. Kuroda, H. Ishikuro, "A 0.5 V 10MHz-to-100MHz 0.47 μz power scalable AD-PLL in 40nm CMOS," *Solid State Circuits Conference, 2012 IEEE Asian*, pp. 33–36, 2012.

[23] 802.11ax - Standard for Information Technology - Telecommunications and Information Exchange Between Systems Local and Metropolitan Area Networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment Enhancements for High Efficiency WLAN, Available Online: http://standards.ieee.org/develop/wg/WG802.11.html.

[24] 802.11ax: Transforming Wi-Fi to bring unprecedented capacity and efficiency, Qualcomm Technologies, Available Online: https://www.qualcomm.com/solutions/networking/features/80211ax.

[25] 802.11ax: Next generation Wi-Fi for the Gigabit home, Broadcom white paper, Available Online: https://www.mobileworldlive.com/broadcom-whitepaper-802-11ax-next-generation-wi-fi-for-the-gigabit-home/.

[26] Introduction to 802.11ax, National Instruments white paper, Available Online: http://www.ni.com/80211ax/.

[27] A. Wang, V. Iyer, V. Talla, J. R. Smith and S. Gollakota, "FM backscatter: Enabling connected cities and smart fabrics," in *Proc. NSDI*, 2017.

[28] V. Talla, B. Kellogg, S. Gollakota and J. R. Smith, "Battry-free cellphone," in *Proc. ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, vol. 1, no. 2, article 25, Jun. 2017.

[29] P. Zhang, P. Hu, V. Pasikanti and D. Ganesan, "EkhoNet: High speed ultra low-power backscatter for next generation sensors," in *Proc. ACM MobiCom*, 2014.

[30] V. Liu, A. Parks, V. Talla, S. Gollakota, D. Wetherall and J. R. Smith, "Ambient backscatter: Wireless communication out of thin air," in *Proc. ACM SIGCOMM*, 2013.

[31] IEEE Std 802.11-2007, "802.11b-1999 - IEEE Standard for Information Technology - Telecommunications and information exchange between systems - Local and Metropolitan networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher Speed Physical Layer (PHY) Extension in the 2.4 GHz band", Available Online: http://ieeexplore.ieee.org/document/817038/.

[32] T. Hwang, C. Yang, G. Wu, S. Li and G. Y. Li, "OFDM and its wireless applications: A survey," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 4, pp. 1673–1694, Aug. 2008.

[33] R. Akl, D. Tummala, X. Li, "Indoor Propagation Modeling at 2.4 GHz for IEEE 802.11 Networks," *Wireless and Optical Communications*, 2006.

[34] N. Sornin and L. Champion. Signal concentrator device, Oct. 17 2017. US Patent 9,794,095.

[35] Open source HitchHike platform, Available Online, https://github.com/pengyuzhang/HitchHike

[36] X. Zhou and C. Xie, Enabling Technologies for High Spectral-efficiency Coherent Optical Communication Networks, 1st ed. New York, NY, USA: Wiley, pp. 399–401, 2016.

[37] Clocking Wizard v6.0, LogiCORE IP Product Guide, Vivado Design Suite, Xilinx, Available online: https://www.xilinx.com/support/documentation/ip_documentation/clk_wiz/v6_0/pg065-clk-wiz.pdf

[38] MMCM and PLL Dynamic Reconfiguration, Xilinx, Available online: https://www.xilinx.com/support/documentation/application_notes/xapp888_7Series_DynamicRecon.pdf

[39] Digital clock manager (DCM), Xilinx, Available online: https://www.xilinx.com/support/documentation/application_notes/xapp462.pdf

[40] FFT v9.1 IP core, Xilinx, Available online: https://www.xilinx.com/support/documentation/ip_documentation/xfft/v9_1/pg109-xfft.pdf

[41] A. N. Parks, A. P. Sample, Y. Zhao and J. R. Smith. "A wireless sensing platform utilizing ambient RF energy," in *proc. IEEE Topical Meeting on Wireless Sensors and Sensor Networks (WiSNet 2013)*, Jan. 2013.

[42] K. H. Cheng, C. W. Lai, Y. L. Lo, "A cmos vco for 1v, 1ghz pll applications, " in *Proceedings of 2004 IEEE Asia-Pacific Conference on Advanced System Integrated Circuits*, 2004

[43] LTC6903/6904 1kHz to 68MHz Serial Port Programmable Oscillator, Linear Technology, https://www.analog.com/media/en/technical-documentation/data-sheets/69034fe.pdf

[44] Si514, Any-Frequency I^2C Programmable XO (100kHz o 250MHz), Silicon Labs, https://www.silabs.com/documents/public/data-sheets/Si514.pdf

[45] LMK61E2 Ultra-Low Jitter Programmable Oscillator With Internal EEPROM, Texas Instrument, http://www.ti.com/lit/ds/symlink/lmk61e2.pdf

[46] FFT HDL Optimized, Mathworks, Available online: https://ww2.mathworks.cn/help/dsp/ref/ffthdloptimized.html

[47] USRP E312 Software Defined Radio, Ettus Research, https://www.ettus.com/all-products/usrp-e312/

[48] Oscilloscope, DSOX3054T by Keysight, Available Online: https://literature.cdn.keysight.com/litweb/pdf/5992-0140EN.pdf?id=2545408

[49] WARP Project, Available Online: http://warpproject.org

[50] SPDT RF switch, HMC190BMS8E by ADI, http://www.analog.com/media/en/technical-documentation/data-sheets/hmc190b.pdf

[51] Power splitter/combiner, BP2U+ by Mini-Circuits, https://www.minicircuits.com/pdfs/BP2U+.pdf

[52] Transmission line calculator, TX-LINE by NI, Available Online: http://www.awrcorp.com/products/additional-products/tx-line-transmission-line-calculator

[53] SPST reflective switch, ADG902 by ADI, http://www.analog.com/media/en/technical-documentation/data-sheets/ADG901_902.pdf

[54] RF multiplexer, ADG904 by ADI, https://www.analog.com/media/en/technical-documentation/data-sheets/ADG904.pdf

[55] Breadboardable Spartan-7 FPGA module, CMOD-S7 by DIGILENT, https://reference.digilentinc.com/reference/programmable-logic/cmod-s7/reference-manual

[56] Xilinx 7 series FPGAs datasheet, https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf

[57] Spartan 7 FPGAs Datasheet: DC and AC Switching Characteristics, Xilinx, https://www.xilinx.com/support/documentation/data_sheets/ds189-spartan-7-data-sheet.pdf

[58] Virtex 7 T and XT FPGAs Datasheet: DC and AC Switching Characteristics, Xilinx, https://www.xilinx.com/support/documentation/data_sheets/ds183_Virtex_7_Data_Sheet.pdf

[59] Artix 7 FPGAs Datasheet: DC and AC Switching Characteristics, Xilinx, https://www.xilinx.com/support/documentation/data_sheets/ds181_Artix_7_Data_Sheet.pdf

[60] Kintex 7 FPGAs Datasheet: DC and AC Switching Characteristics, Xilinx, https://www.xilinx.com/support/documentation/data_sheets/ds182_Kintex_7_Data_Sheet.pdf

[61] Transceiver IC, MAX2829 by maxim integrated, https://datasheets.maximintegrated.com/en/ds/MAX2828-MAX2829.pdf

[62] DigiScatter 2.4GHz Demo Video, http://iiot.sjtu.edu.cn/xtian/video/DigiScatter.mp4